Open edX Enterprise Service Documentation

Release 3.56.10

edX Inc.

Sep 20, 2022

CONTENTS

Your project description goes here

Contents:

CHAPTER

ONE

ENTERPRISE

The `Open edx Enterprise Service` app provides enterprise features to the Open edX platform. The majority of these features are structured around the concept of an `Enterprise Customer`, which is an organization or a group of people that "consumes" courses published on the Open edX platform.

1.1 Documentation

Full documentation for the Open edX Enterprise Service can be found at http://open-edx-enterprise-service-documentation.readthedocs.io/en/latest/.

1.2 License

The code in this repository is licensed under the AGPL 3.0 unless otherwise noted.

Please see `LICENSE.txt` for details.

1.3 How To Contribute

Contributions are very welcome.

Please read How To Contribute for details.

Even though they were written with `edx-platform` in mind, the guidelines should be followed for Open edX code in general.

A Pull Request Description Template can be found at`PULL_REQUEST_TEMPLATE.md`_ - this template is automatically applied when you open a pull request from GitHub. Please make sure to include this template if submitting a pull request via other channels.

After submitting a pull request, please use the Github "Reviewers" widget to add relevant reviewers and track review process.

1.4 Reporting Security Issues

Please do not report security issues in public. Please email security@edx.org.

1.5 Getting Help

Have a question about this repository, or about Open edX in general? Please refer to this list of resources if you need any assistance.

CHAPTER

TWO

GETTING STARTED

2.1 Production setup

`Edx-enterprise` is developed as a pluggable application for the edx-platform and can't currently be used outside of `edx-platform`. `Edx-enterprise` is shipped with default edx-platform setup (see edx-platform requirements file), so new installs should already have it set up and enabled.

If you're migrating from an earlier (i.e. pre-Ficus) release, the only step you *might* have to do manually is to perform database migrations.

```
$ paver update_db

# Or use a more down-to-the-root command (replace aws with your version of config)
$ ./manage.py lms migrate --settings=aws
```

2.2 Local development

If you have not already done so, create/activate a virtualenv.

Alternatively, docker can be used to provide a containerized shell to run tests with.

```
$ make test-shell
```

Dependencies can be installed via the command below.

```
$ make requirements
```

Than you might want to run tests to make sure the setup went fine and there are no pre-existing problems (i.e. failed tests or quality checks)

```
$ make test-all
```

For details on performing other development-related tasks and high-level overview of `edx-enterprise` architecture and development principles, see *Development documentation*

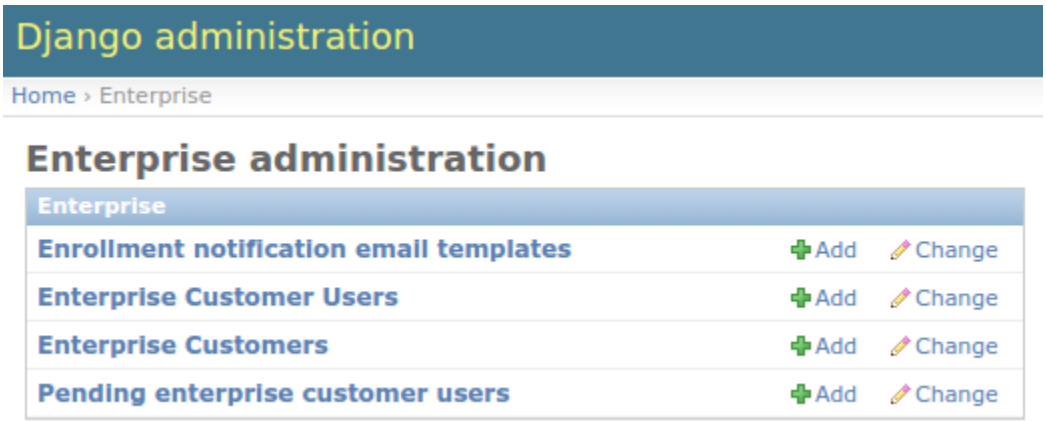
CHAPTER

THREE

CONFIGURATION AND USAGE

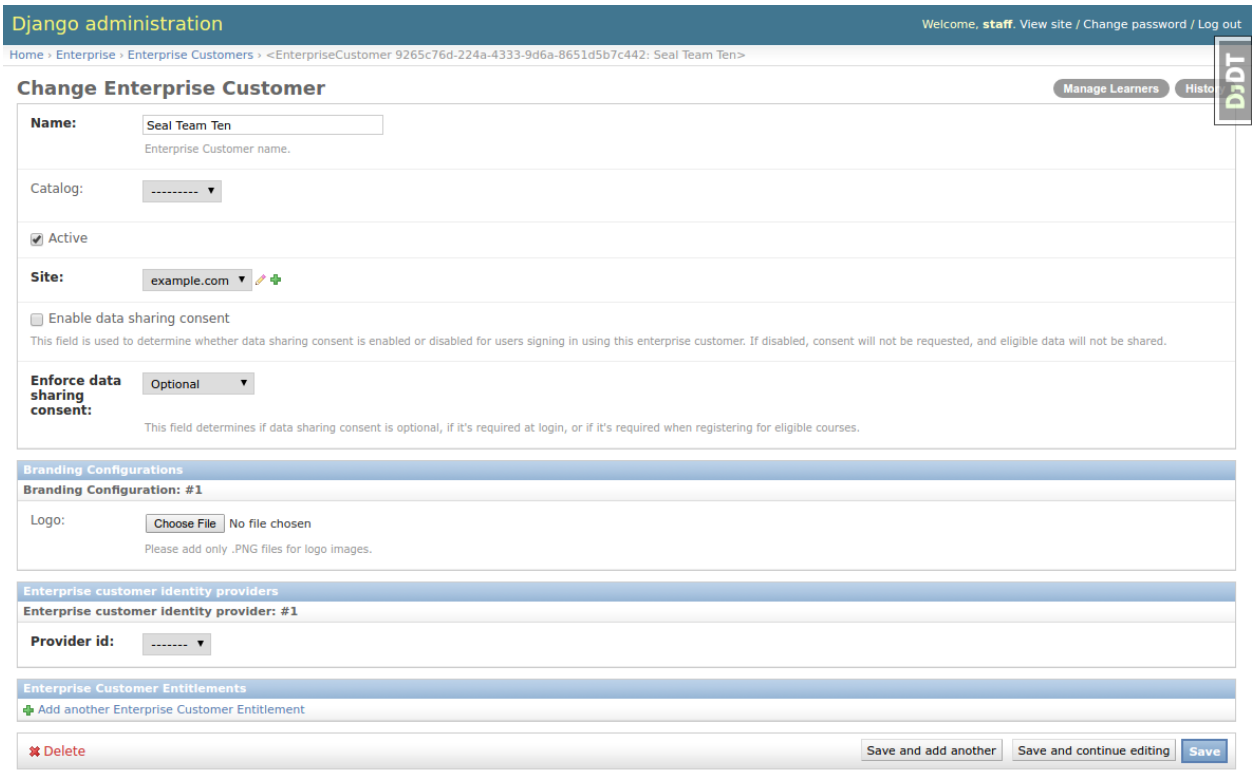
3.1 Getting around the administrator interface

The `edx-enterprise` project is integrated with `edx-platform` Django admin. It can be accessed at `$LMS_SERVER_ADDRESS/admin/enterprise`. The current administrator interface is considered provisional until an independent full-featured enterprise portal is built, so not much time was spent on improving admin tools UX.



So far, there are four items in the admin interface:

- "Enrollment notification email templates" - manages templates used to build enrollment notifications.
- "Enterprise Customers" - lists and manages Enterprise Customer records.
- "Enterprise Customer Users" - lists and manages students associated with Enterprise Customers.
- "Pending Enterprise Customer Users" - lists and manages "pending" Enterprise Customer students.

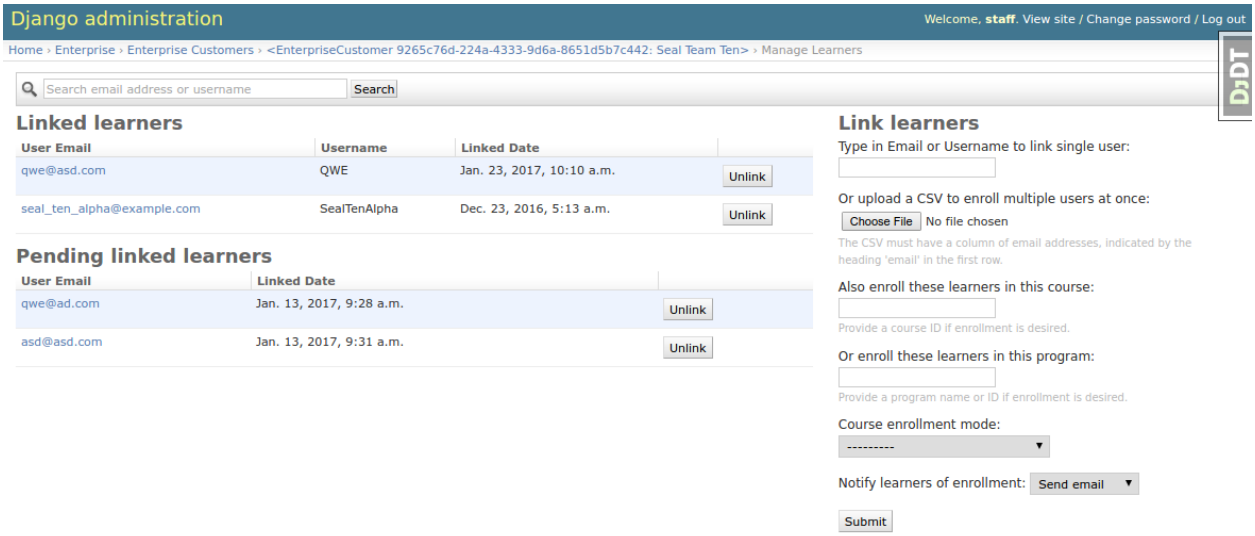


The `edx-enterprise` admin site provides two ways to manage learners: The "Enterprise Customer Users" section mentioned above, and the "Manage Learners" view. The "Manage Learners" view is recommended as it is easier to understand and provides more features. To access it, first go to the "Enterprise Customer" section of the admin site, then click on a customer, and then click on the "Manage Learners" button in the top-right corner of the page.

3.2 Manage Learners View

The Manage Learners view is meant to be used by non-technical staff to administer enterprise learners. It can be used to:

- Associate ("link") and disassociate ("unlink") existing students with an Enterprise Customer.
- Associate learners with an Enterprise customer even if they don't have an account yet (these learners are known as "Pending Enterprise Customer Learners"). Pending Enterprise Customer Learners are identified by their email address only.
- Manually enroll groups of enterprise learners into courses and/or programs.



3.2.1 Integrating with the Course Catalog

In order for course enrollment features to work, `edx-enterprise` talks to the Course Catalog Service. Instructions on how to set up and configure the Course Catalog service can be found at corresponding docs section of Course Catalog and in `edx-enterprise` Pull Request #7 (describes developer setup, but useful for a broader audience).

One particular quirk is that the Course Catalog Service, being an independent application, restricts access to courses and programs according to ownership rules, so only users with certain roles can list all programs and courses, as the `edx-enterprise` admin interface expects. This is covered in greater detail in aforementioned docs section.

3.2.2 Managing Linked Learners Lists

The Manage Learners view allows for two modes of linking users to an Enterprise Customer:

- Singular - A single user email address or username is provided.
- Bulk - A CSV file containing a single column of email addresses is uploaded.

When linking a single user by username, `edx-enterprise` tries to find an existing user with that username and fails the linking if a match was not found. When email address is used (either in singular or bulk mode), existing users using that email address are linked to the Enterprise Customer. If an email address match was not found, a "Pending Linked Learner" record is created for that email address. If that email address is used to register a new user, then that user is automatically linked with the Enterprise Customer.

Note: Each learner can be linked only to one Enterprise Customer, so linking fails if the learner is already associated with some other enterprise customer.

The Manage Learners view also enables unlinking "Linked Learners" and "Pending Linked Learners."

3.2.3 Enrolling Learners into a Course or Program

Enrolling learners into courses or programs depends on Course Catalog integration, because `edx-enterprise` uses the Course Catalog API to fetch course and program information.

When "Course ID" input is filled, "Program ID" input is blocked and "Course Enrollment Mode" is automatically populated with course modes available for chosen course.

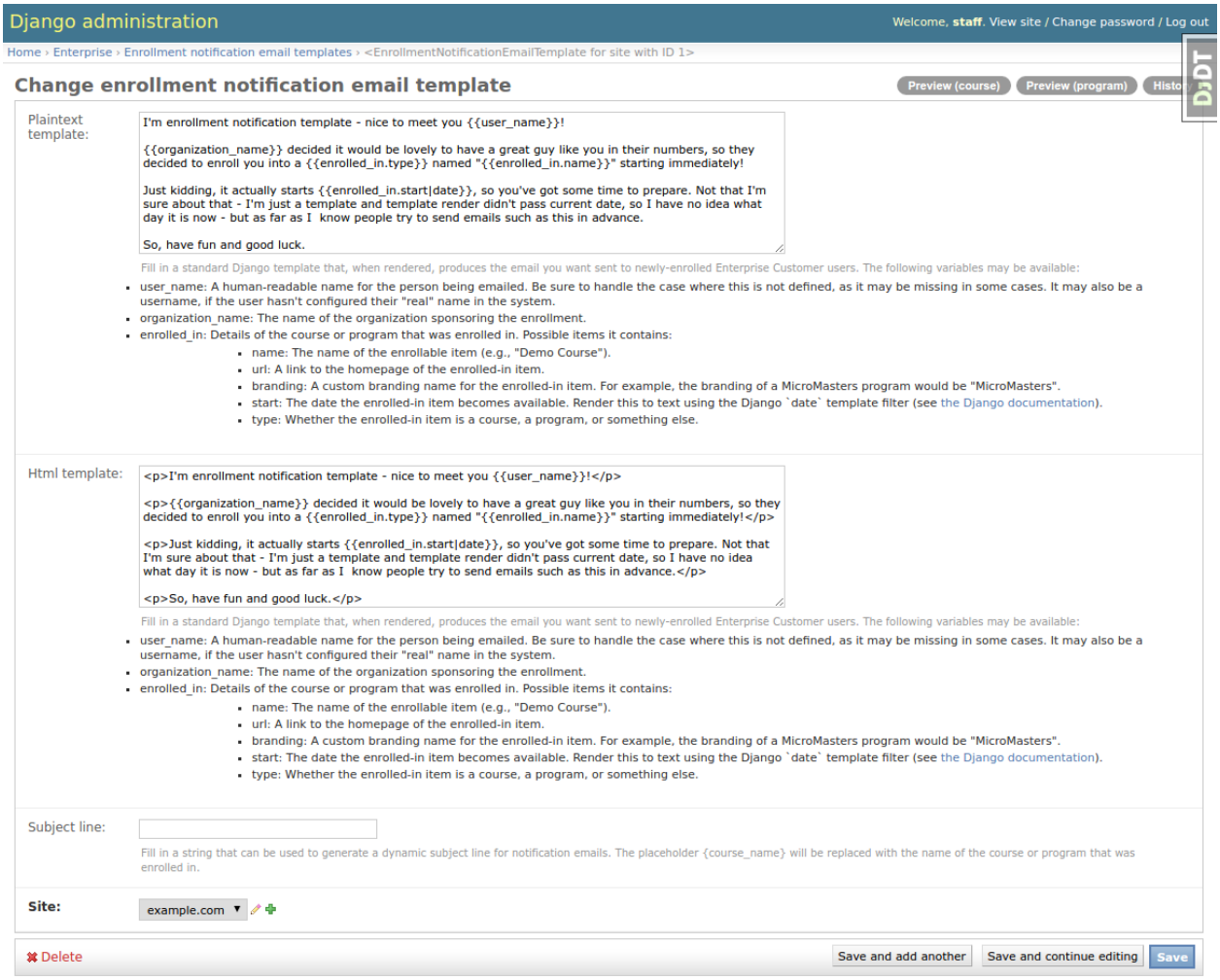
When "Program ID" input is filled, "Course ID" input is blocked and "Course Enrollment Mode" is reset to a list of all course enrollment modes[1]

3.3 Enrollment notification email templates

When learners are enrolled into a course or program, admins might choose to send enrollment notification emails. This admin section manages templates used to build such emails.

The template engine supports plaintext and HTML emails and allows substituting certain placeholders with real values, which are pulled from enrollment data (i.e. username, organization, course or program name, etc.). For details, refer to the help strings provided by the form.

[1] Course Catalog Service API does not expose any means to get a list of modes supported by *all* courses in the program, so it relies on the administrator to choose the right mode. However, validation is performed on the back end, and if an unavailable mode is chosen, a list of available modes is shown in an error message.



You can preview emails in the template edit view using the "Preview (program)" and "Preview (course)" buttons in top-right corner.

3.4 Integrated Channels

3.4.1 SAP SuccessFactors Configuration

One of the integrated channels available for an `EnterpriseCustomer` is SAP SuccessFactors.

To integrate an `EnterpriseCustomer` with SAP SuccessFactors:

- Set up a TPA SAML Integration for the SAP SuccessFactors endpoint.
- From the LMS Admin, ensure that your `EnterpriseCustomer` record has its `identity_provider` field pointing to the SAML Integration created in the previous step.

  This allows learners to create accounts using Single Sign-On from SAP SuccessFactors, and to have their course completion records sent back to SAP SuccessFactors.
- From the LMS Admin, create an `SAP SuccessFactors Enterprise Customer Configuration` record, and link it to your `EnterpriseCustomer`.

  Ensure that the record is marked `Active` to allow the management commands below to send data via this channel.

3.4.2 Management Commands

These Django management commands send data to `EnterpriseCustomer`'s active, integrated channels.

They must be run from the command line on a server with the Open edX environment installed.

Note: If data sharing consent is enabled for your `EnterpriseCustomer`, then any Enterprise learners enrolled in the associated courses must consent to data sharing, thus permitting their data to be sent back to SAP SuccessFactors.

If data sharing consent is *not* enabled for your `EnterpriseCustomer`, then learner data may be sent to SAP SuccessFactors without their explicit consent, so use these settings with care.

Transmit Learner Data

The `transmit_learner_data` command sends course completion data for each `EnterpriseCustomerCourseEnrollment` where evidence of course completion exists for an enrolled learner.

Note: "Course completion" is determined by differently for different types of courses.

- Instructor-paced course enrollments are deemed "complete" by the presence of a certificate for a given learner and course. The grade reported is pulled from the certificate.
- Self-paced courses with an end date are deemed "complete" once the end date has passed. The grade reported is "Pass" or "Fail", pulled from the Grades API, as of the current date.
- Self-paced courses with no end date are deemed "complete" once the learner passes the course. If the learner has not yet passed the course, the grade is reported as "In Progress".

Usage

```
# Login as the edxapp user, and enable the edxapp environment.
$ sudo -u edxapp -Hs
$ cd ~
$ source edxapp_env   # adds $EDX_PLATFORM_SETTINGS to the environment, e.g. aws,␣
↪openstack, devstack..

# View command help
$ ./manage.py lms transmit_learner_data --help --settings=$EDX_PLATFORM_SETTINGS

# Transmit learner data for all EnterpriseCustomers, to all active integrated channels.
# * --api_user must be a user with staff access to all the courses linked to the␣
↪EnterpriseCustomers.
$ ./manage.py lms transmit_learner_data --api_user staff --settings=$EDX_PLATFORM_
↪SETTINGS

# Transmit learner data for a single EnterpriseCustomer, e.g. with uuid 12
$ ./manage.py lms transmit_learner_data --api_user staff --enterprise-customer 12 --
↪settings=$EDX_PLATFORM_SETTINGS

# Transmit learner data only to SAP SuccessFactors
$ ./manage.py lms transmit_learner_data --api_user staff --channel SAP --settings=$EDX_
↪PLATFORM_SETTINGS
```

(continues on next page)

(continued from previous page)

CHAPTER

FOUR

DEVELOPMENT DOCUMENTATION

4.1 High-level Architecture Overview

`Edx-enterprise` is a pluggable Django application designed to be run within edx-platform. It is shipped with any edx-platform install, but can be disabled or even removed from a particular Open edX instance.

General direction is to decouple `edx-enterprise` from the rest of the edx-platform as much as possible, since the overall goal is to eventually make it an independent application or a plugin for a separate admin site, and not be part of the edx-platform LMS.

In order to maintain the separation of edx-enterprise from the edx-platform, follow these practices:

- When integrating with existing `edx-platform features`, use REST APIs rather than Python APIs.
- In cases where the API in question only exists as a Python API, it should be encapsulated into a module or class on the `edx-enterprise` side, to simplify future migration to a REST API.
- Both REST API and Python API clients must be covered with extensive unit tests.

Other noteworthy points:

- `edx-enterprise` does not interact with Studio.
- `edx-enterprise` supports multiple python versions:
  - Python 2.7 and python 3.5.
- High test coverage is expected - all new python code must have 100% diff coverage. JS test were only recently introduced, so it's not as strict so far, but aim for 100% diff coverage as well.
- High code quality standards are expected - both on python and JS sides. Multiple static analyzers are used to enforce it (`pycodestyle`, `pylint`, `isort`, `jshint`, etc.).

4.2 Local setup

Although `edx-enterprise` can't be used outside edx-platform, some development process steps are intended to be performed in a standalone virtual environment.

If you have not already done so, create/activate a virtualenv. Unless otherwise stated, assume all terminal code below is executed within the virtualenv.

Alternatively, docker can be used to provide a containerized shell to run the commands below inside.

```
$ make test-shell
```

4.2.1 Install dependencies

Dependencies can be installed via the command below.

```
$ make requirements
```

4.2.2 Run tests

This application uses tox to run tests under multiple Python and Django version. In addition, the following test utilities are used:

- pytest framework and test runner to run test
- pytest-cov plugin for collecting test coverage
- diff-cover for analyzing changeset coverage

The *edx-enterprise* Makefile provides multiple shortcuts to running tests. The most noteable are:

```
$ make test                     # run all tests in current virtualenv
$ make diff_cover               # run all tests in current virtualenv and report diff␣
↪coverage

# run quality checks, all tests in every supported env (Python and Django versions) and␣
↪jasmine JS tests
$ make test-all
```

More info about running tests is available in *Testing* section.

4.2.3 Check quality

A whole bunch of static analyzers are used to ensure code quality. Most notable are `pycodestyle` (formerly `pep8`), `pylint` and `jshint`.

```
$ tox -e quality    # runs all python and docs quality check
$ make jshint       # runs jshint on files in enterprise folder
```

4.2.4 Build (these) docs

This documentation is generated by Sphinx. Essentially, there are two parts of documentation - manually written (located in `docs` folder) and generated from docstrings.

```
$ tox -e docs   # performs docs quality check and generates docs in HTML
$ make docs     # same, but also opens browser at the index page
```

4.2.5 Update translations

See *Internationalization* chapter for details.

4.2.6 Upgrading local setup from older versions

If you're migrating from an older version (i.e. pre Nov 2016) of edx-platform, you might need to ensure edx-enterprise is installed correctly. Three things need to happen:

1. `edx-enterprise` must be installed in edxapp env.
2. `edx-enterprise` must be added to `INSTALLED_APPS`.
3. Migrations need to be run.

All three should happen automatically if you use paver commands to upgrade your setup, but just in case something goes wrong with the setup, here are instructions to manually perform the upgrade.

First, install `edx-enterprise` into virtualenv. In `edxapp` virtualenv (`$current_release` is 3.56.10)[1]

```
$ cd /edx/app/edxapp/edx-platform
$ pip install edx-enterprise==$current_release
```

Than, make sure `edx-enterprise` is included in `INSTALLED_APPS` or `OPTIONAL_APPS` (see lms/env/common.py as an example) and run migrations:

```
$ paver update_db
# Or use a more down-to-the-root command (replace aws with your version of config)
$ ./manage.py lms migrate --settings=devstack
```

[1] Due to limitations of Sphinx formatting, it is impossible to inject current version into code block while retaining formatting.

CHAPTER

FIVE

TESTING

edx-enterprise has an assortment of test cases and code quality checks to catch potential problems during development.

5.1 Running all of the python tests

To run all unit tests and quality checks in the version of Python you chose for your virtualenv.

Alternatively, docker can be used to provide a containerized shell to run the commands below inside.

```
$ make test-shell
```

```
$ make validate
```

To run just the unit tests:

```
$ make test
```

To run just the unit tests and check diff coverage

```
$ make diff_cover
```

To run the unit tests under every supported Python version and the code quality checks:

```
$ make test-all
```

To run all tests under certain python versions and edx-platform dependency environments:

```
$ tox -e py35-master   # run all tests under python 3.5 and master branch dependencies
```

5.2 Code coverage

To generate and open an HTML report of how much of the code is covered by test cases:

```
$ make coverage
```

There is a useful `pytest.local.ini` file that helps with looking at coverage of only a single module at a time:

```
$ pytest tests/test_enterprise/api -c pytest.local.ini --cov=enterprise.api
```

5.3 Running subsets of tests

Various options to run only subset of tests:

```
$ pytest tests/test_admin/              # run all tests in tests/admin folder
$ pytest tests/test_enterprise/api      # run all tests in tests/test_enterprise_api␣
↪folder
$ pytest tests/test_enterprise/api/test_permissions.py  # run all the tests in the test_
↪permissions.py file

# run all tests in TestEnterpriseCustomer test suite file
$ pytest tests/test_models.py::TestEnterpriseCustomer

# run only `test_ready_connects_user_post_save_handler` in `TestEnterpriseConfig` suite
$ pytest tests/test_apps.py::TestEnterpriseConfig::test_ready_connects_user_post_save_
↪handler
```

5.4 Quality

To run just the code quality checks:

```
$ tox -e quality
```

To run quality checks on specific files:

```
# run the PEP8-style checks on one file (fast)
$ pycodestyle enterprise/api/v1/views.py

# run pylint on one file (not as fast)
$ pylint enterprise/api/v1/views.py

# use isort to fix imports, --check-only means see what isort would change without␣
↪actually changing it
$ isort --check-only enterprise/api/v1/views.py

# use isort to actually change the file(s)
$ isort enterprise/api/v1/views.py enterprise/api/v1/permissions.py
```

CHAPTER

SIX

INTERNATIONALIZATION

All user-facing text content should be marked for translation. Even if this application is only run in English, our open source users may choose to use another language. Marking content for translation ensures our users have this choice.

Follow the internationalization coding guidelines in the edX Developer's Guide when developing new features.

6.1 Updating Translations

This project uses Transifex to translate content. After new features are developed the translation source files should be pushed to Transifex. Our translation community will translate the content, after which we can retrieve the translations.

Pushing source translation files to Transifex requires access to the edx-platform. Request access from the Open Source Team if you will be pushing translation files. You should also configure the Transifex client if you have not done so already.

The *make* targets listed below can be used to push or pull translations.

Target	Description
pull_translations	Pull translations from Transifex
push_translations	Push source translation files to Transifex

6.2 Fake Translations

As you develop features it may be helpful to know which strings have been marked for translation, and which are not. Use the *fake_translations* make target for this purpose. This target will extract all strings marked for translation, generate fake translations in the Esperanto (eo) language directory, and compile the translations.

You can trigger the display of the translations by setting your browser's language to Esperanto (eo), and navigating to a page on the site. Instead of plain English strings, you should see specially-accented English strings that look like this:

> Thé Fütüré øf Ønlïné Édüçätïøn # Før änýøné, änýwhéré, änýtïmé #

CHAPTER

SEVEN

SEGMENT EVENTS

`Edx-enterprise` emits several Segment events.

7.1 edx.bi.user.enterprise.onboarding

Emitted when a new `EnterpriseCourseEnrollment` record gets created.

The event contains these properties:

- **pathway**: A string identifying the mechanism through which the `EntepriseCourseEnrollment` was created. See below for possible values.
- **course_run_id**: EdX course ID associated with the enrollment.
- **url_path**: The url path the user was visiting when the enterprise course enrollment got created. Can be null when the enrollment is created outside of a learner's request (for example admin or REST API enrollments).

The possible values of the **pathway** property are:

- **admin-enrollment**: Enrollment was created by an admin via the "Manage Learners" tool.
- **pending-admin-enrollment**: Enrollment was created from a pending enrollment record previously set up by an admin via the "Manage Learners" tool.
- **rest-api-enrollment**: Enrollment was created via the REST API.
- **data-consent-page-enrollment**: Enrollment was created when learner submitted the form on the data sharing consent page.
- **course-landing-page-enrollment**: Enrollment was created when learner enrolled via the course landing page.
- **direct-audit-enrollment**: Enrollment was created via the direct audit enrollment mechanism.
- **customer-admin-enrollment**: Enrollment was created by a customer admin via the frontend app admin portal.

7.2 edx.bi.user.enterprise.enrollment.course

Emitted when the enterprise app enrolls a user into a course.

The event contains these properties:

- **label**: EdX course ID associated with the enrollment.
- **enterprise_customer_uuid**: UUID of the `EnterpriseCustomer` associated with the enrollment.
- **enterprise_customer_name**: Name of the `EnterpriseCustomer` associated with the enrollment.

- **mode**: The mode of the enrollment (audit, verified, etc.).

7.3 edx.bi.user.consent_form.shown

Emitted when the learner is presented with the data sharing consent page.

The event contains these properties:

- **deferCreation**: True when creation of course enrollment is deferred.
- **successUrl**: URL to redirect the user to if consent is provided.
- **failureUrl**: URL to redirect the user to in case consent is not provided.
- **courseId**: EdX course ID of the associated course.
- **programId**: ID of the associated program.

7.4 edx.bi.user.consent_form.accepted

Emitted after the learner grants data sharing consent.

The event contains the same properties as the *edx.bi.user.consent_form.shown* event.

7.5 edx.bi.user.consent_form.denied

Emitted after the learner denies data sharing consent.

The event contains the same properties as the *edx.bi.user.consent_form.shown* event.

CHAPTER

EIGHT

ENTERPRISE

8.1 enterprise package

8.1.1 Subpackages

enterprise.admin package

Submodules

enterprise.admin.actions module

Custom Django Admin actions used in enterprise app.

`enterprise.admin.actions.`**`export_as_csv_action`**(*description='Export selected objects as CSV file', fields=None, header=True*)

Return an export csv action.

Parameters

- **description** (*string*) – action description
- **fields** (*[string]*) – list of model fields to include
- **header** (*bool*) – whether or not to output the column names as the first row

`enterprise.admin.actions.`**`refresh_catalog`**(*self, request, queryset*)

Kicks off background running tasks for refreshing catalogs

enterprise.admin.forms module

Forms to be used in the enterprise djangoapp.

class `enterprise.admin.forms.`**`AdminNotificationForm`**(*data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, instance=None, use_required_attribute=None, renderer=None*)

Bases: `django.forms.models.ModelForm`

Alternate form for AdminNotification.

class Meta

Bases: object

fields = '__all__'

model

alias of *enterprise.models.AdminNotification*

widgets = {'text': <django.forms.widgets.Textarea object>}

base_fields = {'admin_notification_filter': <django.forms.models.ModelMultipleChoiceField object>, 'expiration_date': <django.forms.fields.DateField object>, 'is_active': <django.forms.fields.BooleanField object>, 'start_date': <django.forms.fields.DateField object>, 'text': <django.forms.fields.CharField object>, 'title': <django.forms.fields.CharField object>}

clean()

1. *start_date* and *expiration_date* are mandatory.
2. *start_date* must always come before the *expiration_date*.
3. There must be only one admin notification in a date range.

declared_fields = {}

property media

Return all media required to render the widgets on this form.

class enterprise.admin.forms.**BulkCatalogQueryUpdateCommandConfigurationAdmin**(*model*, *admin_site*)

Bases: config_models.admin.ConfigurationModelAdmin

Admin form for the BulkCatalogQueryUpdateCommandConfiguration model

property media

class enterprise.admin.forms.**EnterpriseCustomerAdminForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.models.ModelForm

Alternate form for the EnterpriseCustomer admin page.

class Meta

Bases: object

```
fields = ('name', 'slug', 'country', 'active', 'customer_type', 'site',
'enable_data_sharing_consent', 'enforce_data_sharing_consent',
'enable_audit_enrollment', 'enable_audit_data_reporting',
'replace_sensitive_sso_username', 'hide_course_original_price',
'enable_portal_code_management_screen',
'enable_portal_subscription_management_screen', 'enable_learner_portal',
'enable_learner_portal_offers',
'enable_portal_learner_credit_management_screen',
'enable_executive_education_2U_fulfillment', 'hide_labor_market_data',
'enable_integrated_customer_learner_portal_search', 'enable_analytics_screen',
'enable_portal_reporting_config_screen',
'enable_portal_saml_configuration_screen',
'enable_portal_lms_configurations_screen', 'enable_universal_link',
'enable_browse_and_request', 'enable_slug_login', 'contact_email',
'default_contract_discount', 'default_language', 'sender_alias', 'reply_to')
```

model

alias of *enterprise.models.EnterpriseCustomer*

```
base_fields = {'active': <django.forms.fields.BooleanField object>,
'contact_email': <django.forms.fields.EmailField object>, 'country':
<django_countries.fields.LazyTypedChoiceField object>, 'customer_type':
<django.forms.models.ModelChoiceField object>, 'default_contract_discount':
<django.forms.fields.DecimalField object>, 'default_language':
<django.forms.fields.TypedChoiceField object>, 'enable_analytics_screen':
<django.forms.fields.BooleanField object>, 'enable_audit_data_reporting':
<django.forms.fields.BooleanField object>, 'enable_audit_enrollment':
<django.forms.fields.BooleanField object>, 'enable_browse_and_request':
<django.forms.fields.BooleanField object>, 'enable_data_sharing_consent':
<django.forms.fields.BooleanField object>,
'enable_executive_education_2U_fulfillment': <django.forms.fields.BooleanField
object>, 'enable_integrated_customer_learner_portal_search':
<django.forms.fields.BooleanField object>, 'enable_learner_portal':
<django.forms.fields.BooleanField object>, 'enable_learner_portal_offers':
<django.forms.fields.BooleanField object>, 'enable_portal_code_management_screen':
<django.forms.fields.BooleanField object>,
'enable_portal_learner_credit_management_screen': <django.forms.fields.BooleanField
object>, 'enable_portal_lms_configurations_screen':
<django.forms.fields.BooleanField object>, 'enable_portal_reporting_config_screen':
<django.forms.fields.BooleanField object>,
'enable_portal_saml_configuration_screen': <django.forms.fields.BooleanField
object>, 'enable_portal_subscription_management_screen':
<django.forms.fields.BooleanField object>, 'enable_slug_login':
<django.forms.fields.BooleanField object>, 'enable_universal_link':
<django.forms.fields.BooleanField object>, 'enforce_data_sharing_consent':
<django.forms.fields.TypedChoiceField object>, 'hide_course_original_price':
<django.forms.fields.BooleanField object>, 'hide_labor_market_data':
<django.forms.fields.BooleanField object>, 'name': <django.forms.fields.CharField
object>, 'replace_sensitive_sso_username': <django.forms.fields.BooleanField
object>, 'reply_to': <django.forms.fields.EmailField object>, 'sender_alias':
<django.forms.fields.CharField object>, 'site':
<django.forms.models.ModelChoiceField object>, 'slug':
<django.forms.fields.SlugField object>}
```

```
declared_fields = {}
```

property media
Return all media required to render the widgets on this form.

class enterprise.admin.forms.**EnterpriseCustomerCatalogAdminForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Bases: `django.forms.models.ModelForm`

form for EnterpriseCustomerCatalogAdmin class.

class Meta
Bases: `object`

fields = '__all__'

model
alias of *enterprise.models.EnterpriseCustomerCatalog*

base_fields = {'content_filter': <jsonfield.forms.JSONField object>, 'enabled_course_modes': <jsonfield.forms.JSONField object>, 'enterprise_catalog_query': <django.forms.models.ModelChoiceField object>, 'enterprise_customer': <django.forms.models.ModelChoiceField object>, 'preview_button': <django.forms.fields.Field object>, 'publish_audit_enrollment_urls': <django.forms.fields.BooleanField object>, 'title': <django.forms.fields.CharField object>}

declared_fields = {'preview_button': <django.forms.fields.Field object>}

static get_catalog_preview_uuid(*post_data*)
Return the uuid of the catalog the preview button was clicked on There must be only one preview button in the POST data.

e.g: 'enterprise_customer_catalogs-0-preview_button'

property media
Return all media required to render the widgets on this form.

class enterprise.admin.forms.**EnterpriseCustomerIdentityProviderAdminForm**(**args*, ***kwargs*)

Bases: `django.forms.models.ModelForm`

Alternate form for the EnterpriseCustomerIdentityProvider admin page.

This form fetches identity providers from lms third_party_auth app. If third_party_auth app is not avilable it displays provider_id as a CharField.

class Meta
Bases: `object`

fields = '__all__'

model
alias of *enterprise.models.EnterpriseCustomerIdentityProvider*

base_fields = {'default_provider': <django.forms.fields.BooleanField object>, 'enterprise_customer': <django.forms.models.ModelChoiceField object>, 'provider_id': <django.forms.fields.SlugField object>}

clean()

Final validations of model fields.

1. Validate that selected site for enterprise customer matches with the selected identity provider's site.

declared_fields = {}

property media

Return all media required to render the widgets on this form.

class enterprise.admin.forms.**EnterpriseCustomerReportingConfigAdminForm**(*data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, instance=None, use_required_attribute=None, renderer=None*)

Bases: `django.forms.models.ModelForm`

Alternate form for the EnterpriseCustomerReportingConfiguration admin page.

This form uses the PasswordInput widget to obscure passwords as they are being entered by the user.

class Meta

Bases: `object`

fields = ('enterprise_customer', 'active', 'data_type', 'report_type', 'delivery_method', 'enable_compression', 'pgp_encryption_key', 'frequency', 'day_of_month', 'day_of_week', 'hour_of_day', 'include_date', 'email', 'decrypted_password', 'sftp_hostname', 'sftp_port', 'sftp_username', 'decrypted_sftp_password', 'sftp_file_path', 'enterprise_customer_catalogs')

model

alias of *enterprise.models.EnterpriseCustomerReportingConfiguration*

widgets = {'decrypted_password': <django.forms.widgets.PasswordInput object>, 'decrypted_sftp_password': <django.forms.widgets.PasswordInput object>}

base_fields = {'active': <django.forms.fields.BooleanField object>, 'data_type': <django.forms.fields.TypedChoiceField object>, 'day_of_month': <django.forms.fields.IntegerField object>, 'day_of_week': <django.forms.fields.TypedChoiceField object>, 'decrypted_password': <django.forms.fields.CharField object>, 'decrypted_sftp_password': <django.forms.fields.CharField object>, 'delivery_method': <django.forms.fields.TypedChoiceField object>, 'email': <multi_email_field.forms.MultiEmailField object>, 'enable_compression': <django.forms.fields.BooleanField object>, 'enterprise_customer': <django.forms.models.ModelChoiceField object>, 'enterprise_customer_catalogs': <django.forms.models.ModelMultipleChoiceField object>, 'frequency': <django.forms.fields.TypedChoiceField object>, 'hour_of_day': <django.forms.fields.IntegerField object>, 'include_date': <django.forms.fields.BooleanField object>, 'pgp_encryption_key': <django.forms.fields.CharField object>, 'report_type': <django.forms.fields.TypedChoiceField object>, 'sftp_file_path': <django.forms.fields.CharField object>, 'sftp_hostname': <django.forms.fields.CharField object>, 'sftp_port': <django.forms.fields.IntegerField object>, 'sftp_username': <django.forms.fields.CharField object>}

clean()
Override of clean method to perform additional validation

declared_fields = {'enterprise_customer_catalogs': <django.forms.models.ModelMultipleChoiceField object>}

property media
Return all media required to render the widgets on this form.

class enterprise.admin.forms.**EnterpriseFeatureUserRoleAssignmentForm**(**args*, ***kwargs*)
Bases: edx_rbac.admin.forms.UserRoleAssignmentAdminForm

Form for EnterpriseFeatureUserRoleAssignments.

class Meta
Bases: object

fields = ['user', 'role']

model
alias of *enterprise.models.EnterpriseFeatureUserRoleAssignment*

base_fields = {'role': <django.forms.models.ModelChoiceField object>, 'user': <edx_rbac.fields.UserFromEmailField object>}

declared_fields = {'user': <edx_rbac.fields.UserFromEmailField object>}

property media
Return all media required to render the widgets on this form.

class enterprise.admin.forms.**ManageLearnersDataSharingConsentForm**(**args*, ***kwargs*)
Bases: django.forms.forms.Form

Form to request DSC from a learner.

class Fields
Bases: object

Namespace class for field names.

COURSE = 'course'

EMAIL_OR_USERNAME = 'email_or_username'

base_fields = {'course': <django.forms.fields.CharField object>, 'email_or_username': <django.forms.fields.CharField object>}

clean_course()
Verify course ID has an associated course in LMS.

clean_email_or_username()
Verify email_or_username has associated user in our database.

declared_fields = {'course': <django.forms.fields.CharField object>, 'email_or_username': <django.forms.fields.CharField object>}

is_course_in_catalog(*course_id*)
Check whether course exists in enterprise customer catalog.

is_user_linked(*email*)
Check whether user is linked to the enterprise customer or not.

property media
Return all media required to render the widgets on this form.

class enterprise.admin.forms.**ManageLearnersForm**(**args*, ***kwargs*)
Bases: `django.forms.forms.Form`

Form to manage learner additions.

class CsvColumns
Bases: `object`

Namespace class for CSV column names.

COURSE_ID = 'course_id'

EMAIL = 'email'

class Fields
Bases: `object`

Namespace class for field names.

BULK_UPLOAD = 'bulk_upload_csv'

COURSE = 'course'

COURSE_MODE = 'course_mode'

DISCOUNT = 'discount'

EMAIL_OR_USERNAME = 'email_or_username'

GENERAL_ERRORS = '__all__'

MODE = 'mode'

NOTIFY = 'notify_on_enrollment'

REASON = 'reason'

SALES_FORCE_ID = 'sales_force_id'

class Modes
Bases: `object`

Namespace class for form modes.

MODE_BULK = 'bulk'

MODE_SINGULAR = 'singular'

class NotificationTypes

Bases: object

Namespace class for notification types

BY_EMAIL = 'by_email'

DEFAULT = 'by_email'

NO_NOTIFICATION = 'do_not_notify'

base_fields = {'bulk_upload_csv': <django.forms.fields.FileField object>, 'course': <django.forms.fields.CharField object>, 'course_mode': <django.forms.fields.ChoiceField object>, 'discount': <django.forms.fields.DecimalField object>, 'email_or_username': <django.forms.fields.CharField object>, 'notify_on_enrollment': <django.forms.fields.ChoiceField object>, 'reason': <django.forms.fields.CharField object>, 'sales_force_id': <django.forms.fields.CharField object>}

clean()

Clean fields that depend on each other.

In this case, the form can be used to link single user or bulk link multiple users. These are mutually exclusive modes, so this method checks that only one field is passed.

clean_course()

Verify course ID and retrieve course details.

clean_discount()

Verify that discount value should be from 0 to 100.

clean_email_or_username()

Clean email form field

Returns the cleaned value, converted to an email address (or an empty string)

Return type str

clean_notify()

Clean the notify_on_enrollment field.

clean_reason()

Clean the reason for enrollment field

declared_fields = {'bulk_upload_csv': <django.forms.fields.FileField object>, 'course': <django.forms.fields.CharField object>, 'course_mode': <django.forms.fields.ChoiceField object>, 'discount': <django.forms.fields.DecimalField object>, 'email_or_username': <django.forms.fields.CharField object>, 'notify_on_enrollment': <django.forms.fields.ChoiceField object>, 'reason': <django.forms.fields.CharField object>, 'sales_force_id': <django.forms.fields.CharField object>}

property media

Return all media required to render the widgets on this form.

class enterprise.admin.forms.**SystemWideEnterpriseUserRoleAssignmentForm**(**args*, ***kwargs*)

Bases: edx_rbac.admin.forms.UserRoleAssignmentAdminForm

Form for SystemWideEnterpriseUserRoleAssignments.

class Meta

Bases: object

fields = ['user', 'role', 'enterprise_customer', 'applies_to_all_contexts']

model
alias of *enterprise.models.SystemWideEnterpriseUserRoleAssignment*

base_fields = {'applies_to_all_contexts': <django.forms.fields.BooleanField object>, 'enterprise_customer': <django.forms.models.ModelChoiceField object>, 'role': <django.forms.models.ModelChoiceField object>, 'user': <edx_rbac.fields.UserFromEmailField object>}

declared_fields = {'user': <edx_rbac.fields.UserFromEmailField object>}

property media
Return all media required to render the widgets on this form.

class enterprise.admin.forms.**TransmitEnterpriseCoursesForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: `django.forms.forms.Form`

Form to transmit courses metadata for enterprise customers.

base_fields = {'channel_worker_username': <django.forms.fields.CharField object>}

clean_channel_worker_username()
Clean enterprise channel worker user form field

Returns the cleaned value of channel user username for transmitting courses metadata.

Return type str

declared_fields = {'channel_worker_username': <django.forms.fields.CharField object>}

property media
Return all media required to render the widgets on this form.

enterprise.admin.paginator module

Custom paginator to implement smart pagination.

class enterprise.admin.paginator.**CustomPaginator**(*object_list*, *per_page*, *orphans=0*, *allow_empty_first_page=True*)

Bases: `django.core.paginator.Paginator`

Adopted from django/core/paginator so as to implement smart links pagination in custom views.

property page_range
We have customized the getter so that it can return the value of the page_range property instead of always calculating the result.

enterprise.admin.utils module

Admin utilities.

class enterprise.admin.utils.**UrlNames**

Bases: object

Collection on URL names used in admin

MANAGE_LEARNERS = 'enterprise_manage_learners'

MANAGE_LEARNERS_DSC = 'enterprise_manage_learners_data_sharing_consent'

PREVIEW_EMAIL_TEMPLATE = 'enterprise_preview_email_template'

TRANSMIT_COURSES_METADATA = 'enterprise_transmit_courses_metadata'

URL_PREFIX = 'enterprise_'

enterprise.admin.utils.**email_or_username__to__email**(*email_or_username*)

Convert email_or_username to email.

Returns

If *email_or_username* was a username returns user's email, otherwise assumes it was an email and returns as is.

Return type str

enterprise.admin.utils.**paginated_list**(*object_list*, *page*, *page_size=25*)

Returns paginated list.

Parameters

- **object_list** (*QuerySet*) – A list of records to be paginated.
- **page** (*int*) – Current page number.
- **page_size** (*int*) – Number of records displayed in each paginated set.
- **show_all** (*bool*) – Whether to show all records.

Adopted from django/contrib/admin/templatetags/admin_list.py https://github.com/django/django/blob/1.11.1/django/contrib/admin/templatetags/admin_list.py#L50

enterprise.admin.utils.**parse_csv**(*file_stream*, *expected_columns=None*)

Parse csv file and return a stream of dictionaries representing each row.

First line of CSV file must contain column headers.

Parameters

- **file_stream** – input file
- **expected_columns** (*set[unicode]*) – columns that are expected to be present

Yields *dict* – CSV line parsed into a dictionary.

enterprise.admin.utils.**split_usernames_and_emails**(*email_field*)

Split the contents of the email field into a list.

In some cases, a user could enter a comma-separated value inline in the Manage Learners form. We should check to see if that's the case, and provide a list of email addresses or usernames if it is.

enterprise.admin.utils.**validate_csv**(*file_stream*, *expected_columns=None*)

Validate csv file for encoding and expected header.

Parameters

- **file_stream** – input file
- **expected_columns** – list of column names that are expected to be present in csv

Returns an iterable for csv datat if csv passes the validation

Return type reader

Raises **ValidationError** –

enterprise.admin.views module

Custom Django Admin views used in enterprise app.

class enterprise.admin.views.**BaseEnterpriseCustomerView**(***kwargs*)

Bases: `django.views.generic.base.View`

Base class for Enterprise Customer views.

get_form_view(*request*, *customer_uuid*, *additional_context=None*)

render the form with appropriate context.

template = None

class enterprise.admin.views.**EnterpriseCustomerManageLearnerDataSharingConsentView**(***kwargs*)

Bases: *`enterprise.admin.views.BaseEnterpriseCustomerView`*

Manage Learners Data Sharing Consent View.

Allows to request the DSC from a learner for a specific course.

class ContextParameters

Bases: `object`

Namespace-style class for custom context parameters.

ENTERPRISE_CUSTOMER = 'enterprise_customer'

MANAGE_LEARNERS_DSC_FORM = 'manage_learners_data_sharing_consent_form'

get(*request*, *customer_uuid*)

Handle GET request - render "Request a DSC from Learner" form.

Parameters

- **request** (*`django.http.request.HttpRequest`*) – Request instance
- **customer_uuid** (*`str`*) – Enterprise Customer UUID

Returns HttpResponse

Return type django.http.response.HttpResponse

post(*request*, *customer_uuid*)

Handle POST request - handle form submissions.

Parameters

- **request** (*`django.http.request.HttpRequest`*) – Request instance
- **customer_uuid** (*`str`*) – Enterprise Customer UUID

template = 'enterprise/admin/clear_learners_data_sharing_consent.html'

class enterprise.admin.views.**EnterpriseCustomerManageLearnersView**(***kwargs*)

Bases: *enterprise.admin.views.BaseEnterpriseCustomerView*

Manage Learners view.

Lists learners linked to chosen Enterprise Customer and allows adding and deleting them.

class ContextParameters

Bases: object

Namespace-style class for custom context parameters.

ENROLLMENT_URL = 'ENROLLMENT_API_ROOT_URL'

ENTERPRISE_CUSTOMER = 'enterprise_customer'

LEARNERS = 'learners'

MANAGE_LEARNERS_FORM = 'manage_learners_form'

PENDING_LEARNERS = 'pending_learners'

SEARCH_KEYWORD = 'search_keyword'

delete(*request*, *customer_uuid*)

Handle DELETE request - handle unlinking learner.

Parameters

- **request** (*django.http.request.HttpRequest*) – Request instance
- **customer_uuid** (*str*) – Enterprise Customer UUID

Returns HttpResponse

Return type django.http.response.HttpResponse

get(*request*, *customer_uuid*)

Handle GET request - render linked learners list and "Link learner" form.

Parameters

- **request** (*django.http.request.HttpRequest*) – Request instance
- **customer_uuid** (*str*) – Enterprise Customer UUID

Returns HttpResponse

Return type django.http.response.HttpResponse

get_enterprise_customer_user_queryset(*request*, *search_keyword*, *customer_uuid*, *page_size=25*)

Get the list of EnterpriseCustomerUsers we want to render.

Parameters

- **request** (*HttpRequest*) – HTTP Request instance.
- **search_keyword** (*str*) – The keyword to search for in users' email addresses and usernames.
- **customer_uuid** (*str*) – A unique identifier to filter down to only users linked to a
- **EnterpriseCustomer.** (*particular*) –
- **page_size** (*int*) – Number of learners displayed in each paginated set.

classmethod get_failed_enrollment_message(*users*, *enrolled_in*)

Create message for the users who were not able to be enrolled in a course.

Parameters

- **users** – An iterable of users who were not successfully enrolled
- **enrolled_in** (*str*) – A string identifier for the course with which enrollment was attempted

Returns: tuple: A 2-tuple containing a message type and message text

classmethod get_pending_enrollment_message(*pending_users*, *enrolled_in*)

Create message for the users who were enrolled in a course.

Parameters

- **users** – An iterable of PendingEnterpriseCustomerUsers who were successfully linked with a pending enrollment
- **enrolled_in** (*str*) – A string identifier for the course the pending users were linked to

Returns A 2-tuple containing a message type and message text

Return type tuple

get_pending_users_queryset(*search_keyword*, *customer_uuid*)

Get the list of PendingEnterpriseCustomerUsers we want to render.

Parameters

- **search_keyword** (*str*) – The keyword to search for in pending users' email addresses.
- **customer_uuid** (*str*) – A unique identifier to filter down to only pending users
- **EnterpriseCustomer.** (*linked to a particular*) –

get_search_keyword(*request*)

Retrieve the search querystring from the GET parameters.

classmethod get_success_enrollment_message(*users*, *enrolled_in*)

Create message for the users who were enrolled in a course.

Parameters

- **users** – An iterable of users who were successfully enrolled
- **enrolled_in** (*str*) – A string identifier for the course the users were enrolled in

Returns A 2-tuple containing a message type and message text

Return type tuple

post(*request*, *customer_uuid*)

Handle POST request - handle form submissions.

Parameters

- **request** (*django.http.request.HttpRequest*) – Request instance
- **customer_uuid** (*str*) – Enterprise Customer UUID

Returns HttpResponse

Return type django.http.response.HttpResponse

classmethod send_messages(*http_request*, *message_requests*)

Deduplicate any outgoing message requests, and send the remainder.

Parameters

- **http_request** – The HTTP request in whose response we want to embed the messages

- **message_requests** – A list of undeduplicated messages in the form of tuples of message type and text- for example, ('error', 'Something went wrong')

template = 'enterprise/admin/manage_learners.html'

class enterprise.admin.views.**EnterpriseCustomerTransmitCoursesView**(***kwargs*)

Bases: *enterprise.admin.views.BaseEnterpriseCustomerView*

Transmit courses view.

Allows transmitting of courses metadata for provided enterprise.

class ContextParameters

Bases: object

Namespace-style class for custom context parameters.

ENTERPRISE_CUSTOMER = 'enterprise_customer'

TRANSMIT_COURSES_METADATA_FORM = 'transmit_courses_metadata_form'

get(*request*, *customer_uuid*)

Handle GET request - render "Transmit courses metadata" form.

Parameters

- **request** (*django.http.request.HttpRequest*) – Request instance
- **customer_uuid** (*str*) – Enterprise Customer UUID

Returns HttpResponse

Return type django.http.response.HttpResponse

post(*request*, *customer_uuid*)

Handle POST request - handle form submissions.

Parameters

- **request** (*django.http.request.HttpRequest*) – Request instance
- **customer_uuid** (*str*) – Enterprise Customer UUID

template = 'enterprise/admin/transmit_courses_metadata.html'

class enterprise.admin.views.**TemplatePreviewView**(***kwargs*)

Bases: django.views.generic.base.View

Renders a given NotificationTemplate object to HTML for online viewing.

get(*request*, *template_id*, *view_type*)

Render the given template with the stock data.

static get_user_name(*request*)

Get a human-readable name for the user.

view_type_contexts = {'course': {'enrolled_in': {'name': 'OpenEdX Demo Course', 'start': datetime.datetime(2016, 1, 1, 0, 0), 'type': 'course', 'url': 'http://example.com/courses/edx-demo-course'}, 'organization_name': 'OpenEdX'}, 'program': {'enrolled_in': {'branding': 'MicroMasters', 'name': 'OpenEdX Demo Program', 'start': datetime.datetime(2016, 1, 1, 0, 0), 'type': 'program', 'url': 'http://example.com/programs/edx-demo-program'}, 'organization_name': 'OpenEdX'}}

enterprise.admin.widgets module

Widgets to be used in the enterprise djangoapp.

class enterprise.admin.widgets.**SubmitInput**(*attrs=None*)

Bases: `django.forms.widgets.Input`

Widget for input type field

input_type = 'submit'

property media

template_name = 'django/forms/widgets/text.html'

Module contents

Django admin integration for enterprise app.

class enterprise.admin.**AdminNotificationAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin for AdminNotification model.

filter_horizontal = ('admin_notification_filter',)

form

alias of *enterprise.admin.forms.AdminNotificationForm*

list_display = ('id', 'title', 'text', 'is_active', 'start_date', 'expiration_date', 'created', 'modified')

property media

model

alias of *enterprise.models.AdminNotification*

class enterprise.admin.**AdminNotificationFilterAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin for AdminNotificationFilter model.

list_display = ('id', 'filter', 'created', 'modified')

property media

model

alias of *enterprise.models.AdminNotificationFilter*

class enterprise.admin.**AdminNotificationReadAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin for AdminNotificationRead model.

list_display = ('id', 'enterprise_customer_user', 'admin_notification', 'is_read', 'created', 'modified')

property media

model

alias of *enterprise.models.AdminNotificationRead*

class enterprise.admin.**BigTableMysqlPaginator**(*object_list*, *per_page*, *orphans=0*, *allow_empty_first_page=True*)

Bases: `django.core.paginator.Paginator`

A paginator that uses INFORMATION_SCHEMA.TABLES to estimate the total number of rows in a table.

ARBITRARILY_LARGE_NUMBER = 10000000

property count

Returns the number of items in the object list (possibly an estimate).

class enterprise.admin.**EnrollmentNotificationEmailTemplateAdmin**(*model*, *admin_site*)

Bases: `django_object_actions.utils.DjangoObjectActions`, `django.contrib.admin.options.ModelAdmin`

Django admin for EnrollmentNotificationEmailTemplate model

class Meta

Bases: `object`

model

alias of *enterprise.models.EnrollmentNotificationEmailTemplate*

change_actions = ('preview_as_course', 'preview_as_program')

get_urls()

Returns the additional urls used by the custom object tools.

property media

preview(*obj*, *preview_type*)

Object tool handler method - redirects to "Preview" view

preview_as_course(*request*, *obj*)

Redirect to preview the HTML template in the context of a course.

preview_as_program(*request*, *obj*)

Redirect to preview the HTML template in the context of a program.

class enterprise.admin.**EnterpriseCatalogQueryAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCatalogQuery.

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseCatalogQuery*

discovery_query_url(*obj*)

Return discovery url for preview.

has_delete_permission(*request*, *obj=None*)

Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to delete the *obj* model instance. If *obj* is None, this should return True if the given request has permission to delete *any* object of the given type.

list_display = ('title', 'discovery_query_url')

property media

readonly_fields = ('discovery_query_url', 'uuid')

class enterprise.admin.**EnterpriseCourseEnrollmentAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCourseEnrollment

class Meta

Bases: `object`

model

alias of `enterprise.models.EnterpriseCourseEnrollment`

change_list_template = 'enterprise/admin/enterprise_course_enrollments_list.html'

changelist_view(*request*, *extra_context=None*)

Override to conditionally show the button.

get_urls()

Append *Enrollment Attribute Override* view url with default urls

has_add_permission(*request*)

Disable add permission for EnterpriseCourseEnrollment.

has_delete_permission(*request*, *obj=None*)

Disable deletion for EnterpriseCourseEnrollment.

license_uuid(*obj*)

Return the subscription license UUID (if any exists) associated with this enrollment.

list_display = ('enterprise_customer_user', 'course_id', 'saved_for_later')

property media

readonly_fields = ('enterprise_customer_user', 'course_id', 'saved_for_later', 'license_uuid')

search_fields = ('enterprise_customer_user__user_id', 'course_id')

class enterprise.admin.**EnterpriseCustomerAdmin**(*model*, *admin_site*)

Bases: `django_object_actions.utils.DjangoObjectActions`, `simple_history.admin.SimpleHistoryAdmin`

Django admin model for EnterpriseCustomer.

EXPORT_AS_CSV_FIELDS = ['name', 'active', 'site', 'uuid', 'identity_provider']

class Meta

Bases: `object`

model

alias of `enterprise.models.EnterpriseCustomer`

actions = [<function export_as_csv_action.<locals>.export_as_csv>]

change_actions = ('manage_learners', 'manage_learners_data_sharing_consent', 'transmit_courses_metadata')

change_view(*request*, *object_id*, *form_url=''*, *extra_context=None*)

enable_dsc(*instance*)

Return True if data sharing consent is enabled for EnterpriseCustomer.

Parameters **instance** (`enterprise.models.EnterpriseCustomer`) – *EnterpriseCustomer* model instance

form
alias of *enterprise.admin.forms.EnterpriseCustomerAdminForm*

get_form(*request*, *obj=None*, *change=False*, ***kwargs*)
Retrieve the appropriate form to use, saving the request user into the form for use in loading catalog details

get_search_results(*request*, *queryset*, *search_term*)
Return a tuple containing a queryset to implement the search and a boolean indicating if the results may contain duplicates.

get_urls()
Returns the additional urls used by the custom object tools.

has_identity_provider(*instance*)
Return True if EnterpriseCustomer has related identity provider.

Parameters **instance** (*enterprise.models.EnterpriseCustomer*) – *EnterpriseCustomer* model instance

has_logo(*instance*)
Return True if EnterpriseCustomer has a logo.

Parameters **instance** (*enterprise.models.EnterpriseCustomer*) – *EnterpriseCustomer* model instance

inlines = [<class 'enterprise.admin.EnterpriseCustomerBrandingConfigurationInline'>, <class 'enterprise.admin.EnterpriseCustomerIdentityProviderInline'>, <class 'enterprise.admin.EnterpriseCustomerCatalogInline'>, <class 'enterprise.admin.PendingEnterpriseCustomerAdminUserInline'>]

list_display = ('name', 'slug', 'customer_type', 'site', 'country', 'active', 'has_logo', 'enable_dsc', 'has_identity_provider', 'uuid')

list_filter = ('active',)

manage_learners(*request*, *obj*)
Object tool handler method - redirects to "Manage Learners" view

manage_learners_data_sharing_consent(*request*, *obj*)
Object tool handler method - redirects to "Clear Learners Data Sharing Consent" view

property media

ordering = ('name',)

search_fields = ('name', 'uuid')

transmit_courses_metadata(*request*, *obj*)
Object tool handler method - redirects to *Transmit Courses Metadata* view.

class enterprise.admin.**EnterpriseCustomerBrandingConfigurationInline**(*parent_model*, *admin_site*)

Bases: `django.contrib.admin.options.StackedInline`

Django admin model for EnterpriseCustomerBrandingConfiguration.

The admin interface has the ability to edit models on the same page as a parent model. These are called inlines. https://docs.djangoproject.com/en/1.8/ref/contrib/admin/#django.contrib.admin.StackedInline

can_delete = False

property media

model
alias of *enterprise.models.EnterpriseCustomerBrandingConfiguration*

class enterprise.admin.**EnterpriseCustomerCatalogAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCustomerCatalog.

class Media

Bases: `object`

js = ('enterprise/admin/enterprise_customer_catalog.js',)

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseCustomerCatalog*

actions = [<function refresh_catalog>]

fields = ('title', 'enterprise_customer', 'enterprise_catalog_query', 'content_filter', 'enabled_course_modes', 'publish_audit_enrollment_urls')

get_actions(*request*)

Disallow the delete selected action as that does not send a DELETE request to enterprise-catalog

get_form(*request*, *obj=None*, *change=False*, ***kwargs*)

Return a Form class for use in the admin add view. This is used by add_view and change_view.

list_display = ('uuid_nowrap', 'enterprise_customer', 'title', 'preview_catalog_url')

property media

ordering = ('enterprise_customer__name', 'title')

preview_catalog_url(*obj*)

Return enterprise catalog url for preview.

readonly_fields = ('preview_catalog_url',)

search_fields = ('uuid', 'title', 'enterprise_customer__name', 'enterprise_customer__uuid')

uuid_nowrap(*obj*)

Inject html for disabling wrap for uuid

class enterprise.admin.**EnterpriseCustomerCatalogInline**(*parent_model*, *admin_site*)

Bases: `django.contrib.admin.options.TabularInline`

Django admin model for EnterpriseCustomerCatalog. The admin interface has the ability to edit models on the same page as a parent model. These are called inlines. https://docs.djangoproject.com/en/1.8/ref/contrib/admin/#django.contrib.admin.StackedInline

can_delete = False

extra = 0

form

alias of *enterprise.admin.forms.EnterpriseCustomerCatalogAdminForm*

get_formset(*request*, *obj=None*, ***kwargs*)

Return a BaseInlineFormSet class for use in admin add/change views.

property media

model

alias of *enterprise.models.EnterpriseCustomerCatalog*

class enterprise.admin.**EnterpriseCustomerIdentityProviderInline**(*parent_model*, *admin_site*)

Bases: `django.contrib.admin.options.StackedInline`

Django admin model for EnterpriseCustomerIdentityProvider.

The admin interface has the ability to edit models on the same page as a parent model. These are called inlines. https://docs.djangoproject.com/en/1.8/ref/contrib/admin/#django.contrib.admin.StackedInline

extra = 0

form

alias of *enterprise.admin.forms.EnterpriseCustomerIdentityProviderAdminForm*

property media

model

alias of *enterprise.models.EnterpriseCustomerIdentityProvider*

class enterprise.admin.**EnterpriseCustomerInviteKeyAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCustomerInviteKey.

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseCustomerInviteKey*

fields = ('enterprise_customer', 'usage_count', 'usage_limit', 'expiration_date', 'is_active')

get_readonly_fields(*request*, *obj=None*)

Hook for specifying custom readonly fields.

list_display = ('uuid', 'enterprise_customer_id', 'usage_limit', 'expiration_date', 'is_active')

list_filter = ('is_active',)

property media

readonly_fields = ('uuid', 'usage_count')

search_fields = ('uuid__startswith', 'enterprise_customer__name__startswith')

class enterprise.admin.**EnterpriseCustomerReportingConfigurationAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCustomerReportingConfiguration.

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseCustomerReportingConfiguration*

autocomplete_fields = ['enterprise_customer']

form

alias of *enterprise.admin.forms.EnterpriseCustomerReportingConfigAdminForm*

get_fields(*request*, *obj=None*)

Return the fields that should be displayed on the admin form.

list_display = ('enterprise_customer', 'active', 'delivery_method', 'frequency', 'data_type', 'report_type')

list_filter = ('active',)

property media

ordering = ('enterprise_customer__name',)

search_fields = ('enterprise_customer__name', 'email')

class enterprise.admin.**EnterpriseCustomerTypeAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCustomerType.

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseCustomerType*

fields = ('name',)

list_display = ('name',)

property media

search_fields = ('name',)

class enterprise.admin.**EnterpriseCustomerUserAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for EnterpriseCustomerUser.

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseCustomerUser*

enterprise_enrollments(*enterprise_customer_user*)

Return a string representing a given EnterpriseCustomerUser's enterprise course enrollments

Parameters **enterprise_customer_user** – The instance of EnterpriseCustomerUser being rendered with this admin form.

fields = ('user_id', 'enterprise_customer', 'user_email', 'username', 'created', 'enterprise_enrollments', 'other_enrollments', 'invite_key', 'active', 'should_inactivate_other_customers')

get_enrolled_course_string(*course_ids*)

Get an HTML string representing the courses the user is enrolled in.

get_enterprise_customer(*obj*)

Returns the name of enterprise customer linked with the enterprise customer user.

get_readonly_fields(*request*, *obj=None*)

Make all fields readonly when editing existing model.

get_search_results(*request*, *queryset*, *search_term*)

Return a tuple containing a queryset to implement the search and a boolean indicating if the results may contain duplicates.

list_display = ('username', 'user_email', 'get_enterprise_customer')

property media

other_enrollments(*enterprise_customer_user*)

Return a string representing a given EnterpriseCustomerUser's non-enterprise course enrollments

Parameters **enterprise_customer_user** – The instance of EnterpriseCustomerUser being rendered with this admin form.

readonly_fields = ('user_email', 'username', 'created', 'enterprise_enrollments', 'other_enrollments')

search_fields = ('user_id',)

class enterprise.admin.**EnterpriseFeatureUserRoleAssignmentAdmin**(*model*, *admin_site*)

Bases: `edx_rbac.admin.UserRoleAssignmentAdmin`

Django admin model for EnterpriseFeatureUserRoleAssignment.

class Meta

Bases: `object`

model

alias of *enterprise.models.EnterpriseFeatureUserRoleAssignment*

form

alias of *enterprise.admin.forms.EnterpriseFeatureUserRoleAssignmentForm*

property media

class enterprise.admin.**PendingEnrollmentAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for PendingEnrollment

class Meta

Bases: `object`

model

alias of *enterprise.models.PendingEnrollment*

has_add_permission(*request*)

Disable add permission for PendingEnrollment.

has_delete_permission(*request*, *obj=None*)

Disable deletion for PendingEnrollment.

list_display = ('user', 'course_id', 'course_mode')

property media

readonly_fields = ('user', 'course_id', 'course_mode')

search_fields = ('user__user_email', 'course_id')

class enterprise.admin.**PendingEnterpriseCustomerAdminUserAdmin**(*model*, *admin_site*)

Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for PendingEnterpriseCustomerAdminUser

class Meta

Bases: `object`

model

alias of *enterprise.models.PendingEnterpriseCustomerAdminUser*

fields = ('user_email', 'enterprise_customer', 'get_admin_registration_url')

get_admin_registration_url(*obj*)
Formats the `admin_registration_url` model property as an HTML link.

get_enterprise_customer(*obj*)
Returns the name of the associated EnterpriseCustomer.

list_display = ('user_email', 'get_enterprise_customer', 'get_admin_registration_url')

property media

readonly_fields = ('get_admin_registration_url',)

search_fields = ('user_email', 'enterprise_customer__name')

class enterprise.admin.**PendingEnterpriseCustomerAdminUserInline**(*parent_model*, *admin_site*)
Bases: `django.contrib.admin.options.TabularInline`

Django admin inline model for PendingEnterpriseCustomerAdminUser.

extra = 0

fieldsets = ((None, {'fields': ('user_email', 'get_admin_registration_url')}),)

get_admin_registration_url(*obj*)
Formats the `admin_registration_url` model property as an HTML link.

property media

model
alias of *enterprise.models.PendingEnterpriseCustomerAdminUser*

readonly_fields = ('get_admin_registration_url',)

class enterprise.admin.**PendingEnterpriseCustomerUserAdmin**(*model*, *admin_site*)
Bases: `django.contrib.admin.options.ModelAdmin`

Django admin model for PendingEnterpriseCustomerUser

class Meta
Bases: `object`

model
alias of *enterprise.models.PendingEnterpriseCustomerUser*

fields = ('user_email', 'enterprise_customer', 'created')

property media

readonly_fields = ('user_email', 'enterprise_customer', 'created')

class enterprise.admin.**SystemWideEnterpriseUserRoleAssignmentAdmin**(*model*, *admin_site*)
Bases: `edx_rbac.admin.UserRoleAssignmentAdmin`

Django admin model for SystemWideEnterpriseUserRoleAssignment.

class Meta
Bases: `object`

model
alias of *enterprise.models.SystemWideEnterpriseUserRoleAssignment*

effective_enterprise_customer(*instance*)
Return the comma separated names of all enterprise customers attached to the user.

Parameters **instance** (`SystemWideEnterpriseUserRoleAssignment`) – model instance

fields = ('user', 'role', 'enterprise_customer', 'applies_to_all_contexts', 'effective_enterprise_customer')

form
alias of *enterprise.admin.forms.SystemWideEnterpriseUserRoleAssignmentForm*

get_form(*request*, *obj=None*, ***kwargs*)
Adds help text to the callable-defined `effective_enterprise_customer` field.

list_display = ('user', 'role', 'enterprise_customer', 'applies_to_all_contexts')

list_per_page = 25

list_select_related = ('user', 'role', 'enterprise_customer')

property media

paginator
alias of *enterprise.admin.BigTableMysqlPaginator*

readonly_fields = ('effective_enterprise_customer',)

search_fields = ('user__email', 'role__name', 'enterprise_customer__name')

show_full_result_count = False

enterprise.api package

Subpackages

enterprise.api.v1 package

Submodules

enterprise.api.v1.decorators module

Decorators for Enterprise API views.

`enterprise.api.v1.decorators.`**require_at_least_one_query_parameter**(**query_parameter_names*)
Ensure at least one of the specified query parameters are included in the request.

This decorator checks for the existence of at least one of the specified query parameters and passes the values as function parameters to the decorated view. If none of the specified query parameters are included in the request, a ValidationError is raised.

Usage:

```
@require_at_least_one_query_parameter('program_uuids', 'course_run_ids')
def my_view(request, program_uuids, course_run_ids):
    # Some functionality ...
```

enterprise.api.v1.fields module

Fields for Enterprise API serializers.

class enterprise.api.v1.fields.**Base64EmailCSVField**(**args*, ***kwargs*)

Bases: rest_framework.fields.Field

Serializers a Base64 encoded CSV with emails into an array of emails

to_internal_value(*data*)

Transform the *incoming* primitive data into a native value.

to_representation(*value*)

Transform the *outgoing* native value into primitive data.

enterprise.api.v1.permissions module

Custom API permissions.

class enterprise.api.v1.permissions.**IsInEnterpriseGroup**

Bases: rest_framework.permissions.BasePermission

Find out if the requesting user belongs to a django group meant for granting access to an enterprise feature. This check applies to both staff and non-staff users.

ALLOWED_API_GROUPS = []

has_permission(*request*, *view*)

Return *True* if permission is granted, *False* otherwise.

message = 'User is not allowed to access the view.'

enterprise.api.v1.serializers module

Serializers for enterprise api version 1.

class enterprise.api.v1.serializers.**AdminNotificationSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for AdminNotification model.

class Meta

Bases: object

fields = ('id', 'title', 'text')

model

alias of *enterprise.models.AdminNotification*

class enterprise.api.v1.serializers.**BaseEnterpriseCustomerInviteKeySerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Base serializer for writing to the EnterpriseCustomerInviteKey model.

class Meta

Bases: object

fields = ('uuid', 'enterprise_customer_uuid', 'usage_limit', 'expiration_date', 'is_active', 'is_valid')

model
alias of *enterprise.models.EnterpriseCustomerInviteKey*

class enterprise.api.v1.serializers.**CourseDetailSerializer**(**args*, ***kwargs*)
Bases: *enterprise.api.v1.serializers.ImmutableStateSerializer*

Serializer for course data retrieved from the discovery service course detail API endpoint.

This serializer updates the course and course run data with the EnterpriseCustomer-specific enrollment page URL for the given course and course runs.

to_representation(*instance*)
Return the updated course data dictionary.

Parameters **instance** (*dict*) – The course data.

Returns The updated course data.

Return type dict

class enterprise.api.v1.serializers.**CourseRunDetailSerializer**(**args*, ***kwargs*)
Bases: *enterprise.api.v1.serializers.ImmutableStateSerializer*

Serializer for course run data retrieved from the discovery service course_run detail API endpoint.

This serializer updates the course run data with the EnterpriseCustomer-specific enrollment page URL for the given course run.

to_representation(*instance*)
Return the updated course run data dictionary.

Parameters **instance** (*dict*) – The course run data.

Returns The updated course run data.

Return type dict

class enterprise.api.v1.serializers.**EnterpriseCourseEnrollmentReadOnlySerializer**(**args*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

Serializer for EnterpriseCourseEnrollment model.

class Meta
Bases: object

fields = ('enterprise_customer_user', 'course_id')

model
alias of *enterprise.models.EnterpriseCourseEnrollment*

class enterprise.api.v1.serializers.**EnterpriseCourseEnrollmentWriteSerializer**(**args*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

Serializer for writing to the EnterpriseCourseEnrollment model.

class Meta
Bases: object

fields = ('username', 'course_id')

model
alias of *enterprise.models.EnterpriseCourseEnrollment*

enterprise_customer_user = None

save()
Save the model with the found EnterpriseCustomerUser.

validate_username(*value*)
Verify that the username has a matching user, and that the user has an associated EnterpriseCustomerUser.

class enterprise.api.v1.serializers.**EnterpriseCustomerBasicSerializer**(**args*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

Serializer for EnterpriseCustomer model only for name and id fields.

class Meta
Bases: object

fields = ('id', 'name')

model
alias of *enterprise.models.EnterpriseCustomer*

class enterprise.api.v1.serializers.**EnterpriseCustomerBrandingConfigurationSerializer**(**args*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

Serializer for EnterpriseCustomerBrandingConfiguration model.

class Meta
Bases: object

fields = ('enterprise_customer', 'enterprise_slug', 'logo', 'primary_color', 'secondary_color', 'tertiary_color')

model
alias of *enterprise.models.EnterpriseCustomerBrandingConfiguration*

get_enterprise_customer(*obj*)
Return a string representation of the associated enterprise customer's UUID.

get_enterprise_slug(*obj*)
Return the slug of the associated enterprise customer.

get_logo(*obj*)
Use EnterpriseCustomerBrandingConfiguration.safe_logo_url to return an absolute URL for either the saved customer logo or the platform logo by default

get_primary_color(*obj*)
Return the primary color of the branding config OR the default primary color code

get_secondary_color(*obj*)
Return the secondary color of the branding config OR the default secondary color code

get_tertiary_color(*obj*)
Return the tertiary color of the branding config OR the default tertiary color code

class enterprise.api.v1.serializers.**EnterpriseCustomerBulkEnrollmentsSerializer**(**args*, ***kwargs*)
Bases: rest_framework.serializers.Serializer

Serializes a email_csv or email field for bulk enrollment requests.

create(*validated_data*)

validate(*data*)

class enterprise.api.v1.serializers.**EnterpriseCustomerBulkSubscriptionEnrollmentsSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

Serializes a licenses info field for bulk enrollment requests.

create(*validated_data*)

validate(*data*)

class enterprise.api.v1.serializers.**EnterpriseCustomerCatalogDetailSerializer**(**args*, ***kwargs*)

Bases: *enterprise.api.v1.serializers.EnterpriseCustomerCatalogSerializer*

Serializer for the EnterpriseCustomerCatalog model which includes the catalog's discovery service search query results.

to_representation(*instance*)

Serialize the EnterpriseCustomerCatalog object.

Parameters **instance** (EnterpriseCustomerCatalog) – The EnterpriseCustomerCatalog to serialize.

Returns The EnterpriseCustomerCatalog converted to a dict.

Return type dict

class enterprise.api.v1.serializers.**EnterpriseCustomerCatalogSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for the EnterpriseCustomerCatalog model.

class Meta

Bases: object

fields = ('uuid', 'title', 'enterprise_customer')

model

alias of *enterprise.models.EnterpriseCustomerCatalog*

class enterprise.api.v1.serializers.**EnterpriseCustomerCourseEnrollmentsListSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ListSerializer

Serializes a list of enrollment requests.

Meant to be used in conjunction with EnterpriseCustomerCourseEnrollmentsSerializer.

create(*validated_data*)

This selectively calls the child create method based on whether or not validation failed for each payload.

to_internal_value(*data*)

This implements the same relevant logic as ListSerializer except that if one or more items fail validation, processing for other items that did not fail will continue.

to_representation(*data*)

This selectively calls to_representation on each result that was processed by create.

class enterprise.api.v1.serializers.**EnterpriseCustomerCourseEnrollmentsSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

Serializes enrollment information for a collection of students/emails.

This is mainly useful for implementing validation when performing enrollment operations.

class Meta

Bases: object

list_serializer_class

alias of *enterprise.api.v1.serializers.EnterpriseCustomerCourseEnrollmentsListSerializer*

create(*validated_data*)

Perform the enrollment for existing enterprise customer users, or create the pending objects for new users.

validate(*data*)

Validate that at least one of the user identifier fields has been passed in.

validate_course_run_id(*value*)

Validates that the course run id is part of the Enterprise Customer's catalog.

validate_lms_user_id(*value*)

Validates the lms_user_id, if is given, to see if there is an existing EnterpriseCustomerUser for it.

validate_tpa_user_id(*value*)

Validates the tpa_user_id, if is given, to see if there is an existing EnterpriseCustomerUser for it.

It first uses the third party auth api to find the associated username to do the lookup.

validate_user_email(*value*)

Validates the user_email, if given, to see if an existing EnterpriseCustomerUser exists for it.

If it does not, it does not fail validation, unlike for the other field validation methods above.

class enterprise.api.v1.serializers.**EnterpriseCustomerIdentityProviderSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for EnterpriseCustomerIdentityProvider model.

class Meta

Bases: object

fields = ('provider_id', 'default_provider')

model

alias of *enterprise.models.EnterpriseCustomerIdentityProvider*

class enterprise.api.v1.serializers.**EnterpriseCustomerInviteKeyPartialUpdateSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for updating the EnterpriseCustomerInviteKey model.

class Meta

Bases: object

fields = ('expiration_date', 'is_active')

model

alias of *enterprise.models.EnterpriseCustomerInviteKey*

class enterprise.api.v1.serializers.**EnterpriseCustomerInviteKeyReadOnlySerializer**(**args*, ***kwargs*)

Bases: *enterprise.api.v1.serializers.BaseEnterpriseCustomerInviteKeySerializer*

Serializer for reading the EnterpriseCustomerInviteKey model.

class Meta

Bases: *enterprise.api.v1.serializers.BaseEnterpriseCustomerInviteKeySerializer.Meta*

additional_fields = ('enterprise_customer_name', 'usage_count', 'created')

fields = ('uuid', 'enterprise_customer_uuid', 'usage_limit', 'expiration_date', 'is_active', 'is_valid', 'enterprise_customer_name', 'usage_count', 'created')

get_enterprise_customer_name(*obj*)

get_enterprise_customer_uuid(*obj*)

class enterprise.api.v1.serializers.**EnterpriseCustomerInviteKeyWriteSerializer**(**args*, ***kwargs*)

Bases: *enterprise.api.v1.serializers.BaseEnterpriseCustomerInviteKeySerializer*

Serializer for writing to the EnterpriseCustomerInviteKey model.

save()

validate_enterprise_customer_uuid(*value*)

Validates an EnterpriseCustomer with the given enterprise_customer_uuid exists.

class enterprise.api.v1.serializers.**EnterpriseCustomerReportingConfigurationSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for EnterpriseCustomerReportingConfiguration model.

class Meta

Bases: object

fields = ('enterprise_customer', 'enterprise_customer_id', 'active', 'delivery_method', 'email', 'frequency', 'day_of_month', 'day_of_week', 'hour_of_day', 'include_date', 'encrypted_password', 'sftp_hostname', 'sftp_port', 'sftp_username', 'encrypted_sftp_password', 'sftp_file_path', 'data_type', 'report_type', 'pgp_encryption_key', 'enterprise_customer_catalogs', 'uuid', 'enterprise_customer_catalog_uuids', 'enable_compression')

model

alias of *enterprise.models.EnterpriseCustomerReportingConfiguration*

create(*validated_data*)

Perform the creation of model instance and link the enterprise customer catalogs.

Parameters **validated_data** (*dict*) – A dictionary containing serializer's validated data.

Returns

Instance of the newly created enterprise customer reporting configuration.

Return type (*EnterpriseCustomerReportingConfiguration*)

update(*instance*, *validated_data*)

Update the instance of enterprise customer reporting configuration and link the enterprise customer catalogs.

Parameters

- **instance** (EnterpriseCustomerReportingConfiguration) – Instance of the enterprise customer reporting configuration being updated.
- **validated_data** (*dict*) – A dictionary containing serializer's validated data.

Returns

Instance of the newly created enterprise customer reporting configuration.

Return type (*EnterpriseCustomerReportingConfiguration*)

validate(*data*)

class enterprise.api.v1.serializers.**EnterpriseCustomerSerializer**(**args*, ***kwargs*)

Bases: `rest_framework.serializers.ModelSerializer`

Serializer for EnterpriseCustomer model.

class Meta

Bases: `object`

fields = ('uuid', 'name', 'slug', 'active', 'site', 'enable_data_sharing_consent', 'enforce_data_sharing_consent', 'branding_configuration', 'identity_provider', 'enable_audit_enrollment', 'replace_sensitive_sso_username', 'enable_portal_code_management_screen', 'sync_learner_profile_data', 'enable_audit_data_reporting', 'enable_learner_portal', 'enable_learner_portal_offers', 'enable_portal_learner_credit_management_screen', 'enable_executive_education_2U_fulfillment', 'enable_portal_reporting_config_screen', 'enable_portal_saml_configuration_screen', 'contact_email', 'enable_portal_subscription_management_screen', 'hide_course_original_price', 'enable_analytics_screen', 'enable_integrated_customer_learner_portal_search', 'enable_portal_lms_configurations_screen', 'sender_alias', 'identity_providers', 'enterprise_customer_catalogs', 'reply_to', 'enterprise_notification_banner', 'hide_labor_market_data', 'modified', 'enable_universal_link', 'enable_browse_and_request', 'admin_users')

model

alias of `enterprise.models.EnterpriseCustomer`

get_admin_users(*obj*)

get_branding_configuration(*obj*)

Return the serialized branding configuration object OR default object if null

get_enterprise_customer_catalogs(*obj*)

Return list of catalog uuids associated with the enterprise customer.

get_enterprise_notification_banner(*obj*)

Return the notification text if exist OR None

class enterprise.api.v1.serializers.**EnterpriseCustomerToggleUniversalLinkSerializer**(**args*, ***kwargs*)

Bases: `rest_framework.serializers.Serializer`

Serializer for toggling an EnterpriseCustomer enable_universal_link field.

class enterprise.api.v1.serializers.**EnterpriseCustomerUnlinkUsersSerializer**(**args*, ***kwargs*)

Bases: `rest_framework.serializers.Serializer`

Serializer for the `EnterpriseCustomerViewSet` unlink_users action.

class enterprise.api.v1.serializers.**EnterpriseCustomerUserReadOnlySerializer**(**args*, ***kwargs*)

Bases: `rest_framework.serializers.ModelSerializer`

Serializer for EnterpriseCustomerUser model.

class Meta

Bases: object

fields = ('id', 'enterprise_customer', 'active', 'user_id', 'user', 'data_sharing_consent_records', 'groups', 'created', 'invite_key', 'role_assignments')

model

alias of *enterprise.models.EnterpriseCustomerUser*

get_data_sharing_consent_records(*obj*)

Return serialization of EnterpriseCustomerUser.data_sharing_consent_records property.

Parameters **EnterpriseCustomerUser** – The EnterpriseCustomerUser.

Returns The serialized DataSharingConsent records associated with the EnterpriseCustomerUser.

Return type list of dict

get_groups(*obj*)

Return the enterprise related django groups that this user is a part of.

get_role_assignments(*obj*)

Return the enterprise role assignments for this enterprise customer user.

class enterprise.api.v1.serializers.**EnterpriseCustomerUserWriteSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for writing to the EnterpriseCustomerUser model.

class Meta

Bases: object

fields = ('enterprise_customer', 'username', 'active')

model

alias of *enterprise.models.EnterpriseCustomerUser*

USER_DOES_NOT_EXIST = 'User does not exist'

save()

Save the EnterpriseCustomerUser.

user = None

validate_username(*value*)

Verify that the username has a matching user.

class enterprise.api.v1.serializers.**ImmutableStateSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

Base serializer for any serializer that inhibits state changing requests.

create(*validated_data*)

Do not perform any operations for state changing requests.

update(*instance*, *validated_data*)

Do not perform any operations for state changing requests.

class enterprise.api.v1.serializers.**LicensedEnterpriseCourseEnrollmentReadOnlySerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for LicensedEnterpriseCourseEnrollment model.

class Meta

Bases: object

fields = ('enterprise_course_enrollment', 'license_uuid')

model

alias of *enterprise.models.LicensedEnterpriseCourseEnrollment*

class enterprise.api.v1.serializers.**LicensesInfoSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

Nested serializer class to allow for many license info dictionaries.

create(*validated_data*)

validate(*data*)

class enterprise.api.v1.serializers.**LinkLearnersSerializer**(**args*, ***kwargs*)

Bases: *enterprise.api.v1.serializers.PendingEnterpriseCustomerUserSerializer*

Extends the PendingEnterpriseCustomerSerializer to validate that the enterprise customer uuid matches the uuid the user has permissions to update

NOT_AUTHORIZED_ERROR = 'Not authorized for this enterprise'

validate_enterprise_customer(*value*)

Check that the enterprise customer is the same as the one the user has permissions for The value recieved is an EnterpriseCustomer object

class enterprise.api.v1.serializers.**PendingEnterpriseCustomerUserSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for writing to the PendingEnterpriseCustomerUser model.

class Meta

Bases: object

fields = ('enterprise_customer', 'user_email')

model

alias of *enterprise.models.PendingEnterpriseCustomerUser*

create(*attrs*)

Create the PendingEnterpriseCustomerUser, or EnterpriseCustomerUser if a user with the validated_email already exists.

to_representation(*instance*)

Because we are returning whether or not the instance was created from the create method, we must use the instance for to_representation and ignore the "created" half of the tuple

class enterprise.api.v1.serializers.**ProgramDetailSerializer**(**args*, ***kwargs*)

Bases: *enterprise.api.v1.serializers.ImmutableStateSerializer*

Serializer for program data retrieved from the discovery service program detail API endpoint.

This serializer updates the program data and child course run data with EnterpriseCustomer-specific enrollment page URLs for the given content types.

to_representation(*instance*)

Return the updated program data dictionary.

Parameters **instance** (*dict*) – The program data.

Returns The updated program data.

Return type dict

class enterprise.api.v1.serializers.**ResponsePaginationSerializer**(**args*, ***kwargs*)

Bases: *enterprise.api.v1.serializers.ImmutableStateSerializer*

Serializer for responses that require pagination.

class enterprise.api.v1.serializers.**SiteSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for Site model.

class Meta

Bases: object

fields = ('domain', 'name')

model

alias of django.contrib.sites.models.Site

class enterprise.api.v1.serializers.**UserSerializer**(**args*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for User model.

class Meta

Bases: object

fields = ('id', 'username', 'first_name', 'last_name', 'email', 'is_staff', 'is_active', 'date_joined')

model

alias of django.contrib.auth.models.User

enterprise.api.v1.urls module

URL definitions for enterprise api version 1 endpoint.

enterprise.api.v1.views module

Views for enterprise api version 1 endpoint.

class enterprise.api.v1.views.**CatalogQueryView**(***kwargs*)

Bases: rest_framework.views.APIView

View for enterprise catalog query. This will be called from django admin tool to populate *content_filter* field of *EnterpriseCustomerCatalog* model.

authentication_classes = [<class 'rest_framework.authentication.SessionAuthentication'>]

get(*request*, *catalog_query_id*)

API endpoint for fetching an enterprise catalog query.

http_method_names = ['get']

permission_classes = [<class 'rest_framework.permissions.IsAuthenticated'>, <class 'rest_framework.permissions.IsAdminUser'>]

class enterprise.api.v1.views.**CouponCodesView**(***kwargs*)

Bases: rest_framework.views.APIView

API to request coupon codes.

MISSING_REQUIRED_PARAMS_MSG = 'Some required parameter(s) missing: {}'

OPTIONAL_PARAM_NOTES = 'notes'

OPTIONAL_PARAM_NUMBER_OF_CODES = 'number_of_codes'

REQUIRED_PARAM_EMAIL = 'email'

REQUIRED_PARAM_ENTERPRISE_NAME = 'enterprise_name'

authentication_classes = (<class 'edx_rest_framework_extensions.auth.jwt.authentication.JwtAuthentication'>, <class 'rest_framework.authentication.SessionAuthentication'>)

get_missing_params_message(*parameter_state*)

Get a user-friendly message indicating a missing parameter for the API endpoint.

get_required_query_params(*request*)

Gets email, enterprise_name, number_of_codes, and notes, which are the relevant parameters for this API endpoint.

Parameters **request** – The request to this endpoint.

Returns The email, enterprise_name, number_of_codes and notes from the request.

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

post(*request*)

POST /enterprise/api/v1/request_codes

Requires a JSON object of the following format:

```
{
    "email": "bob@alice.com",
    "enterprise_name": "IBM",
    "number_of_codes": "50",
    "notes": "Help notes for codes request",
}
```

Keys

- **email** – Email of the customer who has requested more codes.
- **enterprise_name** – The name of the enterprise requesting more codes.
- **number_of_codes** – The number of codes requested.
- **notes** – Help notes related to codes request.

throttle_classes = (<class 'enterprise.api.throttles.ServiceUserThrottle'>,)

exception enterprise.api.v1.views.**EnrollmentModificationException**

Bases: Exception

An exception that represents an error when modifying the state of an enrollment via the EnrollmentApiClient.

class enterprise.api.v1.views.**EnterpriseCourseEnrollmentViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API views for the enterprise-course-enrollment API endpoint.

FIELDS = ('enterprise_customer_user', 'course_id')

USER_ID_FILTER = 'enterprise_customer_user__user_id'

basename = None

description = None

detail = None

filterset_fields = ('enterprise_customer_user', 'course_id')

get_serializer_class()

Use a special serializer for any requests that aren't read-only.

name = None

ordering_fields = ('enterprise_customer_user', 'course_id')

queryset

suffix = None

class enterprise.api.v1.views.**EnterpriseCustomerBrandingConfigurationViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API views for the `enterprise-customer-branding` API endpoint.

FIELDS = ('enterprise_customer__slug',)

USER_ID_FILTER = 'enterprise_customer__enterprise_customer_users__user_id'

basename = None

description = None

detail = None

filterset_fields = ('enterprise_customer__slug',)

lookup_field = 'enterprise_customer__slug'

name = None

ordering_fields = ('enterprise_customer__slug',)

parser_classes = [<class 'rest_framework.parsers.MultiPartParser'>, <class 'rest_framework.parsers.FormParser'>]

queryset

serializer_class

alias of *enterprise.api.v1.serializers.EnterpriseCustomerBrandingConfigurationSerializer*

suffix = None

update_branding(*request*, *enterprise_uuid*)

PATCH /enterprise/api/v1/enterprise-customer-branding/update_branding/uuid

Requires enterprise customer uuid path parameter

class enterprise.api.v1.views.**EnterpriseCustomerCatalogViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadOnlyModelViewSet*

API Views for performing search through course discovery at the `enterprise_catalogs` API endpoint.

FIELDS = ('uuid', 'enterprise_customer')

USER_ID_FILTER = 'enterprise_customer__enterprise_customer_users__user_id'

basename = None

contains_content_items(*request*, *pk*, *course_run_ids*, *program_uuids*)

Return whether or not the EnterpriseCustomerCatalog contains the specified content.

Multiple course_run_ids and/or program_uuids query parameters can be sent to this view to check for their existence in the EnterpriseCustomerCatalog. At least one course run key or program UUID value must be included in the request.

course_detail(*request*, *pk*, *course_key*)

Return the metadata for the specified course.

The course needs to be included in the specified EnterpriseCustomerCatalog in order for metadata to be returned from this endpoint.

course_run_detail(*request*, *pk*, *course_id*)

Return the metadata for the specified course run.

The course run needs to be included in the specified EnterpriseCustomerCatalog in order for metadata to be returned from this endpoint.

description = None

detail = None

filterset_fields = ('uuid', 'enterprise_customer')

get_serializer_class()

Return the class to use for the serializer. Defaults to using *self.serializer_class*.

You may want to override this if you need to provide different serializations depending on the incoming request.

(Eg. admins get full serialization, others get basic serialization)

list(*request*, **args*, ***kwargs*)

name = None

ordering_fields = ('uuid', 'enterprise_customer')

program_detail(*request*, *pk*, *program_uuid*)

Return the metadata for the specified program.

The program needs to be included in the specified EnterpriseCustomerCatalog in order for metadata to be returned from this endpoint.

queryset

renderer_classes = (<class 'rest_framework.renderers.JSONRenderer'>, <class 'rest_framework_xml.renderers.XMLRenderer'>)

retrieve(*request*, **args*, ***kwargs*)

suffix = None

class enterprise.api.v1.views.**EnterpriseCustomerInviteKeyViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API for accessing enterprise customer keys.

authentication_classes = (<class 'edx_rest_framework_extensions.auth.jwt.authentication.JwtAuthentication'>, <class 'rest_framework.authentication.SessionAuthentication'>)

basename = None

basic_list(*request*, **args*, ***kwargs*)
Unpaginated list of all invite keys matching the filters.

create(*request*, **args*, ***kwargs*)

description = None

destroy(*request*, **args*, ***kwargs*)

detail = None

filter_backends = (<class 'rest_framework.filters.OrderingFilter'>, <class 'django_filters.rest_framework.backends.DjangoFilterBackend'>, <class 'enterprise.api.filters.EnterpriseCustomerInviteKeyFilterBackend'>)

get_serializer_class()
Use a special serializer for any requests that aren't read-only.

http_method_names = ['get', 'post', 'patch']

link_user(*request*, *pk=None*)
Post Links user using enterprise_customer_key /enterprise/api/enterprise-customer-invite-key/{enterprise_customer_key}/link-user

Given a enterprise_customer_key, link user to the appropriate enterprise.

If the key is not found, returns 404 If the key is not valid, returns 422 If we create an *EnterpriseCustomerUser* returns 201 If an *EnterpriseCustomerUser* if found returns 200

list(*request*, **args*, ***kwargs*)

name = None

partial_update(*request*, **args*, ***kwargs*)

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

queryset

retrieve(*request*, **args*, ***kwargs*)

suffix = None

class enterprise.api.v1.views.**EnterpriseCustomerReportTypesView**(***kwargs*)
Bases: `rest_framework.views.APIView`

API for getting the report types associated with an enterprise customer

authentication_classes = [<class 'edx_rest_framework_extensions.auth.jwt.authentication.JwtAuthentication'>, <class 'rest_framework.authentication.SessionAuthentication'>]

get(*request*, *enterprise_uuid*)
Get the dropdown choices for EnterpriseCustomerReportingConfiguration

http_method_names = ['get']

permission_classes = [<class 'rest_framework.permissions.IsAuthenticated'>]

class enterprise.api.v1.views.**EnterpriseCustomerReportingConfigurationViewSet**(***kwargs*)
Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API views for the `enterprise-customer-reporting` API endpoint.

FIELDS = ('enterprise_customer',)

USER_ID_FILTER = 'enterprise_customer__enterprise_customer_users__user_id'

basename = None

create(*request*, **args*, ***kwargs*)

description = None

destroy(*request*, **args*, ***kwargs*)

detail = None

filterset_fields = ('enterprise_customer',)

list(*request*, **args*, ***kwargs*)

lookup_field = 'uuid'

name = None

ordering_fields = ('enterprise_customer',)

partial_update(*request*, **args*, ***kwargs*)

permission_classes = [<class 'rest_framework.permissions.IsAuthenticated'>]

queryset

retrieve(*request*, **args*, ***kwargs*)

serializer_class

alias of *enterprise.api.v1.serializers.EnterpriseCustomerReportingConfigurationSerializer*

suffix = None

update(*request*, **args*, ***kwargs*)

class enterprise.api.v1.views.**EnterpriseCustomerUserViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API views for the `enterprise-learner` API endpoint.

FIELDS = ('enterprise_customer', 'user_id', 'active')

basename = None

description = None

detail = None

filter_backends = (<class 'rest_framework.filters.OrderingFilter'>, <class 'django_filters.rest_framework.backends.DjangoFilterBackend'>, <class 'enterprise.api.filters.EnterpriseCustomerUserFilterBackend'>)

filterset_fields = ('enterprise_customer', 'user_id', 'active')

get_serializer_class()

Use a flat serializer for any requests that aren't read-only.

name = None

ordering_fields = ('enterprise_customer', 'user_id', 'active')

queryset

suffix = None

class enterprise.api.v1.views.**EnterpriseCustomerViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API views for the `enterprise-customer` API endpoint.

FIELDS = ('uuid', 'slug', 'name', 'active', 'site', 'enable_data_sharing_consent', 'enforce_data_sharing_consent')

USER_ID_FILTER = 'enterprise_customer_users__user_id'

basename = None

basic_list(*request*, **arg*, ***kwargs*)
Enterprise Customer's Basic data list without pagination

contains_content_items(*request*, *pk*, *course_run_ids*, *program_uuids*)
Return whether or not the specified content is available to the EnterpriseCustomer.

Multiple course_run_ids and/or program_uuids query parameters can be sent to this view to check for their existence in the EnterpriseCustomerCatalogs associated with this EnterpriseCustomer. At least one course run key or program UUID value must be included in the request.

course_enrollments(*request*, *pk*)
Creates a course enrollment for an EnterpriseCustomerUser.

dashboard_list(*request*, **args*, ***kwargs*)
Supports listing dashboard enterprises for frontend-app-admin-portal.

description = None

detail = None

enroll_learners_in_courses(*request*, *pk*)
Creates a set of licensed enterprise_learners by bulk enrolling them in all specified courses. This endpoint is not transactional, in that any one or more failures will not affect other successful enrollments made within the same request.

Parameters

- **licenses_info** (*list of dicts*) – an array of dictionaries, each containing the necessary information to create a licenced enrollment for a user in a specified course. Each dictionary must contain a user email, a course run key, and a UUID of the license that the learner is using to enroll with.

  Example:

```
licenses_info: [
    {
        'email': 'newuser@test.com',
        'course_run_key': 'course-v1:edX+DemoX+Demo_Course',
        'license_uuid': '5b77bdbade7b4fcb838f8111b68e18ae',
    },
    ...
]
```

- **discount** (*int*) – the percent discount to be applied to all enrollments. Defaults to 100.

Returns

- **All users exist and are enrolled -** {'successes': [], 'pending': [], 'failures': []}, 201
- **Some or none of the users exist but are enrolled -** {'successes': [], 'pending': [], 'failures': []}, 202

Failure cases:

- **Some or all of the users can't be enrolled, no users were enrolled -** {'successes': [], 'pending': [], 'failures': []}, 409

- **Some or all of the provided emails are invalid** {'successes': [], 'pending': [], 'failures': [] 'invalid_email_addresses': []}, 409

Return type Success cases

filter_backends = (<class 'rest_framework.filters.OrderingFilter'>, <class 'django_filters.rest_framework.backends.DjangoFilterBackend'>, <class 'enterprise.api.filters.UserFilterBackend'>, <class 'enterprise.api.filters.EnterpriseLinkedUserFilterBackend'>)

filterset_fields = ('uuid', 'slug', 'name', 'active', 'site', 'enable_data_sharing_consent', 'enforce_data_sharing_consent')

get_permissions()
Instantiates and returns the list of permissions that this view requires.

get_serializer_class()
Return the class to use for the serializer. Defaults to using *self.serializer_class*.

You may want to override this if you need to provide different serializations depending on the incoming request.

(Eg. admins get full serialization, others get basic serialization)

name = None

ordering_fields = ('uuid', 'slug', 'name', 'active', 'site', 'enable_data_sharing_consent', 'enforce_data_sharing_consent')

partial_update(*request*, **args*, ***kwargs*)

queryset

serializer_class
alias of *enterprise.api.v1.serializers.EnterpriseCustomerSerializer*

suffix = None

toggle_universal_link(*request*, *pk=None*)
Enables/Disables universal link config.

unlink_users(*request*, *pk=None*)
Unlinks users with the given emails from the enterprise.

with_access_to(*request*, **args*, ***kwargs*)
Returns the list of enterprise customers the user has a specified group permission access to.

class enterprise.api.v1.views.**EnterpriseModelViewSet**
Bases: *enterprise.api.v1.views.EnterpriseViewSet*

Base class for attribute and method definitions common to all view sets.

USER_ID_FILTER = 'id'

filter_backends = (<class 'rest_framework.filters.OrderingFilter'>, <class 'django_filters.rest_framework.backends.DjangoFilterBackend'>, <class 'enterprise.api.filters.UserFilterBackend'>)

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>, <class 'rest_framework.permissions.DjangoModelPermissions'>)

class enterprise.api.v1.views.**EnterpriseReadOnlyModelViewSet**(***kwargs*)
Bases: *enterprise.api.v1.views.EnterpriseModelViewSet*, rest_framework.viewsets.ReadOnlyModelViewSet

Base class for all read only Enterprise model view sets.

class enterprise.api.v1.views.**EnterpriseReadWriteModelViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseModelViewSet*, rest_framework.viewsets.ModelViewSet

Base class for all read/write Enterprise model view sets.

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>, <class 'rest_framework.permissions.DjangoModelPermissions'>)

class enterprise.api.v1.views.**EnterpriseViewSet**

Bases: object

Base class for all Enterprise view sets.

authentication_classes = (<class 'edx_rest_framework_extensions.auth.jwt.authentication.JwtAuthentication'>, <class 'rest_framework.authentication.SessionAuthentication'>)

ensure_data_exists(*request*, *data*, *error_message=None*)

Ensure that the wrapped API client's response brings us valid data. If not, raise an error and log it.

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

throttle_classes = (<class 'enterprise.api.throttles.ServiceUserThrottle'>,)

class enterprise.api.v1.views.**EnterpriseWrapperApiViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseViewSet*, rest_framework.viewsets.ViewSet

Base class for attribute and method definitions common to all view sets which wrap external APIs.

class enterprise.api.v1.views.**LicensedEnterpriseCourseEnrollmentViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseWrapperApiViewSet*

API views for the `licensed-enterprise-course-enrollment` API endpoint.

class EnrollmentTerminationStatus

Bases: object

Defines statuses related to enrollment states during the course unenrollment process.

COURSE_COMPLETED = 'course already completed'

MOVED_TO_AUDIT = 'moved to audit'

UNENROLLED = 'unenrolled'

UNENROLL_FAILED = 'unenroll_user_from_course returned false.'

OPT_IGNORE_ENROLLMENTS_MODIFIED_AFTER_PARAM = 'ignore_enrollments_modified_after'

REQ_EXP_LICENSE_UUIDS_PARAM = 'expired_license_uuids'

basename = None

bulk_licensed_enrollments_expiration(*request*)

Changes the mode for licensed enterprise course enrollments to the "audit" course mode, or unenroll the user if no audit mode exists for each expired license uuid

Parameters

- **expired_license_uuids** – The expired license uuids.
- **ignore_enrollments_modified_after** – All course enrollments modified past this given date will be ignored, i.e. the enterprise subscription plan expiration date.

description = None

detail = None

license_revoke(*request*, **args*, ***kwargs*)

Changes the mode for a user's licensed enterprise course enrollments to the "audit" course mode, or unenroll the user if no audit mode exists for a given course.

Will return a response with status 200 if no errors were encountered while modifying the course enrollment, or a 422 if any errors were encountered. The content of the response is of the form:

```
{
    'course-v1:puppies': {'success': true, 'message': 'unenrolled'},
    'course-v1:birds': {'success': true, 'message': 'moved to audit'},
    'course-v1:kittens': {'success': true, 'message': 'course already completed
↪'},
    'course-v1:snakes': {'success': false, 'message': 'unenroll_user_from_
↪course returned false'},
    'course-v1:lizards': {'success': false, 'message': 'Some other exception'},
}
```

The first four messages are the values of constants that a client may expect to receive and parse accordingly.

name = None

queryset

serializer_class

alias of *enterprise.api.v1.serializers.LicensedEnterpriseCourseEnrollmentReadOnlySerializer*

suffix = None

class enterprise.api.v1.views.**NotificationReadView**(***kwargs*)

Bases: rest_framework.views.APIView

API to mark notifications as read.

MISSING_REQUIRED_PARAMS_MSG = 'Some required parameter(s) missing: {}'

REQUIRED_PARAM_ENTERPRISE_SLUG = 'enterprise_slug'

REQUIRED_PARAM_NOTIFICATION_ID = 'notification_id'

authentication_classes = (<class 'edx_rest_framework_extensions.auth.jwt.authentication.JwtAuthentication'>, <class 'rest_framework.authentication.SessionAuthentication'>)

get_missing_params_message(*parameter_state*)

Get a user-friendly message indicating a missing parameter for the API endpoint.

get_required_query_params(*request*)

Gets notification_id and enterprise_slug. which are the relevant parameters for this API endpoint.

Parameters **request** – The request to this endpoint.

Returns The notification_id and enterprise_slug from the request.

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

post(*request*)

POST /enterprise/api/v1/read_notification

Requires a JSON object of the following format:

```
{
    'notification_id': 1,
    'enterprise_slug': 'enterprise_slug',
}
```

Keys

- **notification_id** – Notification ID which is read by Current User.
- **enterprise_slug** – The slug of the enterprise.

throttle_classes = (<class 'enterprise.api.throttles.ServiceUserThrottle'>,)

class enterprise.api.v1.views.**PendingEnterpriseCustomerUserEnterpriseAdminViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.PendingEnterpriseCustomerUserViewSet*

Viewset for allowing enterpise admins to create linked learners Endpoint url: link_pending_enterprise_users/(?P<enterprise_uuid>[A-Za-z0-9-]+)/?$ Admin must be an administrator for the enterprise in question

basename = None

description = None

detail = None

link_learners(*request*, *enterprise_uuid*)

Creates a PendingEnterpriseCustomerUser if no EnterpriseCustomerUser for the given (customer, email) combination(s) exists. Can accept one user or a list of users.

Returns 201 if any users were created, 204 if no users were created.

name = None

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

serializer_class

alias of *enterprise.api.v1.serializers.LinkLearnersSerializer*

suffix = None

class enterprise.api.v1.views.**PendingEnterpriseCustomerUserViewSet**(***kwargs*)

Bases: *enterprise.api.v1.views.EnterpriseReadWriteModelViewSet*

API views for the `pending-enterprise-learner` API endpoint. Requires staff permissions

FIELDS = ('enterprise_customer', 'user_email')

UNIQUE = 'unique'

USER_EXISTS_ERROR = 'EnterpriseCustomerUser record already exists'

basename = None

create(*request*, **args*, ***kwargs*)

Creates a PendingEnterpriseCustomerUser if no EnterpriseCustomerUser for the given (customer, email) combination(s) exists. Can accept one user or a list of users.

Returns 201 if any users were created, 204 if no users were created.

description = None

detail = None

filter_backends = (<class 'rest_framework.filters.OrderingFilter'>, <class 'django_filters.rest_framework.backends.DjangoFilterBackend'>)

filterset_fields = ('enterprise_customer', 'user_email')

name = None

ordering_fields = ('enterprise_customer', 'user_email')

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>, <class 'rest_framework.permissions.IsAdminUser'>)

queryset

serializer_class
alias of *enterprise.api.v1.serializers.PendingEnterpriseCustomerUserSerializer*

suffix = None

class enterprise.api.v1.views.**TableauAuthView**(***kwargs*)
Bases: rest_framework.generics.GenericAPIView

API to authenticate user with Tableau.

get(*request*, *enterprise_uuid*)
Get the auth token against logged in user from tableau

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

Module contents

API endpoint for enterprise app.

Submodules

enterprise.api.filters module

Filters for enterprise API.

class enterprise.api.filters.**EnterpriseCustomerInviteKeyFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend

Filter backend to return invite keys under the user's enterprise(s) only. Supports filtering by enterprise_customer_uuid.

- Staff users will bypass this filter.

filter_queryset(*request*, *queryset*, *view*)
Return a filtered queryset.

class enterprise.api.filters.**EnterpriseCustomerUserFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend

Allow filtering on the enterprise customer user api endpoint.

filter_queryset(*request*, *queryset*, *view*)
Apply incoming filters only if user is staff. If not, only filter by user's ID.

class enterprise.api.filters.**EnterpriseLinkedUserFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend

Filter backend to return user's linked enterprises only

- Staff users will bypass this filter.
- Non-staff users will receive only their linked enterprises.

filter_queryset(*request*, *queryset*, *view*)
Filter out enterprise customer if learner is not linked

class enterprise.api.filters.**UserFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend

Filter backend for any view that needs to filter against the requesting user's ID.

- Staff users will bypass this filter.
- Non-staff users will receive only those objects that match their own user ID.

This requires that *USER_ID_FILTER* be set in the view as a class variable, to identify the object's relationship to a user ID.

filter_queryset(*request*, *queryset*, *view*)
Filter only for the user's ID if non-staff.

enterprise.api.pagination module

Pagination helpers for enterprise api.

enterprise.api.pagination.**get_paginated_response**(*data*, *request*)
Update pagination links in course catalog data and return DRF Response.

Parameters

- **data** (*dict*) – Dictionary containing catalog courses.
- **request** (*HttpRequest*) – Current request object.

Returns DRF response object containing pagination links.

Return type (Response)

enterprise.api.throttles module

Throttle classes for enterprise API.

class enterprise.api.throttles.**ServiceUserThrottle**
Bases: rest_framework.throttling.UserRateThrottle

A throttle allowing the service user to override rate limiting.

allow_request(*request*, *view*)
Modify throttling for service users.

Updates throttling rate if the request is coming from the service user, and defaults to UserRateThrottle's configured setting otherwise.

Updated throttling rate comes from *DEFAULT_THROTTLE_RATES* key in *REST_FRAMEWORK* setting. service user throttling is specified in *DEFAULT_THROTTLE_RATES* by *service_user* key

Example Setting:

```
REST_FRAMEWORK = {
    ...
    'DEFAULT_THROTTLE_RATES': {
        ...
        'service_user': '50/day'
    }
}
```

update_throttle_scope()
Update throttle scope so that service user throttle rates are applied.

enterprise.api.urls module

URL definitions for enterprise API endpoint.

enterprise.api.utils module

Utility functions for the Enterprise API.

enterprise.api.utils.**create_message_body**(*email*, *enterprise_name*, *number_of_codes=None*, *notes=None*)
Return the message body with extra information added by user.

enterprise.api.utils.**get_ent_cust_from_enterprise_customer_key**(*enterprise_customer_key*)
Get the enterprise customer id given an enterprise customer key.

enterprise.api.utils.**get_ent_cust_from_report_config_uuid**(*uuid*)
Get the enterprise customer id given an enterprise report configuration UUID.

enterprise.api.utils.**get_enterprise_customer_from_catalog_id**(*catalog_id*)
Get the enterprise customer id given an enterprise customer catalog id.

enterprise.api.utils.**get_enterprise_customer_from_user_id**(*user_id*)
Get the enterprise customer id given an user id

enterprise.api.utils.**get_service_usernames**()
Return the set of service usernames that are given extended permissions in the API.

Module contents

Python API for various enterprise functionality.

enterprise.api.**activate_admin_permissions**(*enterprise_customer_user*)
Activates admin permissions for an existing PendingEnterpriseCustomerAdminUser.

Specifically, the "enterprise_admin" system-wide role is assigned to the user and the PendingEnterpriseCustomerAdminUser record is removed.

Requires an EnterpriseCustomerUser record to exist which ensures the user already has the "enterprise_learner" role as a prerequisite.

Parameters **enterprise_customer_user** – an EnterpriseCustomerUser instance

enterprise.api_client package

Submodules

enterprise.api_client.client module

Base API clients to comunicate with edx services.

class enterprise.api_client.client.**APIClientMixin**

Bases: object

API client mixin.

API_BASE_URL = ''

APPEND_SLASH = False

get_api_url(*path*)

Helper method to construct the API URL.

Parameters **path** (*str*) – the API endpoint path string.

class enterprise.api_client.client.**BackendServiceAPIClient**

Bases: *enterprise.api_client.client.APIClientMixin*

API client based on OAuthAPIClient to cummunicate with edx services.

Uses the backend service user to make requests.

class enterprise.api_client.client.**NoAuthAPIClient**

Bases: *enterprise.api_client.client.APIClientMixin*

API client based on requests.Session to cummunicate with edx services.

Used to call APIs which don't require authentication.

get_health()

Retrieve health details for the service.

Returns Response containing the service health.

Return type dict

class enterprise.api_client.client.**UserAPIClient**(*user*, *expires_in=3600*)

Bases: *enterprise.api_client.client.APIClientMixin*

API client based on requests.Session to cummunicate with edx services.

Requires user object to instantiate the client with the jwt token authentication.

connect()

Connect to the REST API, authenticating with a JWT for the current user.

static refresh_token(*func*)

Use this method decorator to ensure the JWT token is refreshed when needed.

token_is_expired()

Return True if the JWT token has expired, False if not.

enterprise.api_client.discovery module

Utilities to get details from the course catalog API.

class enterprise.api_client.discovery.**CourseCatalogApiClient**(*user*, *site=None*)

Bases: *enterprise.api_client.client.UserAPIClient*

The API client to make calls to the Catalog API.

APPEND_SLASH = True

CATALOGS_COURSES_ENDPOINT = 'catalogs/{}/courses/'

COURSES_ENDPOINT = 'courses'

COURSE_RUNS_ENDPOINT = 'course_runs'

DEFAULT_VALUE_SAFEGUARD = <object object>

PROGRAMS_ENDPOINT = 'programs'

PROGRAM_TYPES_ENDPOINT = 'program_types'

SEARCH_ALL_ENDPOINT = 'search/all/'

get_catalog_results(*content_filter_query*, *query_params=None*, *traverse_pagination=False*)

Return results from the cache or discovery service's search/all endpoint.

Parameters

- **content_filter_query** (*dict*) – query parameters used to filter catalog results.
- **query_params** (*dict*) – query parameters used to paginate results.
- **traverse_pagination** (*bool*) – True to return all results, False to return the paginated response. Defaults to False.

Returns Paginated response or all the records.

Return type dict

get_catalog_results_from_discovery(*content_filter_query*, *query_params=None*, *traverse_pagination=False*)

Return results from the discovery service's search/all endpoint.

get_course_and_course_run(*course_run_id*)

Return the course and course run metadata for the given course run ID.

Parameters **course_run_id** (*str*) – The course run ID.

Returns The course metadata and the course run metadata.

Return type tuple

get_course_details(*course_id*)

Return the details of a single course by id - not a course run id.

Parameters **course_id** (*str*) – The unique id for the course in question.

Returns Details of the course in question.

Return type dict

get_course_id(*course_identifier*)

Return the course id for the given course identifier. The *course_identifier* may be a course id or a course run id; in either case the course id will be returned.

The 'course id' is the identifier for a course (ex. edX+DemoX) The 'course run id' is the identifier for a run of a course (ex. edX+DemoX+demo_run)

Parameters **course_identifier** (*str*) – The course id or course run id

Returns course id

Return type (str)

get_course_run(*course_run_id*)

Return course_run data, including name, ID and seats.

Parameters **course_run_id** (*string*) – Course run ID (aka Course Key) in string format.

Returns Course run data provided by Course Catalog API.

Return type dict

get_course_run_identifiers(*course_run_id*)

Return all course and course run keys and uuids for the specified course run id

get_program_by_uuid(*program_uuid*)

Return single program by UUID, or None if not found.

Parameters **program_uuid** (*string*) – Program UUID in string form

Returns Program data provided by Course Catalog API

Return type dict

get_program_course_keys(*program_uuid*)

Get a list of the course IDs (not course run IDs) contained in the program.

Parameters **program_uuid** (*str*) – Program UUID in string form

Returns List of course keys in string form that are included in the program

Return type list(str)

get_program_type_by_slug(*slug*)

Get a program type by its slug.

Parameters **slug** (*str*) – The slug to identify the program type.

Returns A program type object.

Return type dict

traverse_pagination(*response*, *api_url*, *content_filter_query*, *query_params*)

Traverse a paginated API response and extracts and concatenates "results" returned by API.

Parameters

- **response** (*dict*) – API response object.
- **api_url** (*str*) – API endpoint path.
- **content_filter_query** (*dict*) – query parameters used to filter catalog results.
- **query_params** (*dict*) – query parameters used to paginate results.

Returns all the results returned by the API.

Return type list

class enterprise.api_client.discovery.**CourseCatalogApiServiceClient**(*site=None*)

Bases: *enterprise.api_client.discovery.CourseCatalogApiClient*

Catalog API client which uses the configured Catalog service user.

program_exists(*program_uuid*)
Get whether the program exists or not.

class enterprise.api_client.discovery.**NoAuthDiscoveryClient**
Bases: *enterprise.api_client.client.NoAuthAPIClient*

Class to build a course discovery client to make calls to the discovery service.

API_BASE_URL = 'http://localhost:18381/'

APPEND_SLASH = False

enterprise.api_client.discovery.**get_course_catalog_api_service_client**(*site=None*)
Returns an instance of the CourseCatalogApiServiceClient

Parameters **site** – (Site)

Returns (CourseCatalogServiceClient)

enterprise.api_client.ecommerce module

Client for communicating with the E-Commerce API.

class enterprise.api_client.ecommerce.**EcommerceApiClient**(*user*)
Bases: object

The API client to make calls to the E-Commerce API.

create_manual_enrollment_orders(*enrollments*)
Calls ecommerce to create orders for the manual enrollments passed in.

enrollments should be a list of enrollments with the following format:

```
{
    "lms_user_id": <int>,
    "username": <str>,
    "email": <str>,
    "course_run_key": <str>
}
```

Since *student.CourseEnrollment* lives in LMS, we're just passing around dicts of the relevant information.

get_course_final_price(*mode*, *currency='$'*, *enterprise_catalog_uuid=None*)
Get course mode's SKU discounted price after applying any entitlement available for this user.

Returns Discounted price of the course mode.

Return type str

class enterprise.api_client.ecommerce.**NoAuthEcommerceClient**
Bases: *enterprise.api_client.client.NoAuthAPIClient*

The E-Commerce API client without authentication.

API_BASE_URL = 'http://localhost:18130'

APPEND_SLASH = False

enterprise.api_client.enterprise module

Client for communicating with the Enterprise API.

class enterprise.api_client.enterprise.**EnterpriseApiClient**(*user*, *expires_in=3600*)

Bases: *enterprise.api_client.client.UserAPIClient*

The API client to make calls to the Enterprise API.

API_BASE_URL = 'http://localhost:8000/enterprise/api/v1/'

APPEND_SLASH = True

DEFAULT_VALUE_SAFEGUARD = <object object>

ENTERPRISE_CUSTOMER_CATALOGS_ENDPOINT = 'enterprise_catalogs'

ENTERPRISE_CUSTOMER_ENDPOINT = 'enterprise-customer'

get_content_metadata(*enterprise_customer*, *enterprise_catalogs=None*)

Return all content metadata contained in the catalogs associated with the EnterpriseCustomer.

Parameters

- **enterprise_customer** (EnterpriseCustomer) – The EnterpriseCustomer to return content metadata for.
- **enterprise_catalogs** (EnterpriseCustomerCatalog) – Optional list of EnterpriseCustomerCatalog objects.

Returns List of dicts containing content metadata.

Return type list

enterprise.api_client.enterprise_catalog module

Client for communicating with the Enterprise API.

class enterprise.api_client.enterprise_catalog.**EnterpriseCatalogApiClient**(*user=None*)

Bases: *enterprise.api_client.client.UserAPIClient*

The API client to make calls to the Enterprise Catalog API.

API_BASE_URL = 'http://localhost:18160/api/v1/'

APPEND_SLASH = True

CATALOG_DIFF_ENDPOINT = 'enterprise-catalogs/{}/generate_diff'

ENTERPRISE_CATALOG_ENDPOINT = 'enterprise-catalogs'

ENTERPRISE_CUSTOMER_ENDPOINT = 'enterprise-customer'

GET_CONTENT_METADATA_ENDPOINT = 'enterprise-catalogs/{}/get_content_metadata'

GET_CONTENT_METADATA_PAGE_SIZE = 50

REFRESH_CATALOG_ENDPOINT = 'enterprise-catalogs/{}/refresh_metadata'

contains_content_items(*catalog_uuid*, *content_ids*)

Checks whether an enterprise catalog contains the given content.

The enterprise catalog endpoint does not differentiate between course_run_ids and program_uuids so they can be used interchangeably. The two query parameters are left in for backwards compatability with edx-enterprise.

create_enterprise_catalog(*catalog_uuid*, *enterprise_id*, *enterprise_name*, *title*, *content_filter*, *enabled_course_modes*, *publish_audit_enrollment_urls*, *catalog_query_uuid*, *query_title=None*)

Creates an enterprise catalog.

delete_enterprise_catalog(*catalog_uuid*)

Deletes an enterprise catalog.

enterprise_contains_content_items(*enterprise_uuid*, *content_ids*)

Checks whether an enterprise customer has any catalogs that contain the provided content ids.

The endpoint does not differentiate between course_run_ids and program_uuids so they can be used interchangeably. The two query parameters are left in for backwards compatability with edx-enterprise.

get_catalog_diff(*enterprise_customer_catalog*, *content_keys*, *should_raise_exception=True*)

Gets the representational difference between a list of course keys and the current state of content under an enterprise catalog. This difference is returned as three buckets of data: *items_not_found*, *items_not_included* and *items_found*.

Parameters

- **enterprise_customer_catalog** (EnterpriseCustomerCatalog) – The catalog object whose content is being diffed.
- **content_keys** (*list*) – List of string content keys
- **should_raise_exception** (*Bool*) – Optional param for whether or not api response exceptions should be raised.

Returns dictionaries of content_keys to create items_to_delete (list): dictionaries of content_keys to delete items_found (list): dictionaries of content_keys and date_updated datetimes of content to update

Return type items_to_create (list)

get_content_metadata(*enterprise_customer*, *enterprise_catalogs=None*, *content_keys_filter=None*)

Return all content metadata contained in the catalogs associated with the EnterpriseCustomer.

Parameters

- **enterprise_customer** (EnterpriseCustomer) – The EnterpriseCustomer to return content metadata for.
- **enterprise_catalogs** (EnterpriseCustomerCatalog) – Optional list of EnterpriseCustomerCatalog objects.
- **content_keys_filter** (*List*) – List of content keys to filter by in the content metadata endpoint

Returns List of dicts containing content metadata.

Return type list

static get_content_metadata_url(*uuid*)

Get the url for the preview information for an enterprise catalog

get_enterprise_catalog(*catalog_uuid*, *should_raise_exception=True*)

Gets an enterprise catalog.

Parameters

- **catalog_uuid** (*uuid*) – The uuid of an EnterpriseCatalog instance
- **should_raise_exception** (*bool*) – Whether an exception should be logged if a catalog does not yet exist in enterprise-catalog. Defaults to True.

Returns a dictionary representing an enterprise catalog

Return type dict

refresh_catalogs(*enterprise_catalogs*)

Kicks off async tasks to refresh catalogs so recent changes will populate to production without needing to wait for the daily jobs to run

Parameters **enterprise_catalogs** (EnterpriseCustomerCatalog) – List of EnterpriseCustomerCatalog objects to refresh

Returns Dict of async task ids for each catalog id

Return type Dict

traverse_get_content_metadata(*api_url*, *query*, *catalog_uuid*)

Helper method to traverse over a paginated response from the enterprise-catalog service's *get_content_metadata* endpoint.

update_enterprise_catalog(*catalog_uuid*, ***kwargs*)

Updates an enterprise catalog.

class enterprise.api_client.enterprise_catalog.**NoAuthEnterpriseCatalogClient**

Bases: *enterprise.api_client.client.NoAuthAPIClient*

The enterprise catalog API client to make calls to the discovery service.

API_BASE_URL = 'http://localhost:18160'

APPEND_SLASH = False

enterprise.api_client.lms module

Utilities to get details from the course catalog API.

class enterprise.api_client.lms.**CertificatesApiClient**(*user*, *expires_in=3600*)

Bases: *enterprise.api_client.client.UserAPIClient*

The API client to make calls to the LMS Certificates API.

Note that this API client requires a JWT token, and so it keeps its token alive.

API_BASE_URL = 'http://localhost:8000/api/certificates/v0/'

APPEND_SLASH = True

get_course_certificate(*course_id*, *username*)

Retrieve the certificate for the given username for the given course_id.

Args: * `course_id` (str): The string value of the course's unique identifier * `username` (str): The username ID identifying the user for which to retrieve the certificate

Returns:

a dict containing:

- `username`: A string representation of an user's username passed in the request.
- `course_id`: A string representation of a Course ID.
- `certificate_type`: A string representation of the certificate type.
- `created_date`: Datetime the certificate was created (tz-aware).
- `status`: A string representation of the certificate status.

- `is_passing`: True if the certificate has a passing status, False if not.
- `download_url`: A string representation of the certificate url.
- `grade`: A string representation of a float for the user's course grade.

class enterprise.api_client.lms.**CourseApiClient**

Bases: *enterprise.api_client.client.NoAuthAPIClient*

The API client to make calls to the Course API.

API_BASE_URL = 'http://localhost:8000/api/courses/v1/'

APPEND_SLASH = True

get_course_details(*course_id*)

Retrieve all available details about a course.

Parameters **course_id** (*str*) – The course ID identifying the course for which to retrieve details.

Returns Contains keys identifying those course details available from the courses API (e.g., name).

Return type dict

class enterprise.api_client.lms.**EmbargoApiClient**

Bases: `object`

Client interface for using the edx-platform embargo API.

static redirect_if_blocked(*request*, *course_run_ids*, *user=None*)

Return redirect to embargo error page if the given user is blocked.

class enterprise.api_client.lms.**EnrollmentApiClient**

Bases: *enterprise.api_client.client.BackendServiceAPIClient*

The API client to make calls to the Enrollment API.

API_BASE_URL = 'http://localhost:8000/api/enrollment/v1/'

enroll_user_in_course(*username*, *course_id*, *mode*, *cohort=None*, *enterprise_uuid=None*)

Call the enrollment API to enroll the user in the course specified by course_id.

Parameters

- **username** (*str*) – The username by which the user goes on the OpenEdX platform
- **course_id** (*str*) – The string value of the course's unique identifier
- **mode** (*str*) – The enrollment mode which should be used for the enrollment
- **cohort** (*str*) – Add the user to this named cohort
- **enterprise_uuid** (*str*) – Add course enterprise uuid

Returns A dictionary containing details of the enrollment, including course details, mode, username, etc.

Return type dict

get_course_details(*course_id*)

Query the Enrollment API for the course details of the given course_id.

Parameters **course_id** (*str*) – The string value of the course's unique identifier

Returns A dictionary containing details about the course, in an enrollment context (allowed modes, etc.)

Return type dict

`get_course_enrollment`(*username*, *course_id*)

Query the enrollment API to get information about a single course enrollment.

Parameters

- **username** (*str*) – The username by which the user goes on the OpenEdX platform
- **course_id** (*str*) – The string value of the course's unique identifier

Returns A dictionary containing details of the enrollment, including course details, mode, username, etc.

Return type dict

`get_course_modes`(*course_id*)

Query the Enrollment API for the specific course modes that are available for the given course_id.

Parameters **course_id** (*str*) – The string value of the course's unique identifier

Returns A list of course mode dictionaries.

Return type list

`get_enrolled_courses`(*username*)

Query the enrollment API to get a list of the courses a user is enrolled in.

Parameters **username** (*str*) – The username by which the user goes on the OpenEdX platform

Returns A list of course objects, along with relevant user-specific enrollment details.

Return type list

`has_course_mode`(*course_run_id*, *mode*)

Query the Enrollment API to see whether a course run has a given course mode available.

Parameters **course_run_id** (*str*) – The string value of the course run's unique identifier

Returns Whether the course run has the given mode avaialble for enrollment.

Return type bool

`is_enrolled`(*username*, *course_run_id*)

Query the enrollment API and determine if a learner is enrolled in a course run.

Parameters

- **username** (*str*) – The username by which the user goes on the OpenEdX platform
- **course_run_id** (*str*) – The string value of the course's unique identifier

Returns Indicating whether the user is enrolled in the course run. Returns False under any errors.

Return type bool

`unenroll_user_from_course`(*username*, *course_id*)

Call the enrollment API to unenroll the user in the course specified by course_id. :param username: The username by which the user goes on the OpenEdx platform :type username: str :param course_id: The string value of the course's unique identifier :type course_id: str

Returns Whether the unenrollment succeeded

Return type bool

update_course_enrollment_mode_for_user(*username*, *course_id*, *mode*)
Call the enrollment API to update a user's course enrollment to the specified mode, e.g. "audit".

Parameters

- **username** (*str*) – The username by which the user goes on the OpenEdx platform
- **course_id** (*str*) – The string value of the course's unique identifier
- **mode** (*str*) – The string value of the course mode, e.g. "audit"

Returns A dictionary containing details of the enrollment, including course details, mode, username, etc.

Return type dict

class enterprise.api_client.lms.**GradesApiClient**(*user*, *expires_in=3600*)
Bases: *enterprise.api_client.client.UserAPIClient*

The API client to make calls to the LMS Grades API.

Note that this API client requires a JWT token, and so it keeps its token alive.

API_BASE_URL = 'http://localhost:8000/api/grades/v1/'

APPEND_SLASH = True

get_course_assessment_grades(*course_id*, *username*)
Retrieve the assessment grades for the given username for the given course_id.

Args: * `course_id` (str): The string value of the course's unique identifier * `username` (str): The username ID identifying the user for which to retrieve the grade.

Raises:

HTTPError if no grade found for the given user+course.

Returns:

a list of dicts containing:

- `attempted`: A boolean representing whether the learner has attempted the subsection yet.
- `subsection_name`: String representation of the subsection's name.
- `category`: String representation of the subsection's category.
- `label`: String representation of the subsection's label.
- `score_possible`: The total amount of points that the learner could have earned on the subsection.
- `score_earned`: The total amount of points that the learner earned on the subsection.
- `percent`: A float representing the overall grade for the course.
- `module_id`: The ID of the subsection.

get_course_grade(*course_id*, *username*)
Retrieve the grade for the given username for the given course_id.

Args: * `course_id` (str): The string value of the course's unique identifier * `username` (str): The username ID identifying the user for which to retrieve the grade.

Raises:

HTTPError if no grade found for the given user+course.

Returns:

a dict containing:

- `username`: A string representation of a user's username passed in the request.
- `course_key`: A string representation of a Course ID.
- `passed`: Boolean representing whether the course has been passed according the course's grading policy.
- `percent`: A float representing the overall grade for the course
- `letter_grade`: A letter grade as defined in grading_policy (e.g. 'A' 'B' 'C' for 6.002x) or None

class `enterprise.api_client.lms.`**`NoAuthLMSClient`**

Bases: *`enterprise.api_client.client.NoAuthAPIClient`*

The LMS API client to make calls to the LMS without authentication.

`API_BASE_URL = 'http://localhost:8000'`

`APPEND_SLASH = False`

`get_health`()

Retrieve health details for LMS service.

Returns Response containing LMS service health.

Return type dict

class `enterprise.api_client.lms.`**`ThirdPartyAuthApiClient`**(*user*, *expires_in=3600*)

Bases: *`enterprise.api_client.client.UserAPIClient`*

The API client to make calls to the Third Party Auth API.

`API_BASE_URL = 'http://localhost:8000/api/third_party_auth/v0/'`

`get_remote_id`(*identity_provider*, *username*)

Retrieve the remote identifier for the given username.

Args: * `identity_provider` (str): identifier slug for the third-party authentication service used during SSO. * `username` (str): The username ID identifying the user for which to retrieve the remote name.

Returns the remote name of the given user. None if not found.

Return type string or None

`get_username_from_remote_id`(*identity_provider*, *remote_id*)

Retrieve the remote identifier for the given username.

Args: * `identity_provider` (str): identifier slug for the third-party authentication service used during SSO. * `remote_id` (str): The remote id identifying the user for which to retrieve the usernamename.

Returns the username of the given user. None if not found.

Return type string or None

Module contents

enterprise.config package

Submodules

enterprise.config.models module

Module for defining models needed for configuration of internal concerns like management command options and parameters.

class enterprise.config.models.**UpdateRoleAssignmentsWithCustomersConfig**(**args*, ***kwargs*)

Bases: `config_models.models.ConfigurationModel`

Model that specifies parameters for the `update_role_assignments_with_customers` management command.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

ROLE_CHOICES = [('enterprise_admin', 'enterprise_admin'), ('enterprise_learner', 'enterprise_learner'), ('enterprise_catalog_admin', 'enterprise_catalog_admin'), ('enterprise_openedx_operator', 'enterprise_openedx_operator')]

batch_size

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

changed_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

dry_run

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer_uuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_change_date(**, field=<django.db.models.fields.DateTimeField: change_date>, is_next=True, **kwargs*)

get_previous_by_change_date(**, field=<django.db.models.fields.DateTimeField: change_date>, is_next=False, **kwargs*)

get_role_display(**, field=<django.db.models.fields.CharField: role>*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

role
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

Module contents

enterprise.heartbeat package

Submodules

enterprise.heartbeat.checks module

Perform health checks om the services enterprise is dependant on.

`enterprise.heartbeat.checks.`**`check_discovery`**`()`
Check if course discovery service is up and running and accessible via API.

Raises (*DiscoveryNotAvailable*) – raised if LMS service is not accessible for some reason.

Returns A tuple containing service name and a message.

Return type (str, str)

`enterprise.heartbeat.checks.`**`check_ecommerce`**`()`
Check if E-Commerce service is up and running and accessible via API.

Raises (*EcommerceNotAvailable*) – raised if LMS service is not accessible for some reason.

Returns A tuple containing service name and a message.

Return type (str, str)

`enterprise.heartbeat.checks.`**`check_enterprise_catalog`**`()`
Check if enterprise catalog service is up and running and accessible via API.

Raises (*EnterpriseCatalogNotAvailable*) – raised if LMS service is not accessible for some reason.

Returns A tuple containing service name and a message.

Return type (str, str)

`enterprise.heartbeat.checks.`**`check_lms`**`()`
Check if LMS service is up and running and accessible via API.

Raises (*LMSNotAvailable*) – raised if LMS service is not accessible for some reason.

Returns A tuple containing service name and a message.

Return type (str, str)

enterprise.heartbeat.exceptions module

Exceptions for use by enterprise heartbeat module.

exception enterprise.heartbeat.exceptions.**DiscoveryNotAvailable**(*message*, **args*)
Bases: *enterprise.heartbeat.exceptions.ServiceNotAvailable*

Raised when Course Discovery service is not available.

service_name = 'Course Discovery'

exception enterprise.heartbeat.exceptions.**EcommerceNotAvailable**(*message*, **args*)
Bases: *enterprise.heartbeat.exceptions.ServiceNotAvailable*

Raised when E-Commerce service is not available.

service_name = 'E-Commerce'

exception enterprise.heartbeat.exceptions.**EnterpriseCatalogNotAvailable**(*message*, **args*)
Bases: *enterprise.heartbeat.exceptions.ServiceNotAvailable*

Raised when Enterprise Catalog service is not available.

service_name = 'Enterprise Catalog'

exception enterprise.heartbeat.exceptions.**LMSNotAvailable**(*message*, **args*)
Bases: *enterprise.heartbeat.exceptions.ServiceNotAvailable*

Raised when Learning Management System (LMS) service is not available.

service_name = 'Learning Management System (LMS)'

exception enterprise.heartbeat.exceptions.**ServiceNotAvailable**(*message*, *service_name*, **args*)
Bases: Exception

All heartbeat service status exception inherit from this excepton.

message = ''

service_name = ''

enterprise.heartbeat.throttles module

Throttles for heartbeat API endpoints.

class enterprise.heartbeat.throttles.**HeartbeatRateThrottle**
Bases: rest_framework.throttling.UserRateThrottle

Throttle class for streamlining heartbeat traffic to manage server load.

rate = '60/min'

scope = 'heartbeat'

enterprise.heartbeat.utils module

Run health checks on all the services enterprise service is dependant on.

class enterprise.heartbeat.utils.**Status**

Bases: object

Status constants for service health check.

OK = 'OK'

UNAVAILABLE = 'UNAVAILABLE'

enterprise.heartbeat.utils.**run_checks**()

Run health checks on all the services and return the status of each service.

Returns

first boolean un the tuple tells of all services are up and running or not, Dictionary containing the following key value pairs status: 'OK' if all the services are up

Return type (bool, dict)

enterprise.heartbeat.views module

Views for enterprise heartbeat check.

enterprise.heartbeat.views.**heartbeat**(*self*, *request*, **args*, ***kwargs*)

Simple view that an external service can use to check if the app is up.

Returns A json doc of service id: status or message. If the status for any service is anything other than True,it returns HTTP code 503 (Service Unavailable); otherwise, it returns 200.

Return type (JsonResponse)

Module contents

Enterprise heartbeat related code.

enterprise.management package

Subpackages

enterprise.management.commands package

Submodules

enterprise.management.commands.backfill_learner_role_assignments module

Management command for assigning enterprise_learner roles to existing linked enterprise users that are missing them.

class enterprise.management.commands.backfill_learner_role_assignments.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Management command for assigning enterprise_learner roles to existing enterprise users.

Example usage: $ ./manage.py backfill_learner_role_assignments

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

backfill_learner_role_assignments(*options*)
Assigns enterprise_learner role to users.

handle(**args*, ***options*)
Entry point for management command execution.

help = 'Assigns enterprise_learner role to linked enterprise users missing them.'

enterprise.management.commands.bulk_update_catalog_query_id module

Django management command for bulk updating EnterpriseCustomerCatalog record's EnterpriseCatalogQuery's.

class enterprise.management.commands.bulk_update_catalog_query_id.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Takes a new and old Enterprise Catalog Query ID, find all most recent records within the historical Enterprise Catalog table where enterprise catalog query ID is the old ID provided and updates the corresponding Enterprise Catalog table entries with the new ID.

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

get_args_from_database()
Returns an options dictionary from the current BulkCatalogQueryUpdateCommandConfiguration model.

handle(**args*, ***options*)
Entry point for management command execution.

help = "Updates specified EnterpriseCatalog's enterprise catalog query ID records with a provided new ID"

enterprise.management.commands.create_enterprise_course_enrollments module

Django management command for creating EnterpriseCourseEnrollment records.

class enterprise.management.commands.create_enterprise_course_enrollments.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Creates EnterpriseCourseEnrollment records (if they do not already exist) for CourseEnrollment records that are associated with an enterprise user and a course run that exists in the enterprise's catalog.

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

help = 'Create EnterpriseCourseEnrollment records for CourseEnrollment records associated with an enterprise.'

enterprise.management.commands.create_missing_dsc_records module

Django management command for creating DataSharingConsent records.

class enterprise.management.commands.create_missing_dsc_records.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django management command for creating DataSharingConsent records

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

enterprise.management.commands.email_drip_for_missing_dsc_records module

Django management command for sending an email to learners with missing DataSharingConsent records.

class enterprise.management.commands.email_drip_for_missing_dsc_records.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django management command for sending an email to learners with missing DataSharingConsent records

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

emit_event(*ec_user*, *course_id*, *enterprise_customer*, *greeting_name*)
Emit the Segment event which will be used by Braze to send the email

get_enterprise_course_enrollments(*options*)
Get EnterpriseCourseEnrollment records according to the options and with dsc enabled

handle(**args*, ***options*)
Management command for sending an email to learners with a missing DSC. Designed to run daily.

Example usage: $ ./manage.py email_drip_for_missing_dsc_records $ ./manage.py email_drip_for_missing_dsc_records –no-commit $ ./manage.py email_drip_for_missing_dsc_records –enrollment-before 2021-05-06 –no-commit $ ./manage.py email_drip_for_missing_dsc_records –enrollment-before 2021-05-06

enterprise.management.commands.ensure_singular_active_enterprise_customer_user module

Django management command to ensure there is at most a single EnterpriseCustomerUser with *active=True* for each enterprise user.

class `enterprise.management.commands.ensure_singular_active_enterprise_customer_user.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=Fa*
force_color=

Bases: `django.core.management.base.BaseCommand`

Django management command to ensure there is at most a single EnterpriseCustomerUser with *active=True* for each enterprise user.

Example usage: ./manage.py lms ensure_singular_active_enterprise_customer_user

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

enterprise.management.commands.fix_dsc_records module

Django management command to Fix DSC records having spaces in there course Id.

class `enterprise.management.commands.fix_dsc_records.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django management command to Fix DSC records having spaces in there course Id.

This Command fixes the DSC records what were saved due to bug in our system and DSC records were saved with spaces.

Example usage: ./manage.py lms fix_dsc_records ./manage.py lms fix_dsc_records –no-commit

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

enterprise.management.commands.migrate_enterprise_catalogs module

Django management command for migrating EnterpriseCustomerCatalog data to new service.

class `enterprise.management.commands.migrate_enterprise_catalogs.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Migrate EnterpriseCustomerCatalog data to new Enterprise Catalog service.

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

help = 'Migrate EnterpriseCustomerCatalog data to new Enterprise Catalog service.'

enterprise.management.commands.monthly_impact_report module

Django management command to send monthly impact report to enterprise admins.

class enterprise.management.commands.monthly_impact_report.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django management command to send monthly impact report to enterprise admins.

Example usage: ./manage.py lms monthly_impact_report ./manage.py lms monthly_impact_report –no-commit

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

emit_event(***kwargs*)
Emit the Segment event which will be used by Braze to send the email

get_query_results_from_snowflake()
Get query results from Snowflake and yield each row.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

enterprise.management.commands.nudge_dormant_enrolled_enterprise_learners module

Django management command to send nudge email to dormant enrolled enterprise learners.

class enterprise.management.commands.nudge_dormant_enrolled_enterprise_learners.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django management command to send nudge email to dormant enrolled enterprise learners.

Example usage: ./manage.py lms nudge_dormant_enrolled_enterprise_learners ./manage.py lms nudge_dormant_enrolled_enterprise_learners –no-commit

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

emit_event(***kwargs*)
Emit the Segment event which will be used by Braze to send the email

get_query_results_from_snowflake()
Get query results from Snowflake and yield each row.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

enterprise.management.commands.revert_enrollment_objects module

Management command for reverting revoked enrollment related objects to a particular time.

class `enterprise.management.commands.revert_enrollment_objects.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Management command for reverting revoked enrollment related objects to a particular time.

Example usage: $ ./manage.py revert_enrollment_objects –year 2021 –month 11 –day 17 –enterprise-customer-name test-co

`add_arguments`(*parser*)

Entry point for subclassed commands to add custom arguments.

`handle`(**args*, ***options*)

Entry point for management command execution.

`help = 'Reverts revoked enrollment related objects to a particular time.'`

`revert_enrollment_objects`(*options*)

Revert all EnterpriseCourseEnrollment, LicensedEnterpriseCourseEnrollment, and "student" CourseEnrollment objects to the date provided, using the history table IF is_revoked = True on LicensedEnterpriseCourseEnrollment

enterprise.management.commands.save_enterprise_customer_users module

Django management command for saving EnterpriseCustomerUser models.

class `enterprise.management.commands.save_enterprise_customer_users.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Calls save() on EnterpriseCustomerUser models.

This is useful for triggering save-related signals causing the associated signal receiver functions to fire.

`add_arguments`(*parser*)

Entry point for subclassed commands to add custom arguments.

`handle`(**args*, ***options*)

The actual logic of the command. Subclasses must implement this method.

`help = 'Save existing EnterpriseCustomerUser models.'`

enterprise.management.commands.seed_enterprise_devstack_data module

Management command for assigning enterprise roles to existing enterprise users.

class enterprise.management.commands.seed_enterprise_devstack_data.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Management command for populating Devstack with initial data for enterprise.

Example usage: $ ./manage.py lms seed_enterprise_devstack_data

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
Entry point for managment command execution.

help = '\n Seeds an enterprise customer, users of various roles and permissions initial\n data in devstack for related Enterprise models.\n '

enterprise.management.commands.unlink_enterprise_customer_learners module

Django management command to unlink the learners, Delete the enrollments and remove the DSC.

class enterprise.management.commands.unlink_enterprise_customer_learners.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django management command to unlink the learners, Delete the enrollments and remove the DSC

Example usage: ./manage.py lms unlink_enterprise_customer_learners -e <enterprise-uuid> –data-csv /path/file.csv ./manage.py lms unlink_enterprise_customer_learners -e <enterprise-uuid> –data-csv /path/file.csv –skip-unlink

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
The actual logic of the command. Subclasses must implement this method.

enterprise.management.commands.update_role_assignments_with_customers module

Management command for updating enterprise user role assignments with appropriate `enterprise_customer` and `applies_to_all_contexts` values.

class enterprise.management.commands.update_role_assignments_with_customers.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Management command for creating enterprise role assignments with a foreign key to an EnterpriseCustomer, or an explicit boolean flag indicating that the assignment grants the role to the user for every EnterpriseCustomer.

add_arguments(*parser*)
Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)
Entry point for managment command execution.

help = '\n Applies explicit enterprise customer context for enterprise admin, learner, and catalog system role assignments.\n It also sets `applies_to_all_contexts` to true for assignments of the `enterprise_openedx_operator` role.\n Example usage:\n # Do a dry run for admin assignments of some customer\n ./manage.py lms update_role_assignments_with_customers --role=enterprise_admin --dry-run --enterprise-customer-uuid=00000000-1111-2222-3333-444444444444\n # Do a real run for all operators\n ./manage.py lms update_role_assignments_with_customers --role=enterprise_openedx_operator\n # Process everything for everyone\n ./manage.py lms update_role_assignments_with_customers\n '

Module contents

Module contents

enterprise.settings package

Submodules

enterprise.settings.test module

These settings are here to use during tests, because django requires them.

In a real-world use case, apps in this project are installed into other Django applications, so these settings will not be used.

enterprise.settings.test.**here**(**args*)
Return the absolute path to a directory from this file.

enterprise.settings.test.**root**(**args*)
Return the absolute path to some file from the project's root.

Module contents

enterprise.templatetags package

Submodules

enterprise.templatetags.enterprise module

Template tags and filters for the Enterprise application.

enterprise.templatetags.enterprise.**alert_messages**(*messages*)
Django template tag that returns an alert message.

Usage: {% alert_messages messages %}

`enterprise.templatetags.enterprise.course_modal(context, course=None)`

Django template tag that returns course information to display in a modal.

You may pass in a particular course if you like. Otherwise, the modal will look for course context within the parent context.

Usage: {% course_modal %} {% course_modal course %}

`enterprise.templatetags.enterprise.expand_button(value, href)`

Django template tag that returns a button used to expand/collapse a container.

You may pass in the ID of the container that this button controls, and a button text value.

Usage: {% expand_button 'Click me!' '#id' %}

`enterprise.templatetags.enterprise.fa_icon(message_type)`

Django template tag that returns font awesome icon depending upon message type.

Usage: {% fa_icon "success" %}

`enterprise.templatetags.enterprise.link_to_modal(link_text, index, autoescape=True)`

Django template filter that returns an anchor with attributes useful for course modal selection.

General Usage: {{ link_text|link_to_modal:index }}

Examples

{{ course_title|link_to_modal:forloop.counter0 }} {{ course_title|link_to_modal:3 }} {{ view_details_text|link_to_modal:0 }}

`enterprise.templatetags.enterprise.only_safe_html(html_text)`

Django template filter that strips all HTML tags excepts those degined in ALLOWED_TAGS.

General Usage: {{ html_text|only_safe_html }}

Module contents

8.1.2 Submodules

8.1.3 enterprise.apps module

Enterprise Django application initialization.

class `enterprise.apps.EnterpriseConfig(app_name, app_module)`

Bases: `django.apps.config.AppConfig`

Configuration for the enterprise Django application.

property auth_user_model

Return User model for django.contrib.auth.

backend_service_edx_oauth2_key = 'test_backend_oauth2_key'

backend_service_edx_oauth2_provider_url = 'http://localhost:8000/oauth2'

backend_service_edx_oauth2_secret = 'test_backend_oauth2_secret'

customer_success_email = 'customersuccess@edx.org'

enterprise_integrations_email = 'enterprise-integrations@edx.org'

name = 'enterprise'

ready()
Perform other one-time initialization steps.

valid_image_extensions = ['.png']

valid_max_image_size = 512

8.1.4 enterprise.constants module

Enterprise Django application constants.

class enterprise.constants.**CourseModes**
Bases: object

Class to group modes that a course might have.

AUDIT = 'audit'

CREDIT = 'credit'

HONOR = 'honor'

NO_ID_PROFESSIONAL = 'no-id-professional'

PROFESSIONAL = 'professional'

VERIFIED = 'verified'

class enterprise.constants.**DefaultColors**
Bases: object

Class to group the default branding color codes. These color codes originated in the Enterprise Learner Portal.

PRIMARY = '#00262B'

SECONDARY = '#EFF8FA'

TERTIARY = '#0A7DA3'

enterprise.constants.**json_serialized_course_modes**()

Returns serialized course modes.

8.1.5 enterprise.decorators module

Decorators for enterprise app.

enterprise.decorators.**deprecated**(*extra*)
Flag a method as deprecated.

Parameters **extra** – Extra text you'd like to display after the default text.

enterprise.decorators.**disable_for_loaddata**(*signal_handler*)
Use this decorator to turn off signal handlers when loading fixture data.

Django docs instruct to avoid further changes to the DB if raw=True as it might not be in a consistent state. See https://docs.djangoproject.com/en/dev/ref/signals/#post-save

enterprise.decorators.**enterprise_login_required**(*view*)
View decorator for allowing authenticated user with valid enterprise UUID.

This decorator requires enterprise identifier as a parameter *enterprise_uuid*.

This decorator will throw 404 if no kwarg *enterprise_uuid* is provided to the decorated view .

If there is no enterprise in database against the kwarg *enterprise_uuid* or if the user is not authenticated then it will redirect the user to the enterprise-linked SSO login page.

Usage:

```
@enterprise_login_required()
def my_view(request, enterprise_uuid):
    # Some functionality ...

OR

class MyView(View):
    ...
    @method_decorator(enterprise_login_required)
    def get(self, request, enterprise_uuid):
        # Some functionality ...
```

`enterprise.decorators.`**`force_fresh_session`**(*view*)

View decorator which terminates stale TPA sessions.

This decorator forces the user to obtain a new session the first time they access the decorated view. This prevents TPA-authenticated users from hijacking the session of another user who may have been previously logged in using the same browser window.

This decorator should be used in conjunction with the enterprise_login_required decorator.

Usage:

```
@enterprise_login_required
@force_fresh_session()
def my_view(request, enterprise_uuid):
    # Some functionality ...

OR

class MyView(View):
    ...
    @method_decorator(enterprise_login_required)
    @method_decorator(force_fresh_session)
    def get(self, request, enterprise_uuid):
        # Some functionality ...
```

`enterprise.decorators.`**`ignore_warning`**(*warning*)

Ignore any emitted warnings from a function.

Parameters **warning** – The category of warning to ignore.

`enterprise.decorators.`**`null_decorator`**(*func*)

Use this decorator to stub out decorators for testing.

If we're unable to import social_core.pipeline.partial, which is the case in our CI platform, we need to be able to wrap the function with something.

8.1.6 enterprise.errors module

Errors thrown by the APIs in the Enterprise application.

exception enterprise.errors.**AdminNotificationAPIRequestError**

Bases: Exception

An exception that represents an error when creating admin notification read status via the NotificationReadApiClient.

exception enterprise.errors.**CodesAPIRequestError**

Bases: Exception

There was a problem with a request to the Codes application's APIs.

exception enterprise.errors.**EnrollmentModificationException**

Bases: Exception

An exception that represents an error when modifying the state of an enrollment via the EnrollmentApiClient.

exception enterprise.errors.**LinkUserToEnterpriseError**

Bases: Exception

An error occurred while linking a user to an enterprise.

exception enterprise.errors.**UnlinkUserFromEnterpriseError**

Bases: Exception

An error occurred while unlinking a user from an enterprise.

8.1.7 enterprise.forms module

User-facing forms for the Enterprise app.

class enterprise.forms.**EnterpriseLoginForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.forms.Form

Enterprise Slug Login Form.

base_fields = {'enterprise_slug': <django.forms.fields.CharField object>}

clean()

Validate POST data.

declared_fields = {'enterprise_slug': <django.forms.fields.CharField object>}

property media

Return all media required to render the widgets on this form.

class enterprise.forms.**EnterpriseSelectionForm**(**args*, ***kwargs*)

Bases: django.forms.forms.Form

Enterprise Selection Form.

base_fields = {'enterprise': <django.forms.fields.ChoiceField object>, 'success_url': <django.forms.fields.CharField object>}

clean()

Validate POST data.

declared_fields = {'enterprise': <django.forms.fields.ChoiceField object>, 'success_url': <django.forms.fields.CharField object>}

property media

Return all media required to render the widgets on this form.

8.1.8 enterprise.messages module

Utility functions for interfacing with the Django messages framework.

enterprise.messages.**add_consent_declined_message**(*request*, *enterprise_customer*, *item*)

Add a message to the Django messages store indicating that the user has declined data sharing consent.

Parameters

- **request** (*HttpRequest*) – The current request.
- **enterprise_customer** (EnterpriseCustomer) – The EnterpriseCustomer associated with this request.
- **item** (*str*) – A string containing information about the item for which consent was declined.

enterprise.messages.**add_generic_error_message_with_code**(*request*, *error_code*)

Add message to request indicating that there was an issue processing request.

Parameters

- **request** – The current request.
- **error_code** – A string error code to be used to point devs to the spot in the code where this error occurred.

enterprise.messages.**add_missing_price_information_message**(*request*, *item*)

Add a message to the Django messages store indicating that we failed to retrieve price information about an item.

Parameters

- **request** – The current request.
- **item** – The item for which price information is missing. Example: a program title, or a course.

enterprise.messages.**add_unenrollable_item_message**(*request*, *item*)

Add a message to the Django message store indicating that the item (i.e. course run, program) is unenrollable.

Parameters

- **request** – The current request.
- **item** – The item that is unenrollable (i.e. a course run).

8.1.9 enterprise.middleware module

Middleware for enterprise app.

class enterprise.middleware.**EnterpriseLanguagePreferenceMiddleware**(*get_response=None*)

Bases: `django.utils.deprecation.MiddlewareMixin`

Middleware for enterprise language preference.

Ensures that, once set, a user's preferences are reflected in the page whenever they are logged in.

process_request(*request*)

Perform the following checks

1. Check that the user is authenticated and belongs to an enterprise customer.
2. **Check that the enterprise customer has a language set via the *default_language* column on** EnterpriseCustomer model.
3. Check that user has not set a language via its account settings page.

If all the above checks are satisfied then set request._anonymous_user_cookie_lang to the *default_language* of EnterpriseCustomer model instance. This attribute will later be used by the *LanguagePreferenceMiddleware* middleware for setting the user preference. Since, this middleware relies on *LanguagePreferenceMiddleware* so it must always be followed by *LanguagePreferenceMiddleware*. Otherwise, it will not work.

8.1.10 enterprise.models module

Database models for enterprise.

class enterprise.models.**AdminNotification**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Model for Admin Notification.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

admin_notification_filter

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

adminnotificationread_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

expiration_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_expiration_date(*, *field=<django.db.models.fields.DateField: expiration_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_next_by_start_date(*, *field=<django.db.models.fields.DateField: start_date>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_expiration_date(*, *field=<django.db.models.fields.DateField: expiration_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_previous_by_start_date(*, *field=<django.db.models.fields.DateField: start_date>*, *is_next=False*, ***kwargs*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

start_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

text
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

title
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**AdminNotificationFilter**(**args*, ***kwargs*)
Bases: `model_utils.models.TimeStampedModel`

Model for Admin Notification Filters.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

filter
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

notification_filter
Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <django.db.models.manager.Manager object>

class enterprise.models.**AdminNotificationRead**(**args*, ***kwargs*)
Bases: `model_utils.models.TimeStampedModel`

Model for Admin Notification Read Status.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

admin_notification
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

admin_notification_id

enterprise_customer_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_user_id

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_read

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class enterprise.models.**BulkCatalogQueryUpdateCommandConfiguration**(**args*, ***kwargs*)

Bases: `config_models.models.ConfigurationModel`

Manages configuration for a run of the cert_generation management command.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

arguments

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

changed_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

get_next_by_change_date(*, *field=<django.db.models.fields.DateTimeField: change_date>*, *is_next=True*, ***kwargs*)

get_previous_by_change_date(*, *field=<django.db.models.fields.DateTimeField: change_date>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnrollmentNotificationEmailTemplate**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Store optional templates to use when emailing users about course enrollment events.

**BODY_HELP_TEXT = 'Fill in a standard Django template that, when rendered, produces the email you want sent to newly-enrolled Enterprise Customer learners. The following variables may be available:\n
user_name: A human-readable name for the person being emailed. Be sure to handle the case where this is not defined, as it may be missing in some cases. It may also be a username, if the learner hasn\'t configured their "real" name in the system. organization_name: The name of the organization sponsoring the enrollment. enrolled_in: Details of the course or program that was enrolled in. It may contain: name: The name of the enrollable item (e.g., "Demo Course"). url: A link to the homepage of the enrolled-in item. branding: A custom branding name for the enrolled-in item. For example, the branding of a MicroMasters program would be "MicroMasters". start: The date the enrolled-in item becomes available. Render this to text using the Django `date` template filter (see the Django documentation).
type: Whether the enrolled-in item is a course, a program, or something else.
'**

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

SUBJECT_HELP_TEXT = 'Enter a string that can be used to generate a dynamic subject line for notification emails. The placeholder {course_name} will be replaced with the name of the course or program that was enrolled in.'

enterprise_customer

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

enterprise_customer_id

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_template_type_display(*, *field=<django.db.models.fields.CharField: template_type>*)

history = <simple_history.manager.HistoryManager object>

html_template
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

plaintext_template
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

render_all_templates(*kwargs*)
Render both templates and return both.

render_html_template(*kwargs*)
Render just the HTML template and return it as a string.

render_plaintext_template(*kwargs*)
Render just the plaintext template and return it as a string.

render_template(*template_text*, *kwargs*)
Create a template from the DB-backed text and render it.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

subject_line
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

template_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

template_type_choices = [('SELF_ENROLL', 'Self Enrollment Template'), ('ADMIN_ENROLL', 'Admin Enrollment Template')]

class enterprise.models.**EnterpriseAnalyticsUser**(**args*, ***kwargs*)
Bases: model_utils.models.TimeStampedModel

Model for analytics users.

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

analytics_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_user_id

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class enterprise.models.**EnterpriseCatalogQuery**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Stores a re-usable catalog query.

This stored catalog query used in *EnterpriseCustomerCatalog* objects to build catalog's content_filter field. This is a saved instance of *content_filter* that can be re-used accross different catalogs.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

content_filter

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

delete(**args*, ***kwargs*)

Deletes this `EnterpriseCatalogQuery`.

enterprise_customer_catalogs

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

title

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnterpriseCourseEnrollment**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Store information about the enrollment of a user in a course.

This model is the central source of truth for information about whether a particular user, linked to a particular EnterpriseCustomer, has been enrolled in a course, and is the repository for any other relevant metadata about such an enrollment.

Do not delete records of this model - there are downstream business reporting processes that rely them, even if the underlying `student.CourseEnrollment` record has been marked inactive/un-enrolled. As a consequence, the only way to determine if a given `EnterpriseCourseEnrollment` is currently active is to examine the `is_active` field of the associated `student.CourseEnrollment`.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

property audit_reporting_disabled

Specify whether audit track data reporting is disabled for this enrollment.

- If the enterprise customer associated with this enrollment enables audit track data reporting, simply return False.
- If the enterprise customer associated with this enrollment does not enable audit track data reporting, return True if we are dealing with an audit enrollment, and False otherwise.

Returns True if audit track data reporting is disabled, False otherwise.

course_enrollment

Returns the `student.CourseEnrollment` associated with this enterprise course enrollment record.

course_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_user_id

classmethod get_enterprise_course_enrollment_id(*user*, *course_id*, *enterprise_customer*)

Return the EnterpriseCourseEnrollment object for a given user in given course_id.

classmethod get_enterprise_uuids_with_user_and_course(*user_id*, *course_run_id*, *is_customer_active=None*)

Returns a list of UUID(s) for EnterpriseCustomer(s) that this enrollment links together with the user_id and course_run_id

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property is_active

Returns True iff this enrollment is currently active.

property is_audit_enrollment

Specify whether the course enrollment associated with this `EnterpriseCourseEnrollment` is in audit mode.

Returns Whether the course enrollment mode is of an audit type.

property license

Returns the license associated with this enterprise course enrollment if one exists.

licensed_with

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

property mode

Returns the mode of the `student.CourseEnrollment` associated with this enterprise course enrollment record.

objects = <enterprise.models.EnterpriseCourseEnrollmentManager object>

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

saved_for_later

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

source

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

source_id

class enterprise.models.**EnterpriseCourseEnrollmentManager**(**args*, ***kwargs*)

Bases: `django.db.models.manager.Manager`

Model manager for *EnterpriseCourseEnrollment*.

get_queryset()

Override to return only those enrollment records for which learner is linked to an enterprise.

class enterprise.models.**EnterpriseCustomer**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Enterprise Customer is an organization or a group of people that "consumes" courses.

Users associated with an Enterprise Customer take courses on the edX platform.

Enterprise Customer might be providing certain benefits to their members, like discounts to paid course enrollments, and also might request (or require) sharing learner results with them.

Fields

- **uuid** (*UUIDField, PRIMARY KEY*) – Enterprise Customer code - used to reference this Enterprise Customer in other parts of the system (SSO, ecommerce, analytics etc.).
- **name** (`django.db.models.CharField`) – Enterprise Customer name.
- **active** (`django.db.models.BooleanField`) – used to mark inactive Enterprise Customers - implements "soft delete" pattern.

AT_ENROLLMENT = 'at_enrollment'

DATA_SHARING_CONSENT_CHOICES = (('at_enrollment', 'At Enrollment'), ('externally_managed', 'Managed externally'))

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

EXTERNALLY_MANAGED = 'externally_managed'

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

active_customers = <enterprise.models.EnterpriseCustomerManager object>

blackboardenterprisecustomerconfiguration_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

branding_configuration
Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

canvasenterprisecustomerconfiguration_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

catalog_contains_course(*course_run_id*)
Determine if the specified course run is contained in enterprise customer catalogs.

Parameters **course_run_id** (*str*) – The string ID of the course or course run in question

Returns Whether the enterprise catalog includes the given course run.

Return type bool

clear_pending_registration(*email*, **course_ids*)
Clear pending enrollments for the user in the given courses.

Parameters

- **email** – The email address which may have previously been used.
- **course_ids** – An iterable containing any number of course IDs.

contact_email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

contentmetadataitemtransmission_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

cornerstoneenterprisecustomerconfiguration_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

country

A descriptor for country fields on a model instance. Returns a Country when accessed so you can do things like:

```
>>> from people import Person
>>> person = Person.object.get(name='Chris')

>>> person.country.name
'New Zealand'

>>> person.country.flag
'/static/flags/nz.gif'
```

customer_type

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_type_id

data_sharing_consent_page

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

default_contract_discount

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

default_language

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property default_provider_idp

Return default_provider if associated with this enterprise customer.

degreed2enterprisecustomerconfiguration_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

degreedenterprisecustomerconfiguration_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

enable_analytics_screen

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_audit_data_reporting

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_audit_enrollment

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_autocohorting

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_browse_and_request
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_data_sharing_consent
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_executive_education_2U_fulfillment
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_integrated_customer_learner_portal_search
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_learner_portal
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_learner_portal_offers
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_code_management_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_learner_credit_management_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_lms_configurations_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_reporting_config_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_saml_configuration_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_subscription_management_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_slug_login
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_universal_link
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property enables_audit_data_reporting
Determine whether the enterprise customer has enabled the ability to report/pass-back audit track data.

enforce_data_sharing_consent
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enforces_data_sharing_consent(*enforcement_location*)
Determine whether the enterprise customer enforce data sharing consent at the given point.

Parameters

- **enforcement_location** (*str*) – the point where to see data sharing consent state.
- **'externally_managed'** (*argument can either be 'at_enrollment' or*) –

enroll_user_pending_registration(*email*, *course_mode*, **course_ids*, ***kwargs*)
Create pending enrollments for the user in any number of courses, which will take effect on registration.

Parameters

- **email** – The email address for the pending link to be created
- **course_mode** – The mode with which the eventual enrollment should be created
- ***course_ids** – An iterable containing any number of course IDs to eventually enroll the user in.
- **cohort** (*optional*) – name of cohort to assign

Returns The PendingEnterpriseCustomerUser attached to the email address

enroll_user_pending_registration_with_status(*email*, *course_mode*, **course_ids*, ***kwargs*)
Create pending enrollments for the user in any number of courses, which will take effect on registration. Return a dictionary representing status of submitted enrollments.

Parameters

- **email** – The email address for the pending link to be created
- **course_mode** – The mode with which the eventual enrollment should be created
- ***course_ids** – An iterable containing any number of course IDs to eventually enroll the user in.
- **cohort** (*optional*) – name of cohort to assign

Returns The PendingEnterpriseCustomerUser attached to the email address new_enrollments (Dict): course ID keys and new enrollment status values.

enterprise_customer_catalogs
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

enterprise_customer_consent
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

property enterprise_customer_identity_provider

Returns the first instance from EnterpriseCustomerIdentityProvider relation.

enterprise_customer_identity_providers

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

enterprise_customer_users

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

enterprise_enrollment_template

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

genericenterprisecustomerpluginconfiguration_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_country_display(*, *field=<django_countries.fields.CountryField: country>*)

get_course_enrollment_url(*course_key*)
Return enterprise landing page url for the given course.

Parameters **course_key** (*str*) – The course key for the course to be displayed.

Returns Enterprise landing page url.

Return type (str)

get_course_run_enrollment_url(*course_run_key*)
Return enterprise landing page url for the given course.

Parameters **course_run_key** (*str*) – The course run id for the course to be displayed.

Returns Enterprise landing page url.

Return type (str)

get_data_sharing_consent_text_overrides(*published_only=True*)
Return DataSharingConsentTextOverrides associated with this instance.

get_default_language_display(*, *field=<django.db.models.fields.CharField: default_language>*)

get_enforce_data_sharing_consent_display(*, *field=<django.db.models.fields.CharField: enforce_data_sharing_consent>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_program_enrollment_url(*program_uuid*)
Return enterprise landing page url for the given program.

Parameters **program_uuid** (*str*) – The program UUID.

Returns Enterprise program landing page url.

Return type (str)

get_tpa_hint(*tpa_hint_param=None*)

Parameters **tpa_hint_param** – query param passed in the URL.

Returns tpa_hint to redirect

property has_identity_providers
Return True if there are any identity providers associated with this enterprise customer.

property has_multiple_idps
Return True if there are multiple identity providers associated with this enterprise customer.

property has_single_idp
Return True if there are exactly one identity provider associated with this enterprise customer.

hide_course_original_price
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

hide_labor_market_data
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history = <simple_history.manager.HistoryManager object>

property identity_provider
Return the first identity provider id associated with this enterprise customer.

property identity_provider_ids
Return the identity provider Ids associated with this enterprise customer.

property identity_providers
Return the identity providers associated with this enterprise customer.

invite_keys
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

moodleenterprisecustomerconfiguration_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

notify_enrolled_learners(*catalog_api_user*, *course_id*, *users*, *admin_enrollment=False*, *activation_links=None*)
Notify learners about a course in which they've been enrolled.

Parameters

- **catalog_api_user** – The user for calling the Catalog API
- **course_id** – The specific course the learners were enrolled in
- **users** – An iterable of the users (or pending users) who were enrolled
- **admin_enrollment** – Default False. Set to true if using bulk enrollment, for example. When true, we use the admin enrollment template instead.
- **activation_links** (*dict*) – a dictionary map of unactivated license user emails to license activation links

objects = <django.db.models.manager.Manager object>

pendingenterprisecustomeradminuser_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

pendingenterprisecustomeruser_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

replace_sensitive_sso_username

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

reply_to

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

reporting_configurations

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

property requests_data_sharing_consent

Determine whether the enterprise customer has enabled the data sharing consent request.

property safe_branding_configuration

Return the associated EnterpriseCustomerBrandingConfiguration object OR default branding config

This function should always be used to access the customer's branding_configuration and prevent uncaught RelatedObjectDoesNotExist exceptions when accessed directly.

sapsuccessfactorsenterprisecustomerconfiguration_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

sender_alias
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property serialized
Return a serialized version of this customer.

site
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

site_id

slug
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property sync_learner_profile_data
Return the sync_learner_profile data flag for the identity provider associated with this enterprise customer.

Returns False if enterprise customer does not have any identity provider.

system_wide_role_assignments
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

toggle_universal_link(*enable_universal_link*, *link_expiration_date=None*)
Sets enable_universal_link

If there is no change to be made, return.

When enable_universal_link changes to:

- True: a new EnterpriseCustomerInviteKey is created

- False: all EnterpriseCustomerInviteKey are deactivated

Parameters

- **enable_universal_link** – new value
- **link_expiration_date** – if passed when enable_universal_link is true new link will be created

uuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

xapilrsconfiguration

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

class enterprise.models.**EnterpriseCustomerBrandingConfiguration**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Model that keeps track of enterprise branding configurations e.g. enterprise customer logo.

Fields

- **enterprise_customer** (*ForeignKey[EnterpriseCustomer]*) – enterprise customer
- **logo** (*ImageField*) – enterprise customer image

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

enterprise_customer

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

enterprise_customer_id

get_next_by_created(**, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(**, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(**, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(**, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

logo
Just like the FileDescriptor, but for ImageFields. The only difference is assigning the width/height to the width_field/height_field, if appropriate.

objects = <django.db.models.manager.Manager object>

primary_color
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property safe_logo_url
Returns an absolute URL for the branding configuration logo OR the platform logo absolute URL

secondary_color
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

tertiary_color
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnterpriseCustomerCatalog**(**args*, ***kwargs*)
Bases: model_utils.models.TimeStampedModel

Store catalog information from course discovery specifically for Enterprises.

We use this model to consolidate course catalog information, which includes information about catalogs, courses, programs, and possibly more in the future, as the course discovery service evolves.

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

contains_courses(*content_ids*)
Return True if this catalog contains the given courses else False.

The content_ids parameter should be a list containing course keys and/or course run ids.

contains_programs(*program_uuids*)
Return true if this catalog contains the given programs.

content_filter
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

content_filter_ids
Return the list of any content IDs specified in the catalog's content filter.

enabled_course_modes
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_catalog_query
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_catalog_query_id

enterprise_customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

enterprisecustomerreportingconfiguration_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_content_filter()

Return content filter of the linked catalog query otherwise content filter of catalog itself.

get_course(*course_key*)

Get all of the metadata for the given course.

Parameters **course_key** (*str*) – The course key which identifies the course.

Returns The course metadata.

Return type dict

get_course_and_course_run(*course_run_id*)

Get course data and all of the metadata for the given course run.

Parameters **course_run_id** (*str*) – The course run key which identifies the course run.

Returns The course and course run metadata.

Return type tuple(course, course_run)

Raises **ImproperlyConfigured** – Missing or invalid catalog integration.

get_course_enrollment_url(*course_key*)

Return enterprise course enrollment page url with the catalog information for the given course.

Parameters **course_key** (*str*) – The course key for the course to be displayed.

Returns Enterprise landing page url.

Return type (str)

get_course_run(*course_run_id*)
Get all of the metadata for the given course run.

Parameters **course_run_id** (*str*) – The course run key which identifies the course run.

Returns The course run metadata.

Return type dict

get_course_run_enrollment_url(*course_run_key*)
Return enterprise course enrollment page url with the catalog information for the given course.

Parameters **course_run_key** (*str*) – The course run id for the course to be displayed.

Returns Enterprise landing page url.

Return type (str)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_paginated_content(*query_parameters*)
Return paginated discovery service search results without expired course runs.

Parameters **query_parameters** (*dict*) – Additional query parameters to add to the search API call, e.g. page.

Returns The paginated discovery service search results.

Return type dict

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_program(*program_uuid*)
Get all of the metadata for the given program.

Parameters **program_uuid** (*str*) – The program UUID which identifies the program.

Returns The program metadata.

Return type dict

get_program_enrollment_url(*program_uuid*)
Return enterprise program enrollment page url with the catalog information for the given program.

Parameters **program_uuid** (*str*) – The program UUID.

Returns Enterprise program landing page url.

Return type (str)

history = <simple_history.manager.HistoryManager object>

objects = <django.db.models.manager.Manager object>

publish_audit_enrollment_urls
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

save(**args*, ***kwargs*)
Saves this `EnterpriseCatalogQuery`.

Copies the `content_filter` of a related CatalogQuery into this instance's `content_filter` if syncing is allowed.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

title
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnterpriseCustomerIdentityProvider**(**args*, ***kwargs*)
Bases: `model_utils.models.TimeStampedModel`

EnterpriseCustomerIdentityProvider is a One to Many relationship between Enterprise Customer and Identity Provider.

There should be a link between an enterprise customer and its Identity Provider. This relationship has following constraints:

1. An enterprise customer may or may not have an identity provider.
2. An enterprise customer can have more than one identity providers.
3. Enterprise customer site should match with identity provider's site. (i.e. same domain names)

Fields

- **enterprise_customer** (*ForeignKey[EnterpriseCustomer]*) – enterprise customer
- **provider_id** (`django.db.models.SlugField`) – The provider_id string of the identity provider.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

default_provider
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property identity_provider

Associated identity provider instance.

objects = <django.db.models.manager.Manager object>

provider_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property provider_name

Readable name for the identity provider.

property sync_learner_profile_data

Return bool indicating if data received from the identity provider shoudl be synced to the edX profile.

class enterprise.models.**EnterpriseCustomerInviteKey**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`, `model_utils.models.SoftDeletableModel`

Stores an invite key used to link a learner to an enterprise.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

enterprise_customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

expiration_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classmethod from_db(*db*, *field_names*, *values*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_expiration_date(*, *field=<django.db.models.fields.DateTimeField: expiration_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_expiration_date(*, *field=<django.db.models.fields.DateTimeField: expiration_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

is_active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property is_valid

Returns whether the key is still valid (non-expired and usage limit has not been reached).

linked_enterprise_customer_users

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

save(**args*, ***kwargs*)

Saves this `EnterpriseCustomerInviteKey`.

Prevents is_active from being updated once it's set to False.

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

property usage_count

usage_limit

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnterpriseCustomerManager**(**args*, ***kwargs*)

Bases: `django.db.models.manager.Manager`

Model manager for *`EnterpriseCustomer`* model.

Filters out inactive Enterprise Customers, otherwise works the same as default model manager.

get_queryset()

Return a new QuerySet object. Filters out inactive Enterprise Customers.

use_for_related_fields = False

class enterprise.models.**EnterpriseCustomerReportingConfiguration**(**args*, ***kwargs*)

Bases: model_utils.models.TimeStampedModel

The Enterprise's configuration for sending automated data reports securely via email to the Enterprise Admin.

ALLOWED_NON_COMPRESSION_DATA_TYPES = ('catalog',)

DATA_TYPE_CATALOG = 'catalog'

DATA_TYPE_CHOICES = (('progress_v3', 'progress_v3'), ('catalog', 'catalog'), ('engagement', 'engagement'), ('grade', 'grade'), ('completion', 'completion'), ('course_structure', 'course_structure'))

DATA_TYPE_COMPLETION = 'completion'

DATA_TYPE_COURSE_STRUCTURE = 'course_structure'

DATA_TYPE_ENGAGEMENT = 'engagement'

DATA_TYPE_GRADE = 'grade'

DATA_TYPE_PROGRESS_V3 = 'progress_v3'

DAYS_OF_WEEK = ((0, 'Monday'), (1, 'Tuesday'), (2, 'Wednesday'), (3, 'Thursday'), (4, 'Friday'), (5, 'Saturday'), (6, 'Sunday'))

DELIVERY_METHOD_CHOICES = (('email', 'email'), ('sftp', 'sftp'))

DELIVERY_METHOD_EMAIL = 'email'

DELIVERY_METHOD_SFTP = 'sftp'

exception DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

FREQUENCY_CHOICES = (('daily', 'daily'), ('monthly', 'monthly'), ('weekly', 'weekly'))

FREQUENCY_TYPE_DAILY = 'daily'

FREQUENCY_TYPE_MONTHLY = 'monthly'

FREQUENCY_TYPE_WEEKLY = 'weekly'

exception MultipleObjectsReturned

Bases: django.core.exceptions.MultipleObjectsReturned

PEARSON_ONLY_REPORTS = ('grade', 'completion', 'course_structure')

REPORT_TYPE_CHOICES = (('csv', 'csv'), ('json', 'json'))

REPORT_TYPE_CSV = 'csv'

REPORT_TYPE_JSON = 'json'

active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

clean()

Override of clean method to perform additional validation on frequency, day_of_month/day_of week and compression.

data_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

day_of_month
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

day_of_week
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

decrypted_password
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

decrypted_sftp_password
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

delivery_method
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

email
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_compression
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property encrypted_password
Return encrypted password as a string.

The data is encrypted in the DB at rest, but is unencrypted in the app when retrieved through the decrypted_password field. This method will encrypt the password again before sending.

property encrypted_sftp_password
Return encrypted SFTP password as a string.

The data is encrypted in the DB at rest, but is unencrypted in the app when retrieved through the decrypted_password field. This method will encrypt the password again before sending.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_catalogs
Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

enterprise_customer_id

frequency

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_data_type_display(*, *field=<django.db.models.fields.CharField: data_type>*)

get_day_of_week_display(*, *field=<django.db.models.fields.SmallIntegerField: day_of_week>*)

get_delivery_method_display(*, *field=<django.db.models.fields.CharField: delivery_method>*)

get_frequency_display(*, *field=<django.db.models.fields.CharField: frequency>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_report_type_display(*, *field=<django.db.models.fields.CharField: report_type>*)

hour_of_day

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

include_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

pgp_encryption_key

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

report_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

sftp_file_path

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

sftp_hostname

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

sftp_port

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

sftp_username
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnterpriseCustomerType**(**args*, ***kwargs*)
Bases: `model_utils.models.TimeStampedModel`

Enterprise Customer Types are used to differentiate Enterprise learners.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

enterprisecustomer_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_next_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=True, **kwargs*)

get_next_by_modified(**, field=<model_utils.fields.AutoLastModifiedField: modified>, is_next=True, **kwargs*)

get_previous_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=False, **kwargs*)

get_previous_by_modified(**, field=<model_utils.fields.AutoLastModifiedField: modified>, is_next=False, **kwargs*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class enterprise.models.**EnterpriseCustomerUser**(**args*, ***kwargs*)
Bases: `model_utils.models.TimeStampedModel`

Model that keeps track of user - enterprise customer affinity.

Fields

- **enterprise_customer** (ForeignKey[*EnterpriseCustomer*]) – enterprise customer
- **user_id** (`django.db.models.IntegerField`) – user identifier

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

adminnotificationread_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

all_objects = <enterprise.models.EnterpriseCustomerUserManager object>

create_order_for_enrollment(*course_run_id*, *discount_percentage*, *mode*, *sales_force_id*)
Create an order on the Ecommerce side for tracking the course enrollment of a enterprise customer user.

property data_sharing_consent_records
Return the DataSharingConsent records associated with this EnterpriseCustomerUser.

Returns The filtered DataSharingConsent QuerySet.

Return type QuerySet (DataSharingConsent)

enroll(*course_run_id*, *mode*, *cohort=None*, *source_slug=None*, *discount_percentage=0.0*, *sales_force_id=None*)
Enroll a user into a course track, and register an enterprise course enrollment.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

enterprise_enrollments
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

enterpriseanalyticsuser_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

classmethod get_active_enterprise_users(*user_id*, *enterprise_customer_uuids=None*)

Return a queryset of all active enterprise users to which the given user is related. Or, if `enterprise_customer_uuids` is non-null, only the enterprise users related to the list of given `enterprise_customer_uuids`.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_remote_id(*idp_id=None*)

Retrieve the SSO provider's identifier for this user from the LMS Third Party API. In absence of idp_id, returns id from default idp

> **Parameters** **idp_id** (*str*) *(optional)* – If provided, idp resolution skipped and specified idp used to locate remote id.

Returns None if: * the user doesn't exist, or * the associated EnterpriseCustomer has no identity_provider, or * the remote identity is not found.

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classmethod inactivate_other_customers(*user_id*, *enterprise_customer*)

Mark as inactive all the enterprise customers of given user except the given enterprise_customer.

invite_key

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

invite_key_id

is_relinkable
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

linked
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <enterprise.models.EnterpriseCustomerUserManager object>

save(**args*, ***kwargs*)
Override to handle creation of EnterpriseCustomerUser records.

This is needed because of soft deletion of EnterpriseCustomerUser records. This will handle all of get_or_create/update_or_create/create methods.

By default, when an EnterpriseCustomerUser record is created/updated with *active=True*, all other linked records for the user will be marked as *active=False*. To disable this side effect, update set *should_inactivate_other_customers=False* on an EnterpriseCustomerUser instance.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

should_inactivate_other_customers
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

unenroll(*course_run_id*)
Unenroll a user from a course track.

update_session(*request*)
Update the session of a request for this learner.

property user
Return User associated with this instance.

Return `django.contrib.auth.models.User` instance associated with this *EnterpriseCustomerUser* instance via email.

property user_email
Return linked user email.

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property username
Return linked user's username.

class enterprise.models.**EnterpriseCustomerUserManager**(**args*, ***kwargs*)
Bases: `django.db.models.manager.Manager`

Model manager for *EnterpriseCustomerUser* entity.

This class should contain methods that create, modify or query *EnterpriseCustomerUser* entities.

get(***kwargs*)
Overridden get method to return the first element in case of learner with multiple enterprises.

Raises EnterpriseCustomerUser.DoesNotExist if no records are found.

get_link_by_email(*user_email*, *enterprise_customer*)
Return link by email and enterprise_customer

get_queryset()
Return linked or unlinked learners based on how the manager is created.

link_user(*enterprise_customer*, *user_email*)
Link user email to Enterprise Customer.

If `django.contrib.auth.models.User` instance with specified email does not exist, *PendingEnterpriseCustomerUser* instance is created instead.

unlink_user(*enterprise_customer*, *user_email*, *is_relinkable=True*)
Unlink user email from Enterprise Customer.

If `django.contrib.auth.models.User` instance with specified email does not exist, *PendingEnterpriseCustomerUser* instance is deleted instead.

Raises EnterpriseCustomerUser.DoesNotExist if instance of `django.contrib.auth.models.User` with specified email exists and corresponding *EnterpriseCustomerUser* instance does not.

Raises PendingEnterpriseCustomerUser.DoesNotExist exception if instance of `django.contrib.auth.models.User` with specified email exists and corresponding *PendingEnterpriseCustomerUser* instance does not.

class enterprise.models.**EnterpriseEnrollmentSource**(**args*, ***kwargs*)
Bases: `model_utils.models.TimeStampedModel`

Define a Name and Source for all Enterprise Enrollment Sources.

API = 'enterprise_api'

CUSTOMER_ADMIN = 'customer_admin'

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

ENROLLMENT_TASK = 'enrollment_task'

ENROLLMENT_URL = 'enrollment_url'

MANAGEMENT_COMMAND = 'management_command'

MANUAL = 'manual'

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

OFFER_REDEMPTION = 'offer_redemption'

enterprisecourseenrollment_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_next_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=True, **kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

classmethod get_source(*source_slug*)

Retrieve the source based on the Slug provided.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

pendingenrollment_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

slug

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**EnterpriseFeatureRole**(**args*, ***kwargs*)

Bases: `edx_rbac.models.UserRole`

Enterprise-specific feature role definitions.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

enterprisefeatureuserroleassignment_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class enterprise.models.**EnterpriseFeatureUserRoleAssignment**(**args*, ***kwargs*)

Bases: `edx_rbac.models.UserRoleAssignment`

Model to map users to an EnterpriseFeatureRole.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

property enterprise_customer_uuids

Get the enterprise customer uuids linked to the user.

get_context()

Returns a non-empty list of enterprise customer uuid strings to which `self.user` is linked, or `None` if the user is not linked to any EnterpriseCustomer.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

role

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

role_class
alias of *enterprise.models.EnterpriseFeatureRole*

role_id

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

class enterprise.models.**EnterpriseRoleAssignmentContextMixin**
Bases: `object`

Mixin for RoleAssignment models related to enterprises.

DEPRECATED: Not removing since it's referenced in a migration (0001_squashed_0092_auto_20200312_1650).

class enterprise.models.**HistoricalEnrollmentNotificationEmailTemplate**(*id*, *created*, *modified*, *plaintext_template*, *html_template*, *subject_line*, *template_type*, *enterprise_customer*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)
Bases: `simple_history.models.HistoricalChanges`, `django.db.models.base.Model`

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

get_template_type_display(*, *field=<django.db.models.fields.CharField: template_type>*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

html_template

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.EnrollmentNotificationEmailTemplate*

modified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

plaintext_template
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

subject_line
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

template_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalEnterpriseAnalyticsUser**(*id*, *created*, *modified*, *analytics_user_id*, *enterprise_customer_user*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: `simple_history.models.HistoricalChanges`, `django.db.models.base.Model`

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

analytics_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_user_id

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.EnterpriseAnalyticsUser*

modified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class enterprise.models.**HistoricalEnterpriseCourseEnrollment**(*id*, *created*, *modified*, *course_id*, *saved_for_later*, *enterprise_customer_user*, *source*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

course_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

enterprise_customer_user_id

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.EnterpriseCourseEnrollment*

modified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record

Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record

Get the previous history record for the instance. *None* if first.

revert_url()

URL for this change in the default admin site.

saved_for_later

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

source

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

source_id

class enterprise.models.**HistoricalEnterpriseCustomer**(*created*, *modified*, *uuid*, *name*, *slug*, *active*, *country*, *hide_course_original_price*, *enable_data_sharing_consent*, *enforce_data_sharing_consent*, *enable_audit_enrollment*, *enable_audit_data_reporting*, *replace_sensitive_sso_username*, *enable_autocohorting*, *enable_portal_code_management_screen*, *enable_portal_reporting_config_screen*, *enable_portal_subscription_management_screen*, *enable_portal_saml_configuration_screen*, *enable_universal_link*, *enable_browse_and_request*, *enable_learner_portal*, *enable_learner_portal_offers*, *enable_portal_learner_credit_management_screen*, *hide_labor_market_data*, *enable_integrated_customer_learner_portal_search*, *enable_analytics_screen*, *enable_portal_lms_configurations_screen*, *enable_slug_login*, *enable_executive_education_2U_fulfillment*, *contact_email*, *default_contract_discount*, *default_language*, *sender_alias*, *reply_to*, *site*, *customer_type*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: `simple_history.models.HistoricalChanges`, `django.db.models.base.Model`

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

contact_email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

country

A descriptor for country fields on a model instance. Returns a Country when accessed so you can do things like:

```
>>> from people import Person
>>> person = Person.object.get(name='Chris')

>>> person.country.name
'New Zealand'

>>> person.country.flag
'/static/flags/nz.gif'
```

created

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer_type

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_type_id

default_contract_discount

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

default_language

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_analytics_screen

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_audit_data_reporting

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_audit_enrollment

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_autocohorting

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_browse_and_request
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_data_sharing_consent
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_executive_education_2U_fulfillment
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_integrated_customer_learner_portal_search
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_learner_portal
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_learner_portal_offers
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_code_management_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_learner_credit_management_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_lms_configurations_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_reporting_config_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_saml_configuration_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_portal_subscription_management_screen
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_slug_login
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enable_universal_link
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enforce_data_sharing_consent
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_country_display(*, *field=<django_countries.fields.CountryField: country>*)

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_default_language_display(*, *field=<django.db.models.fields.CharField: default_language>*)

get_enforce_data_sharing_consent_display(*, *field=<django.db.models.fields.CharField: enforce_data_sharing_consent>*)

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

hide_course_original_price
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

hide_labor_market_data
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

property instance

instance_type
alias of *enterprise.models.EnterpriseCustomer*

modified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

replace_sensitive_sso_username
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

reply_to
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()
URL for this change in the default admin site.

sender_alias
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

site
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

site_id

slug
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalEnterpriseCustomerCatalog**(*created*, *modified*, *uuid*, *title*, *content_filter*, *enabled_course_modes*, *publish_audit_enrollment_urls*, *enterprise_customer*, *enterprise_catalog_query*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: django.core.exceptions.MultipleObjectsReturned

content_filter

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enabled_course_modes

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_catalog_query

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

enterprise_catalog_query_id

enterprise_customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

enterprise_customer_id

static get_default_history_user(*instance*)

Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

property instance

instance_type

alias of *enterprise.models.EnterpriseCustomerCatalog*

modified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record

Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record

Get the previous history record for the instance. *None* if first.

publish_audit_enrollment_urls

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()
URL for this change in the default admin site.

title
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalEnterpriseCustomerInviteKey**(*created*, *modified*, *is_removed*, *uuid*, *usage_limit*, *expiration_date*, *is_active*, *enterprise_customer*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

enterprise_customer_id

expiration_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static **get_default_history_user**(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_expiration_date(*, *field=<django.db.models.fields.DateTimeField: expiration_date>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_expiration_date(*, *field=<django.db.models.fields.DateTimeField: expiration_date>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

property instance

instance_type

alias of *enterprise.models.EnterpriseCustomerInviteKey*

is_active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_removed

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

modified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record

Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

usage_limit
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalEnterpriseCustomerUser**(*id*, *created*, *modified*, *user_id*, *active*, *linked*, *is_relinkable*, *should_inactivate_other_customers*, *enterprise_customer*, *invite_key*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

enterprise_customer_id

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.EnterpriseCustomerUser*

invite_key

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

invite_key_id

is_relinkable
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

linked
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

modified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

should_inactivate_other_customers
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalLicensedEnterpriseCourseEnrollment**(*id*, *created*, *modified*, *license_uuid*, *is_revoked*, *enterprise_course_enrollment*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_course_enrollment
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_course_enrollment_id

static get_default_history_user(*instance*)

Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type
alias of *enterprise.models.LicensedEnterpriseCourseEnrollment*

is_revoked
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

license_uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

modified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class enterprise.models.**HistoricalPendingEnrollment**(*id*, *created*, *modified*, *course_id*, *course_mode*, *cohort_name*, *discount_percentage*, *sales_force_id*, *license_uuid*, *user*, *source*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist
Bases: *django.core.exceptions.ObjectDoesNotExist*

exception MultipleObjectsReturned
Bases: *django.core.exceptions.MultipleObjectsReturned*

cohort_name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_mode
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

discount_percentage
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.PendingEnrollment*

license_uuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

modified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

sales_force_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

source
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

source_id

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

class enterprise.models.**HistoricalPendingEnterpriseCustomerAdminUser**(*id*, *created*, *modified*, *user_email*, *enterprise_customer*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: `simple_history.models.HistoricalChanges`, `django.db.models.base.Model`

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.PendingEnterpriseCustomerAdminUser*

modified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record

Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record

Get the previous history record for the instance. *None* if first.

revert_url()

URL for this change in the default admin site.

user_email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalPendingEnterpriseCustomerUser**(*id*, *created*, *modified*, *user_email*, *enterprise_customer*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: `simple_history.models.HistoricalChanges`, `django.db.models.base.Model`

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

created

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

static get_default_history_user(*instance*)

Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.PendingEnterpriseCustomerUser*

modified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

user_email
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**HistoricalSystemWideEnterpriseUserRoleAssignment**(*id*, *created*, *modified*, *applies_to_all_contexts*, *user*, *role*, *enterprise_customer*, *history_id*, *history_date*, *history_change_reason*, *history_type*, *history_user*)

Bases: simple_history.models.HistoricalChanges, django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

applies_to_all_contexts
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

enterprise_customer_id

static get_default_history_user(*instance*)
Returns the user specified by *get_user* method for manually creating historical objects

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property instance

instance_type

alias of *enterprise.models.SystemWideEnterpriseUserRoleAssignment*

modified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

property next_record

Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

property prev_record

Get the previous history record for the instance. *None* if first.

revert_url()

URL for this change in the default admin site.

role

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

role_id

user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

class enterprise.models.**LicensedEnterpriseCourseEnrollment**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

An Enterprise Course Enrollment that is enrolled via a license.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

classmethod enrollments_for_user(*enterprise_customer_user*)

Returns a QuerySet of LicensedEnterpriseCourseEnrollments, along with their associated (hydrated) enterprise enrollments, users, and customers.

enterprise_course_enrollment

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

enterprise_course_enrollment_id

get_next_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=True, **kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_revoked

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

license_uuid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

revoke()

Marks this object as revoked and marks the associated EnterpriseCourseEnrollment as "saved for later". This object and the associated EnterpriseCourseEnrollment are both saved.

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class enterprise.models.**PendingEnrollment**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Track future enrollments for PendingEnterpriseCustomerUser.

Store a course ID, an intended enrollment mode, and a link to a PendingEnterpriseCustomerUser; when the PendingEnterpriseCustomerUser is converted to a full EnterpriseCustomerUser, API calls will be made to enroll the newly-created user in whatever courses have been added.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

cohort_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

course_mode

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

discount_percentage
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

license_uuid
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

sales_force_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

source
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

source_id

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

class enterprise.models.**PendingEnterpriseCustomerAdminUser**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Model for pending enterprise admin users.

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

admin_registration_url

Returns a URL to be used by a pending enterprise admin user to register their account.

enterprise_customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

get_next_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=True, **kwargs*)

get_next_by_modified(**, field=<model_utils.fields.AutoLastModifiedField: modified>, is_next=True, **kwargs*)

get_previous_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=False, **kwargs*)

get_previous_by_modified(**, field=<model_utils.fields.AutoLastModifiedField: modified>, is_next=False, **kwargs*)

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

user_email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**PendingEnterpriseCustomerUser**(**args*, ***kwargs*)

Bases: `model_utils.models.TimeStampedModel`

Model that stores "future members" of enterprise customer.

Fields

- **enterprise_customer** (ForeignKey[*EnterpriseCustomer*]) – enterprise customer
- **user_email** (`django.db.models.EmailField`) – user email

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

enterprise_customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

fulfill_pending_course_enrollments(*enterprise_customer_user*)
Enrolls a newly created EnterpriseCustomerUser in any courses attached to their PendingEnterpriseCustomerUser record.

Parameters **enterprise_customer_user** – a EnterpriseCustomerUser instance

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

link_pending_enterprise_user(*user*, *is_user_created*)
Link a PendingEnterpriseCustomerUser to the appropriate EnterpriseCustomer by creating or updating an EnterpriseCustomerUser record.

Parameters

- **is_user_created** – a boolean whether the User instance was created or updated
- **user** – a User instance

Returns: an EnterpriseCustomerUser instance

objects = <django.db.models.manager.Manager object>

pendingenrollment_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

user_email
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class enterprise.models.**SystemWideEnterpriseRole**(**args*, ***kwargs*)
Bases: `edx_rbac.models.UserRole`

System wide user role definitions specific to Enterprise.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

get_next_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=True, **kwargs*)

get_next_by_modified(**, field=<model_utils.fields.AutoLastModifiedField: modified>, is_next=True, **kwargs*)

get_previous_by_created(**, field=<model_utils.fields.AutoCreatedField: created>, is_next=False, **kwargs*)

get_previous_by_modified(**, field=<model_utils.fields.AutoLastModifiedField: modified>, is_next=False, **kwargs*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

system_wide_role_assignments
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

class enterprise.models.**SystemWideEnterpriseUserRoleAssignment**(**args*, ***kwargs*)
Bases: `edx_rbac.models.UserRoleAssignment`

Model to map users to a SystemWideEnterpriseRole.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

enterprise_customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise_customer_id

classmethod get_assignments(*user*, *role_names=None*)

Return an iterator of (rolename, [enterprise customer uuids]) for the given user (and maybe role_names).

Differs from super().get_assignments(...) in that it yields (role name, customer uuid list) pairs such that the first item in the customer uuid list for each role corresponds to the currently *active* EnterpriseCustomerUser for the user.

The resulting generated pairs are sorted by role name, and within role_name, by (active, customer uuid). For example:

> ('enterprise_admin', ['active-enterprise-uuid', 'inactive-enterprise-uuid', 'other-inactive-enterprise-uuid']) ('enterprise_learner', ['active-enterprise-uuid', 'inactive-enterprise-uuid']), ('enterprise_openedx_operator', ['*'])

get_context()

Return a non-empty list of contexts for which `self.user` is assigned `self.role`.

classmethod get_distinct_assignments_by_role_name(*user*, *role_names=None*)

Returns a mapping of role names to sets of enterprise customer uuids for which the user is assigned that role.

get_next_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=True*, ***kwargs*)

get_next_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<model_utils.fields.AutoCreatedField: created>*, *is_next=False*, ***kwargs*)

get_previous_by_modified(*, *field=<model_utils.fields.AutoLastModifiedField: modified>*, *is_next=False*, ***kwargs*)

has_access_to_all_contexts()

Returns true if the role for this assignment is `ENTERPRISE_OPERATOR_ROLE`, or if `applies_to_all_contexts` is true; returns false otherwise.

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

role

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

role_id

save(**args*, ***kwargs*)
Overriding the save method in order to make sure that modified field is updated even if it is not given as a parameter to the update field argument.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

enterprise.models.**get_default_customer_type**()
Get default enterprise customer type id to use when creating a new EnterpriseCustomer model.

8.1.11 enterprise.roles_api module

Python API for doing CRUD operations on roles and user role assignments.

enterprise.roles_api.**admin_role**()
Return the enterprise admin role.

enterprise.roles_api.**assign_admin_role**(*user*, *enterprise_customer=None*, *applies_to_all_contexts=False*)
Assigns the given user the *enterprise_admin* role in the given customer.

enterprise.roles_api.**assign_learner_role**(*user*, *enterprise_customer=None*, *applies_to_all_contexts=False*)
Assigns the given user the *enterprise_learner* role in the given customer.

enterprise.roles_api.**catalog_admin_role**()
Returns the enterprise catalog admin role.

enterprise.roles_api.**delete_admin_role_assignment**(*user*, *enterprise_customer=None*)
Deletes the admin role assignment for the given user in the given enterprise customer. If *enterprise_customer* is null, will delete *every* admin role assignment for this user.

enterprise.roles_api.**delete_learner_role_assignment**(*user*, *enterprise_customer=None*)
Deletes the learner role assignment for the given user in the given enterprise customer. If *enterprise_customer* is null, will delete *every* learner role assignment for this user.

enterprise.roles_api.**delete_role_assignment**(*user*, *role*, *enterprise_customer=None*)
Deletes the given role assignment for the given user in the given enterprise customer. If *enterprise_customer* is null, will delete *every* role assignment for this user.

`enterprise.roles_api.get_or_create_system_wide_role(role_name)`
Gets or creates the *SystemWideEnterpriseRole* with the given name. Caches the result for 60 minutes in the Django cache. Returns the role object.

`enterprise.roles_api.learner_role()`
Returns the enterprise learner role.

`enterprise.roles_api.openedx_operator_role()`
Returns the enterprise openedx operator role.

`enterprise.roles_api.roles_by_name()`
Returns a mapping of system wide roles by name.

8.1.12 enterprise.rules module

Rules needed to restrict access to the enterprise data api.

8.1.13 enterprise.signals module

Django signal handlers.

`enterprise.signals.assign_or_delete_enterprise_learner_role(sender, instance, **kwargs)`
Assign or delete enterprise_learner role for EnterpriseCustomerUser when created or updated.

The enterprise_learner role is assigned when a new EnterpriseCustomerUser record is initially created and removed when a EnterpriseCustomerUser record is updated and unlinked (i.e., soft delete - see ENT-2538).

`enterprise.signals.create_enterprise_enrollment_receiver(sender, instance, **kwargs)`
Watches for post_save signal for creates on the CourseEnrollment table.

Spin off an async task to generate an EnterpriseCourseEnrollment if appropriate.

`enterprise.signals.create_pending_enterprise_admin_user(sender, instance, **kwargs)`
Creates a PendingEnterpriseCustomerUser when a PendingEnterpriseCustomerAdminUser is created.

`enterprise.signals.default_content_filter(sender, instance, **kwargs)`
Set default value for *EnterpriseCustomerCatalog.content_filter* if not already set.

`enterprise.signals.delete_enterprise_admin_role_assignment(sender, instance, **kwargs)`
Delete the associated enterprise admin role assignment record when deleting an EnterpriseCustomerUser record.

`enterprise.signals.delete_enterprise_analytics_user(sender, instance, **kwargs)`
Delete the associated enterprise analytics user in Tableau.

`enterprise.signals.delete_enterprise_catalog_data(sender, instance, **kwargs)`
Send deletions of Enterprise Catalogs to the Enterprise Catalog Service.

`enterprise.signals.delete_enterprise_learner_role_assignment(sender, instance, **kwargs)`
Delete the associated enterprise learner role assignment record when deleting an EnterpriseCustomerUser record.

`enterprise.signals.delete_pending_enterprise_admin_user(sender, instance, **kwargs)`
Deletes a PendingEnterpriseCustomerUser when its associated PendingEnterpriseCustomerAdminUser is removed.

`enterprise.signals.handle_user_post_save(sender, **kwargs)`
Handle User model changes. Context: This signal runs any time a user logs in, including b2c users.

Steps:

1. Check for existing PendingEnterpriseCustomerUser(s) for user's email. If one or more exists, create an EnterpriseCustomerUser record for each which will ensure the user has the "enterprise_learner" role.

2. When we get a new EnterpriseCustomerUser record (or an existing record if one existed), check if the PendingEnterpriseCustomerUser has any pending course enrollments. If so, enroll the user in these courses.
3. Delete the PendingEnterpriseCustomerUser record as its no longer needed.
4. Using the newly created EnterpriseCustomerUser (or an existing record if one existed), check if there is a PendingEnterpriseCustomerAdminUser. If so, ensure the user has the "enterprise_admin" role and a Tableau user is created for the user.

`enterprise.signals.`**`save_logo_file`**(*sender*, *instance*, ***kwargs*)
Now that the object is instantiated and instance.id exists, save the image at correct path and re-save the model.

`enterprise.signals.`**`skip_saving_logo_file`**(*sender*, *instance*, ***kwargs*)
To avoid saving the logo image at an incorrect path, skip saving it.

`enterprise.signals.`**`update_enterprise_catalog_data`**(*sender*, *instance*, ***kwargs*)
Send data changes to Enterprise Catalogs to the Enterprise Catalog Service.

Additionally sends a request to update the catalog's metadata from discovery, and index any relevant content for Algolia.

`enterprise.signals.`**`update_enterprise_catalog_query`**(*sender*, *instance*, ***kwargs*)
Sync data changes from Enterprise Catalog Query to the Enterprise Customer Catalog.

`enterprise.signals.`**`update_lang_pref_of_all_learners`**(*sender*, *instance*, ***kwargs*)
Update the language preference of all the learners belonging to the enterprise customer. Set the language preference to the value enterprise customer has used as the *default_language*.

`enterprise.signals.`**`update_learner_language_preference`**(*sender*, *instance*, *created*, ***kwargs*)
Update the language preference of the learner. Set the language preference to the value enterprise customer has used as the *default_language*.

8.1.14 enterprise.tasks module

Django tasks.

`enterprise.tasks.`**`enterprise_course_enrollment_model`**()
Returns the `EnterpriseCourseEnrollment` class. This function is needed to avoid circular ref issues when model classes call tasks in this module.

`enterprise.tasks.`**`enterprise_customer_user_model`**()
Returns the `EnterpriseCustomerUser` class. This function is needed to avoid circular ref issues when model classes call tasks in this module.

`enterprise.tasks.`**`enterprise_enrollment_source_model`**()
Returns the `EnterpriseEnrollmentSource` class. This function is needed to avoid circular ref issues when model classes call tasks in this module.

8.1.15 enterprise.tpa_pipeline module

Module provides elements to be used in third-party auth pipeline.

`enterprise.tpa_pipeline.get_default_idp_user_social_auth(enterprise_customer, user=None, user_idp_id=None)`

Return social auth entry of user for given enterprise default IDP.

Parameters

- **user** (*User*) – user object
- **enterprise_customer** (EnterpriseCustomer) – Instance of the enterprise customer.
- **user_idp_id** (*str*) – User id of user in third party LMS

`enterprise.tpa_pipeline.get_enterprise_customer_for_running_pipeline(request, pipeline)`

Get the EnterpriseCustomer associated with a running pipeline.

`enterprise.tpa_pipeline.get_enterprise_customer_for_sso(sso_provider_id)`

Get the EnterpriseCustomer object tied to an identity provider.

`enterprise.tpa_pipeline.get_sso_provider(request, pipeline)`

Helper method to retrieve the sso provider ID from either a user's SSO login request

`enterprise.tpa_pipeline.get_user_from_social_auth(tpa_providers, user_id, enterprise_customer)`

Find the LMS user from the LMS model *UserSocialAuth*.

Parameters

- **tpa_providers** (*third_party_auth.provider*) – list of third party auth provider objects
- **user_id** (*str*) – User id of user in third party LMS
- **enterprise_customer** (EnterpriseCustomer) – Instance of the enterprise customer.

`enterprise.tpa_pipeline.get_user_social_auth(user, enterprise_customer)`

Return social auth entry of user for given enterprise.

Parameters

- **user** (*User*) – user object
- **enterprise_customer** (EnterpriseCustomer) – Instance of the enterprise customer.

`enterprise.tpa_pipeline.handle_enterprise_logistration(backend, user, **kwargs)`

Perform the linking of user in the process of logging to the Enterprise Customer.

Parameters

- **backend** – The class handling the SSO interaction (SAML, OAuth, etc)
- **user** – The user object in the process of being logged in with
- ****kwargs** – Any remaining pipeline variables

`enterprise.tpa_pipeline.handle_redirect_after_social_auth_login(backend, user)`

Change the redirect url if user has more than 1 EnterpriseCustomer associations.

Parameters

- **backend** (*User*) – social auth backend object
- **user** (*User*) – user object

enterprise.tpa_pipeline.**select_enterprise_page_as_redirect_url**(*strategy*)
Change the redirect url for the user to enterprise selection page.

enterprise.tpa_pipeline.**validate_provider_config**(*enterprise_customer*, *sso_provider_id*)
Helper method to ensure that a customer's provider config is validated

8.1.16 enterprise.urls module

URLs for enterprise.

8.1.17 enterprise.utils module

Utility functions for enterprise app.

exception enterprise.utils.**CourseCatalogApiError**
Bases: Exception

Exception to raise when we we received data from Course Catalog but it contained an error.

exception enterprise.utils.**CourseEnrollmentDowngradeError**
Bases: Exception

Exception to raise when an enrollment attempts to enroll the user in an unpaid mode when they are in a paid mode.

exception enterprise.utils.**CourseEnrollmentPermissionError**
Bases: Exception

Exception to raise when an enterprise attempts to use enrollment features it's not configured to use.

exception enterprise.utils.**NotConnectedToOpenEdX**(**args*, ***kwargs*)
Bases: Exception

Exception to raise when not connected to OpenEdX.

In general, this exception shouldn't be raised, because this package is designed to be installed directly inside an existing OpenEdX platform.

class enterprise.utils.**ValidationMessages**
Bases: object

Namespace class for validation messages.

BOTH_COURSE_FIELDS_SPECIFIED = 'Either "CSV bulk upload" or a singular course ID may be used for manual enrollments, but not both together.'

BOTH_FIELDS_SPECIFIED = 'Either "Email or Username" or "CSV bulk upload" must be specified, but both were.'

BULK_LINK_FAILED = 'Error: Learners could not be added. Correct the following errors.'

COURSE_MODE_INVALID_FOR_COURSE = 'Enrollment track {course_mode} is not available for course {course_id}.'

COURSE_NOT_EXIST_IN_CATALOG = "Course ID {course_id} doesn't exist in Enterprise Customer's Catalog"

COURSE_WITHOUT_COURSE_MODE = 'Select a course enrollment track for the given course(s).'

INVALID_CHANNEL_WORKER = 'Enterprise channel worker user with the username "{channel_worker_username}" was not found.'

INVALID_COURSE_ID = 'Could not retrieve details for the course ID {course_id}. Specify a valid ID.'

INVALID_DISCOUNT = 'Discount percentage should be from 0 to 100.'

INVALID_EMAIL = '{argument} does not appear to be a valid email address.'

INVALID_EMAIL_OR_USERNAME = '{argument} does not appear to be a valid email address or known username'

INVALID_ENCODING = "Unable to parse CSV file. Please make sure it is a CSV 'utf-8' encoded file."

MISSING_EXPECTED_COLUMNS = 'Expected a CSV file with [{expected_columns}] columns, but found [{actual_columns}] columns instead.'

MISSING_REASON = 'Reason field is required but was not filled.'

NO_FIELDS_SPECIFIED = 'Either "Email or Username" or "CSV bulk upload" must be specified, but neither were.'

USER_ALREADY_REGISTERED = 'User with email address {email} is already registered with Enterprise Customer {ec_name}'

USER_NOT_EXIST = "User with email address {email} doesn't exist."

USER_NOT_LINKED = 'User is not linked with Enterprise Customer'

`enterprise.utils.`**`add_user_to_tableau_group`**(*tableau_user_id*, *tableau_group*)

Associate a user with the group in tableau. The function attempts to create the group, except when the server responds with an exception code of 409009, indicating that the group already exists. In the latter case, the user is associated with the existing group.

`enterprise.utils.`**`batch`**(*iterable*, *batch_size=1*)

Break up an iterable into equal-sized batches.

Parameters

- **iterable** (*e.g. list*) – an iterable to batch
- **batch_size** (*int*) – the size of each batch. Defaults to 1.

Returns iterates through each batch of an iterable

Return type generator

`enterprise.utils.`**`clean_html_for_template_rendering`**(*text*)

Given html text that will be rendered as a variable in a template, strip out characters that impact rendering.

Parameters **text** (*str*) – The text to clean.

Returns The cleaned text.

Return type (str)

`enterprise.utils.`**`create_tableau_user`**(*user_id*, *enterprise_customer_user*)

Create the user in tableau and store the tableau user in the EnterpriseAnalyticsUser model.

`enterprise.utils.`**`customer_admin_enroll_user`**(*enterprise_customer*, *user*, *course_mode*, *course_id*, *enrollment_source=None*)

For use with bulk enrollment, or any use case of admin enrolling a user

Enroll a single user in a course using a particular course mode, indicating it's a customer_admin enrolling a user (such as bulk enrollment)

Parameters

- **enterprise_customer** – The EnterpriseCustomer model object which is sponsoring the enrollment
- **user** – The user model object who needs to be enrolled in the course
- **course_mode** – The string representation of the mode with which the enrollment should be created
- **course_id** – An opaque course_id to enroll in

Returns Whether or not enrollment succeeded for the course specified

Return type succeeded (Boolean)

`enterprise.utils.`**`customer_admin_enroll_user_with_status`**(*enterprise_customer*, *user*, *course_mode*, *course_id*, *enrollment_source=None*, *license_uuid=None*)

For use with bulk enrollment, or any use case of admin enrolling a user

Enroll a single user in a course using a particular course mode, indicating it's a customer_admin enrolling a user (such as bulk enrollment). Return a status based on whether the enrollment existed before attempting to enroll.

TODO: The *enroll_user* function above, used by Django admin, for example, should also be rewired to use this new ability, but for now it still uses enrollment client.

Parameters

- **enterprise_customer** – The EnterpriseCustomer model object which is sponsoring the enrollment
- **user** – The user model object who needs to be enrolled in the course
- **course_mode** – The string representation of the mode with which the enrollment should be created
- **course_id** – An opaque course_id to enroll in
- **enrollment_source** – Source of enrollment, used for tracking enrollments
- **license_uuid** – UUID of associated license with the enrollment, used to create a mapping between licenses and enrollments

Returns Whether or not the enrollment succeeded for the course specified created (Boolean): Whether or not the enrollment existed prior to calling method

Return type succeeded (Boolean)

`enterprise.utils.`**`delete_data_sharing_consent`**(*course_id*, *customer_uuid*, *user_email*)

Delete the DSC records from the DB for given learner, course and customer, also its cache.

`enterprise.utils.`**`delete_tableau_user`**(*enterprise_customer_user*)

Delete the user in Tableau.

`enterprise.utils.`**`delete_tableau_user_by_id`**(*tableau_user_id*)

Delete the user in Tableau.

`enterprise.utils.`**`discovery_query_url`**(*content_filter*, *html_format=True*)

Return discovery url for preview.

`enterprise.utils.`**`enroll_licensed_users_in_courses`**(*enterprise_customer*, *licensed_users_info*, *discount=100.0*)

Takes a list of licensed learner data and enrolls each learner in the requested courses.

Parameters

- **enterprise_customer** – The EnterpriseCustomer (object) which is sponsoring the enrollment
- **licensed_users_info** – (list) An array of dictionaries, each containing information necessary to create a licensed enterprise enrollment for a specific learner in a specified course run. Example:

```
licensed_users_info: [
    {
        'email': 'newuser@test.com',
        'course_run_key': 'course-v1:edX+DemoX+Demo_Course',
        'course_mode': 'verified',
        'license_uuid': '5b77bdbade7b4fcb838f8111b68e18ae'
    }
]
```

- **discount** – (int) the discount offered to the learner for their enrollment. Subscription based enrollments default to 100

Expected Return Values:

```
Results: {
    successes: [{ 'email': <email>, 'course_run_key': <key>, 'user': <user object> }
↪ ... ],
    pending: [{ 'email': <email>, 'course_run_key': <key>, 'user': <user object> } .
↪.. ],
    failures: [{ 'email': <email>, 'course_run_key': <key> } ... ]
}
```

`enterprise.utils.`**`enroll_user`**(*enterprise_customer*, *user*, *course_mode*, **course_ids*, ***kwargs*)

Enroll a single user in any number of courses using a particular course mode.

Parameters

- **enterprise_customer** – The EnterpriseCustomer model object which is sponsoring the enrollment
- **user** – The user model object who needs to be enrolled in the course
- **course_mode** – The string representation of the mode with which the enrollment should be created
- ***course_ids** – An iterable containing any number of course IDs to eventually enroll the user in.
- **kwargs** – Should contain enrollment_client if it's already been instantiated and should be passed in.

Returns Whether or not enrollment succeeded for all courses specified

Return type Boolean

enterprise.utils.**enroll_users_in_course**(*enterprise_customer*, *course_id*, *course_mode*, *emails*, *enrollment_requester=None*, *enrollment_reason=None*, *discount=0.0*, *sales_force_id=None*)

Enroll existing users in a course, and create a pending enrollment for nonexisting users.

Parameters

- **enterprise_customer** – The EnterpriseCustomer which is sponsoring the enrollment
- **course_id** (*str*) – The unique identifier of the course in which we're enrolling
- **course_mode** (*str*) – The mode with which we're enrolling in the course
- **emails** – An iterable of email addresses which need to be enrolled
- **enrollment_requester** (*User*) – Admin user who is requesting the enrollment.
- **enrollment_reason** (*str*) – A reason for enrollment.
- **discount** (*Decimal*) – Percentage discount for enrollment.
- **sales_force_id** (*str*) – Salesforce opportunity id.

Returns

A list of users who were successfully enrolled in the course.

pending: A list of PendingEnterpriseCustomerUsers who were successfully linked and had pending enrollments created for them in the database.

failures: A list of users who could not be enrolled in the course.

Return type successes

enterprise.utils.**enterprise_course_enrollment_model**()

Returns the `EnterpriseCourseEnrollment` class.

enterprise.utils.**enterprise_customer_invite_key_model**()

Returns the `EnterpriseCustomerInviteKey` class.

enterprise.utils.**enterprise_customer_model**()

Returns the `EnterpriseCustomer` class.

enterprise.utils.**enterprise_customer_user_model**()

Returns the `EnterpriseCustomerUser` class.

enterprise.utils.**enterprise_enrollment_source_model**()

Returns the `EnterpriseEnrollmentSource` class.

enterprise.utils.**filter_audit_course_modes**(*enterprise_customer*, *course_modes*)

Filter audit course modes out if the enterprise customer has not enabled the 'Enable audit enrollment' flag.

Parameters

- **enterprise_customer** – The EnterpriseCustomer that the enrollment was created using.
- **course_modes** – iterable with dictionaries containing a required 'mode' key

enterprise.utils.**find_enroll_email_template**(*enterprise_customer*, *template_type*)

Find email template from the template database represented by EnrollmentNotificationEmailTemplate model.

Parameters

- **enterprise_customer** (-) – the customer model

- **template_type** (-) – type of template to fetch, must be one of: enterprise.utils.SELF_ENROLL_EMAIL_TEMPLATE_TYPE, or enterprise.utils.ADMIN_ENROLL_EMAIL_TEMPLATE_TYPE

Returns Customer specific template if found. Default template for the given type if found. None if neither default template, nor per customer template found.

`enterprise.utils.format_price(price, currency='$')`

Format the price to have the appropriate currency and digits..

Parameters

- **price** – The price amount.
- **currency** – The currency for the price.

Returns A formatted price string, i.e. '$10', '$10.52'.

`enterprise.utils.get_active_course_runs(course, users_all_enrolled_courses)`

Return active course runs (user is enrolled in) of the given course.

This function will return the course_runs of 'course' which have active enrollment by looking into 'users_all_enrolled_courses'

`enterprise.utils.get_advertised_course_run(course)`

Find the advertised course run for a given course :param course: course dict :type course: dict

Returns a course_run or None

Return type dict

`enterprise.utils.get_all_field_names(model, excluded=None)`

Return all fields' names from a model. Filter out the field names present in *excluded*.

According to Django documentation, `get_all_field_names` should become some monstrosity with chained iterable ternary nested in a list comprehension. For now, a simpler version of iterating over fields and getting their names work, but we might have to switch to full version in future.

`enterprise.utils.get_best_mode_from_course_key(course_key)`

Helper method to retrieve a list of enrollments for a given course and select the one most applicable to enroll an enterprise learner in.

`enterprise.utils.get_cache_key(**kwargs)`

Wrapper method on edx_django_utils get_cache_key utility.

`enterprise.utils.get_catalog_admin_url(catalog_id)`

Get url to catalog details admin page.

Parameters **catalog_id** (*int*) – Catalog id for which to return catalog details url.

Returns URL pointing to catalog details admin page for the give catalog id.

Example

```
>>> get_catalog_admin_url_template(2)
"http://localhost:18381/admin/catalogs/catalog/2/change/"
```

`enterprise.utils.get_catalog_admin_url_template(mode='change')`

Get template of catalog admin url.

URL template will contain a placeholder '{catalog_id}' for catalog id. :param mode e.g. change/add.:

Returns A string containing template for catalog url.

Example

```
>>> get_catalog_admin_url_template('change')
"http://localhost:18381/admin/catalogs/catalog/{catalog_id}/change/"
```

`enterprise.utils.`**`get_closest_course_run`**(*course_runs*)
Return course run with start date closest to now.

`enterprise.utils.`**`get_configuration_value`**(*val_name*, *default=None*, ***kwargs*)
Get a configuration value, or fall back to `default` if it doesn't exist.

Also takes a *type* argument to guide which particular upstream method to use when trying to retrieve a value. Current types include:

- *url* to specifically get a URL.

`enterprise.utils.`**`get_configuration_value_for_site`**(*site*, *key*, *default=None*)
Get the site configuration value for a key, unless a site configuration does not exist for that site.

Useful for testing when no Site Configuration exists in edx-enterprise or if a site in LMS doesn't have a configuration tied to it.

Parameters

- **site** – A Site model object
- **key** – The name of the value to retrieve
- **default** – The default response if there's no key in site config or settings

Returns The value located at that key in the site configuration or settings file.

`enterprise.utils.`**`get_content_metadata_item_id`**(*content_metadata_item*)
Return the unique identifier given a content metadata item dictionary.

`enterprise.utils.`**`get_course_run_duration_info`**(*course_run*)
Return course run's duration(str) info.

`enterprise.utils.`**`get_course_run_start`**(*course_run*, *default=None*)
Return the given course run's start date as a datetime.

`enterprise.utils.`**`get_course_track_selection_url`**(*course_run*, *query_parameters*)
Return track selection url for the given course.

Parameters

- **course_run** (*dict*) – A dictionary containing course run metadata.
- **query_parameters** (*dict*) – A dictionary containing query parameters to be added to course selection url.

Raises (**KeyError**) – Raised when course run dict does not have 'key' key.

Returns Course track selection url.

Return type (str)

`enterprise.utils.`**`get_create_ent_enrollment`**(*course_id*, *enterprise_customer_user*, *enterprise_enrollment_source*, *license_uuid=None*)
Get or Create the Enterprise Course Enrollment.

If `license_uuid` present, will also create a LicensedEnterpriseCourseEnrollment record.

`enterprise.utils.`**`get_current_course_run`**(*course*, *users_active_course_runs*)
Return the current course run on the following conditions:

- If user has active course runs (already enrolled) then return course run with closest start date

Otherwise it will check the following logic:

- Course run is enrollable (see is_course_run_enrollable)
- Course run has a verified seat and the upgrade deadline has not expired.
- If no enrollable/upgradeable course runs, then select all the course runs.
- After filtering the course runs checks whether the filtered course run is about to close or not if yes then return the next course run or the current one.

`enterprise.utils.get_default_catalog_content_filter()`
Return default enterprise customer catalog content filter.

`enterprise.utils.get_duration_of_course_or_courserun(content_metadata_item)`
Returns duration start, end dates given a piece of content_metadata item If course item, extracts start, end dates of closest course run based on current timestamp

Returns in days or 0 start: start field of closest course run item, or None end: end field of closest course run item, or None

Return type duration

`enterprise.utils.get_ecommerce_worker_user()`
Return the user object of ecommerce worker user.

`enterprise.utils.get_enterprise_customer(uuid)`
Get the `EnterpriseCustomer` instance associated with `uuid`.

Parameters `uuid` – The universally unique ID of the enterprise customer.

Returns The `EnterpriseCustomer` instance, or `None` if it doesn't exist.

`enterprise.utils.get_enterprise_customer_by_invite_key_or_404(invite_key_uuid)`
Given an EnterpriseCustomerInviteKey UUID, return the corresponding EnterpriseCustomer or raise a 404.

Parameters `invite_key_uuid` (*str*) – The UUID identifying an EnterpriseCustomerInviteKey.

Returns The EnterpriseCustomer given the EnterpriseCustomerInviteKey UUID.

Return type (*EnterpriseCustomer*)

`enterprise.utils.get_enterprise_customer_by_slug_or_404(slug)`
Given an EnterpriseCustomer slug, return the corresponding EnterpriseCustomer or raise a 404.

Parameters `slug` (*str*) – The unique slug (a string) identifying the customer.

Returns The EnterpriseCustomer given the slug.

Return type (*EnterpriseCustomer*)

`enterprise.utils.get_enterprise_customer_for_user(auth_user)`
Return first found enterprise customer instance for given user.

Some users are associated with an enterprise customer via *EnterpriseCustomerUser* model,

1. if given user is associated with any enterprise customer, return first enterprise customer.
2. otherwise return *None*.

Parameters `auth_user` (*contrib.auth.User*) – Django User

Returns enterprise customer associated with the current user.

Return type (*EnterpriseCustomer*)

`enterprise.utils.`**`get_enterprise_customer_or_404`**(*enterprise_uuid*)

Given an EnterpriseCustomer UUID, return the corresponding EnterpriseCustomer or raise a 404.

Parameters **enterprise_uuid** (*str*) – The UUID (in string form) of the EnterpriseCustomer to fetch.

Returns The EnterpriseCustomer given the UUID.

Return type (*EnterpriseCustomer*)

`enterprise.utils.`**`get_enterprise_customer_user`**(*user_id*, *enterprise_uuid*)

Return the object for EnterpriseCustomerUser.

Parameters

- **user_id** (*str*) – user identifier
- **enterprise_uuid** (*UUID*) – Universally unique identifier for the enterprise customer.

Returns enterprise customer user record

Return type (*EnterpriseCustomerUser*)

`enterprise.utils.`**`get_enterprise_utm_context`**(*enterprise_customer*)

Get the UTM context for the enterprise.

`enterprise.utils.`**`get_enterprise_uuids_for_user_and_course`**(*auth_user*, *course_run_id*, *is_customer_active=None*)

Get the `EnterpriseCustomer` UUID(s) associated with a user and a course id``.

Some users are associated with an enterprise customer via *EnterpriseCustomerUser* model,

1. if given user is enrolled in a specific course via an enterprise customer enrollment, return related enterprise customers as a list.
2. otherwise return empty list.

Parameters

- **auth_user** (*contrib.auth.User*) – Django User
- **course_run_id** (*str*) – Course Run to lookup an enrollment against.
- **active** – (boolean or None): Filter flag for returning active, inactive, or all uuids

Returns enterprise customer uuids associated with the current user and course run or None

Return type (list of str)

`enterprise.utils.`**`get_enterprise_worker_user`**()

Return the user object of enterprise worker user.

`enterprise.utils.`**`get_identity_provider`**(*provider_id*)

Get Identity Provider with given id.

Returns Instance of ProviderConfig or None.

`enterprise.utils.`**`get_idiff_list`**(*list_a*, *list_b*)

Returns a list containing lower case difference of list_b and list_a after case insensitive comparison.

Parameters

- **list_a** – list of strings
- **list_b** – list of string

Returns List of unique lower case strings computed by subtracting list_b from list_a.

`enterprise.utils.get_idp_choices()`
Get a list of identity providers choices for enterprise customer.

Returns A list of choices of all identity providers, None if it can not get any available identity provider.

`enterprise.utils.get_language_code(language)`
Return IETF language tag of given language name.

- if language is not found in the language map then *en-US* is returned.

Parameters **language** (*str*) – string language name

Returns: a language tag (two-letter language code - two letter country code if applicable)

`enterprise.utils.get_last_course_run_end_date(course_runs)`
Returns the end date of the course run that falls at the end.

`enterprise.utils.get_learner_portal_url(enterprise_customer)`
Learner portal url for enterprise_customer

`enterprise.utils.get_notification_subject_line(course_name, template_configuration=None)`
Get a subject line for a notification email.

The method is designed to fail in a "smart" way; if we can't render a database-backed subject line template, then we'll fall back to a template saved in the Django settings; if we can't render _that_ one, then we'll fall through to a friendly string written into the code.

One example of a failure case in which we want to fall back to a stock template would be if a site admin entered a subject line string that contained a template tag that wasn't available, causing a KeyError to be raised.

Parameters

- **course_name** (*str*) – Course name to be rendered into the string
- **template_configuration** – A database-backed object with a stored subject line template

`enterprise.utils.get_oauth2authentication_class()`
Return oauth2 authentication class to authenticate REST APIs with Bearer token.

`enterprise.utils.get_platform_logo_url()`
Return an absolute URL of the platform logo using the branding api.

`enterprise.utils.get_program_type_description(program_type)`
Get the pre-set description associated with this program type.

Parameters **program_type** – The type of the program. Should be one of:

- "MicroMasters Certificate"
- "Professional Certificate"
- "XSeries Certificate"

Returns The description associated with the program type. If none exists, then the empty string.

`enterprise.utils.get_request_value(request, key, default=None)`
Get the value in the request, either through query parameters or posted data, from a key.

Parameters

- **request** – The request from which the value should be gotten.

- **key** – The key to use to get the desired value.
- **default** – The backup value to use in case the input key cannot help us get the value.

Returns The value we're looking for.

`enterprise.utils.get_social_auth_from_idp(idp, user=None, user_idp_id=None)`
Return social auth entry of user for given enterprise IDP.

idp (EnterpriseCustomerIdentityProvider): EnterpriseCustomerIdentityProvider Object user (User): User Object user_idp_id (str): User id of user in third party LMS

`enterprise.utils.get_tableau_server()`
Return Tableau ServerAuth and Server instances or None

`enterprise.utils.get_user_valid_idp(user, enterprise_customer)`
Return the default idp if it has user social auth record else it will return any idp with valid user social auth record

user (User): user object enterprise_customer (EnterpriseCustomer): EnterpriseCustomer object

`enterprise.utils.get_users_by_email(emails)`
Accept a list of emails, and separate them into users that exist on OpenEdX and users who don't.

Parameters **emails** – An iterable of email addresses to split between existing and nonexisting

Returns Queryset of users who exist in the OpenEdX platform and who were in the list of email addresses unregistered_emails: List of unique emails which were in the original list, but do not yet exist as users

Return type users

`enterprise.utils.has_course_run_available_for_enrollment(course_runs)`

Iterates over all course runs to check if there any course run that is available for enrollment.

Argument: course_runs: list of course runs

Returns True if found else false

`enterprise.utils.is_course_run_about_to_end(current_course_run)`
Return False if end - now > course run's weeks to complete otherwise True.

`enterprise.utils.is_course_run_active(course_run)`
Checks whether a course run is active. That is, whether the course run is published, enrollable, and marketable. :param course_run: The metadata about a course run. :type course_run: dict

Returns True if course run is "active"

Return type bool

`enterprise.utils.is_course_run_available_for_enrollment(course_run)`
Check if a course run is available for enrollment.

`enterprise.utils.is_course_run_enrollable(course_run)`
Return true if the course run is enrollable, false otherwise.

We look for the following criteria:

1. **end date is greater than a reasonably-defined enrollment window, or undefined.** A reasonably-defined enrollment window is 1 day before course run end date.
2. enrollment_start is less than now, or undefined.
3. enrollment_end is greater than now, or undefined.

enterprise.utils.**is_course_run_published**(*course_run*)
Return True if the course run's status value is "published".

enterprise.utils.**is_course_run_upgradeable**(*course_run*)
Return true if the course run has a verified seat with an unexpired upgrade deadline, false otherwise.

enterprise.utils.**is_pending_user**(*user*)
Returns true if pending_user attributes are detected

enterprise.utils.**is_user_enrolled**(*user*, *course_id*, *course_mode*, *enrollment_client=None*)
Query the enrollment API and determine if a learner is enrolled in a given course run track.

Parameters

- **user** – The user whose enrollment needs to be checked
- **course_mode** – The mode with which the enrollment should be checked
- **course_id** – course id of the course where enrollment should be checked.
- **enrollment_client** – An optional EnrollmentAPIClient if it's already been instantiated and should be passed in.

Returns Whether or not enrollment exists

Return type Boolean

enterprise.utils.**is_valid_url**(*url*)
Return where the specified URL is a valid absolute url.

enterprise.utils.**licensed_enterprise_course_enrollment_model**()
returns the `LicensedEnterpriseCourseEnrollment` class.

enterprise.utils.**localized_utcnow**()
Helper function to return localized utcnow().

enterprise.utils.**logo_path**(*instance*, *filename*)
Returns the enterprise customer logo image path.

Parameters

- **instance** (*EnterpriseCustomerBrandingConfiguration*) – EnterpriseCustomerBrandingConfiguration object
- **filename** (*str*) – file to upload

Returns path of image file e.g. enterprise/branding/<enterprise_uuid>/logo_<uuid>.<ext>.lower()

Return type path

enterprise.utils.**parse_datetime_handle_invalid**(*datetime_value*)
Return the parsed version of a datetime string. If the string is invalid, return None.

enterprise.utils.**parse_lms_api_datetime**(*datetime_string*, *datetime_format='%Y-%m-%dT%H:%M:%SZ'*)
Parse a received datetime into a timezone-aware, Python datetime object.

Parameters

- **datetime_string** – A string to be parsed.
- **datetime_format** – A datetime format string to be used for parsing

enterprise.utils.**send_email_notification_message**(*user*, *enrolled_in*, *dashboard_url*, *enterprise_customer_uuid*, *email_connection=None*, *admin_enrollment=False*)

Send an email notifying a user about their enrollment in a course.

Parameters

- **user** – a dict with either of the following forms: - 1: { 'first_name': name, 'username': user_name, 'email': email } (similar to a User object) - 2: { 'user_email' : user_email } (similar to a PendingEnterpriseCustomerUser object)
- **enrolled_in** (*dict*) – The dictionary contains details of the enrollable object (either course or program) that the user enrolled in. This MUST contain a *name* key, and MAY contain the other following keys:

```
- url: A human-friendly link to the enrollable's home page

- type: Either `course` or `program` at present

- branding: A special name for what the enrollable "is"; for␣
↪example,
    "MicroMasters" would be the branding for a "MicroMasters␣
↪Program"

- start: A datetime object indicating when the enrollable will be␣
↪available.
```

- **dashboard_url** – link to enterprise customer's unique homepage for user
- **enterprise_customer_uuid** – The EnterpriseCustomer uuid that the enrollment was created using.
- **email_connection** – An existing Django email connection that can be used without creating a new connection for each individual message
- **admin_enrollment** – If true, uses admin enrollment template instead of default ones.

enterprise.utils.**serialize_notification_content**(*enterprise_customer*, *course_details*, *course_id*, *users*, *admin_enrollment=False*, *activation_links=None*)

Prepare serializable contents to send emails with (if using tasks to send emails)

Parameters

- **enterprise_customer** (enterprise.models.EnterpriseCustomer) –
- **course_details** (*dict*) – With at least 'title', 'start' and 'course' keys (usually obtained via CourseCatalogApiClient)
- **course_id** (*str*) –
- **users** (*list*) – list of users to enroll (each user should be a User or PendingEnterpriseCustomerUser)
- **activation_links** (*dict*) – a dictionary map of unactivated license user emails to license activation links

Returns: A list of dictionary objects that are of the form:

```
{
  "user": user
```

(continues on next page)

(continued from previous page)

```
  "enrolled_in": {
      'name': course_name,
      'url': destination_url,
      'type': 'course',
      'start': course_start,
  },
  "dashboard_url": dashboard_url,
  "enterprise_customer_uuid": self.uuid,
  "admin_enrollment": admin_enrollment,
}
```

where user is one of

- 1: { 'first_name': name, 'username': user_name, 'email': email } (dict of User object)
- 2: { 'user_email' : user_email } (dict of PendingEnterpriseCustomerUser object)

`enterprise.utils.`**`strip_html_tags`**(*text*, *allowed_tags=None*)

Strip all tags from a string except those tags provided in *allowed_tags* parameter.

Parameters

- **text** (*str*) – string to strip html tags from
- **allowed_tags** (*list*) – allowed list of html tags

Returns: a string without html tags

`enterprise.utils.`**`track_enrollment`**(*pathway*, *user_id*, *course_run_id*, *url_path=None*)

Emit a track event for enterprise course enrollment.

`enterprise.utils.`**`track_enterprise_user_linked`**(*user_id*, *enterprise_customer_key*, *enterprise_customer_id*, *created_new_ent_user*)

Emit a track event when user is linked to an enterprise

`enterprise.utils.`**`track_event`**(*user_id*, *event_name*, *properties*)

Emit a track event to segment (and forwarded to GA) for some parts of the Enterprise workflows.

`enterprise.utils.`**`traverse_pagination`**(*response*, *client*, *api_url*)

Traverse a paginated API response.

Extracts and concatenates "results" (list of dict) returned by DRF-powered APIs.

Parameters

- **response** (*Dict*) – Current response dict from service API;
- **client** (*requests.Session*) – either the OAuthAPIClient (from edx_rest_api_client) or requests.Session object;
- **api_url** (*str*) – API endpoint URL to call.

Returns list of dict.

`enterprise.utils.`**`ungettext_min_max`**(*singular*, *plural*, *range_text*, *min_val*, *max_val*)

Return grammatically correct, translated text based off of a minimum and maximum value.

Example

min = 1, max = 1, singular = '{} hour required for this course', plural = '{} hours required for this course' output = '1 hour required for this course'

min = 2, max = 2, singular = '{} hour required for this course', plural = '{} hours required for this course' output = '2 hours required for this course'

min = 2, max = 4, range_text = '{}-{} hours required for this course' output = '2-4 hours required for this course'

min = None, max = 2, plural = '{} hours required for this course' output = '2 hours required for this course'

Expects `range_text` to already have a translation function called on it.

Returns `None` if both of the input values are `None`. `singular` formatted if both are equal or one of the inputs, but not both, are `None`, and the value is 1. `plural` formatted if both are equal or one of its inputs, but not both, are `None`, and the value is > 1. `range_text` formatted if min != max and both are valid values.

`enterprise.utils.`**`unset_enterprise_learner_language`**(*enterprise_customer_user*)

Unset the language preference of the given enterprise learners.

Parameters **`enterprise_customer_user`** (`EnterpriseCustomerUser`) – Instance of the enterprise customer user.

`enterprise.utils.`**`unset_language_of_all_enterprise_learners`**(*enterprise_customer*)

Unset the language preference of all the learners belonging to the given enterprise customer.

Parameters **`enterprise_customer`** (`EnterpriseCustomer`) – Instance of the enterprise customer.

`enterprise.utils.`**`update_query_parameters`**(*url*, *query_parameters*)

Return url with updated query parameters.

Parameters

- **`url`** (*`str`*) – Original url whose query parameters need to be updated.
- **`query_parameters`** (*`dict`*) – A dictionary containing query parameters to be added to course selection url.

Returns

slug identifier for the identity provider that can be used for identity verification of users associated the enterprise customer of the given user.

Return type (slug)

`enterprise.utils.`**`validate_course_exists_for_enterprise`**(*enterprise_customer*, *course_id*, ***kwargs*)

Validates that a specified course id exists within the LMS and within the enterprise_customer's catalog(s).

Parameters

- **`enterprise_customer`** (`EnterpriseCustomer`) – Instance of the enterprise customer.
- **`course_id`** (*`string`*) – The unique identifier of a course.
- **`kwargs`** – Should contain enrollment_client if it's already been instantiated and is passed in.

`enterprise.utils.`**`validate_email_to_link`**(*email*, *enterprise_customer*, *raw_email=None*, *message_template=None*, *raise_exception=True*)

Validate email to be linked to Enterprise Customer.

Performs two checks:

- Checks that email is valid
- Checks that it is not already linked to the provided Enterprise Customer

Parameters

- **email** (*str*) – user email to link
- **enterprise_customer** (EnterpriseCustomer) – the enterprise customer to link the email to
- **raw_email** (*str*) – raw value as it was passed by user - used in error message.
- **message_template** (*str*) – Validation error template string.
- **raise_exception** (*bool*) – whether to raise an exception when an email is invalidated

Raises **ValidationError** – if email is invalid or already linked to Enterprise Customer.

Returns Whether or not there is an existing record with the same email address.

Return type bool

8.1.18 enterprise.validators module

Database models field validators.

enterprise.validators.**get_app_config**()
:return Application configuration.

enterprise.validators.**validate_content_filter_fields**(*content_filter*)
Validate particular fields (if present) passed in through content_filter are certain types.

enterprise.validators.**validate_hex_color**(*value*)
Validate value is suitable for a color hex value.

enterprise.validators.**validate_image_extension**(*value*)
Validate that a particular image extension.

enterprise.validators.**validate_image_size**(*image*)
Validate that a particular image size.

8.1.19 enterprise.views module

User-facing views for the Enterprise app.

class enterprise.views.**CourseEnrollmentView**(***kwargs*)
Bases: *enterprise.views.NonAtomicView*

Enterprise landing page view.

This view will display the course mode selection with related enterprise information.

PACING_FORMAT = {'instructor_paced': 'Instructor-Paced', 'self_paced': 'Self-Paced'}

get(*request*, *enterprise_uuid*, *course_id*)
Show course track selection page for the enterprise.

Based on *enterprise_uuid* in URL, the view will decide which enterprise customer's course enrollment page is to use.

Unauthenticated learners will be redirected to enterprise-linked SSO.

A 404 will be raised if any of the following conditions are met:

- No enterprise customer uuid kwarg *enterprise_uuid* in request.
- **No enterprise customer found against the enterprise customer** uuid *enterprise_uuid* in the request kwargs.
- No course is found in database against the provided *course_id*.

get_available_course_modes(*request*, *course_run_id*, *enterprise_catalog*)
Return the available course modes for the course run.

The provided EnterpriseCustomerCatalog is used to filter and order the course modes returned using the EnterpriseCustomerCatalog's field "enabled_course_modes".

get_base_details(*request*, *enterprise_uuid*, *course_run_id*)
Retrieve fundamental details used by both POST and GET versions of this view.

Specifically, take an EnterpriseCustomer UUID and a course run ID, and transform those into an actual EnterpriseCustomer, a set of details about the course, and a list of the available course modes for that course run.

get_enterprise_course_enrollment_page(*request*, *enterprise_customer*, *course*, *course_run*, *course_modes*, *enterprise_course_enrollment*, *data_sharing_consent*)
Render enterprise-specific course track selection page.

post(*request*, *enterprise_uuid*, *course_id*)
Process a submitted track selection form for the enterprise.

set_final_prices(*modes*, *request*)
Set the final discounted price on each premium mode.

class enterprise.views.**EnterpriseLoginView**(***kwargs*)
Bases: `django.views.generic.edit.FormView`

Allow an enterprise learner to login by enterprise customer's slug login.

form_class
alias of `enterprise.forms.EnterpriseLoginForm`

form_invalid(*form*)
If the form is invalid then return the errors.

form_valid(*form*)
If the form is valid, redirect to the third party auth login page.

get_context_data(***kwargs*)
Return the context data needed to render the view.

template_name = 'enterprise/enterprise_customer_login_page.html'

class enterprise.views.**EnterpriseProxyLoginView**(***kwargs*)
Bases: `django.views.generic.base.View`

Allows an enterprise learner to login via existing flow from the learner portal.

get(*request*)
Redirects a learner through login with their enterprise's third party auth if it uses tpa.

class enterprise.views.**EnterpriseSelectionView**(***kwargs*)
Bases: `django.views.generic.edit.FormView`

Allow an enterprise learner to activate one of learner's linked enterprises.

dispatch(*request*, **args*, ***kwargs*)

form_class
alias of `enterprise.forms.EnterpriseSelectionForm`

form_invalid(*form*)
If the form is invalid then return the errors.

form_valid(*form*)
If the form is valid, activate the selected enterprise.

get_context_data(***kwargs*)
Return the context data needed to render the view.

get_initial()
Return the initial data to use for forms on this view.

template_name = 'enterprise/enterprise_customer_select_form.html'

class enterprise.views.**FailedEnrollmentReason**(*enrollment_client_error*, *failure_reason_message*)
Bases: `tuple`

enrollment_client_error
Alias for field number 0

failure_reason_message
Alias for field number 1

class enterprise.views.**GrantDataSharingPermissions**(***kwargs*)
Bases: `django.views.generic.base.View`

Provide a form and form handler for data sharing consent.

View handles the case in which we get to the "verify consent" step, but consent hasn't yet been provided - this view contains a GET view that provides a form for consent to be provided, and a POST view that consumes said form.

course_or_program_exist(*course_id*, *program_uuid*)
Return whether the input course or program exist.

static create_enterprise_course_enrollment(*request*, *enterprise_customer*, *course_id*, *license_uuid=None*)
Create EnterpriseCustomerUser and EnterpriseCourseEnrollment record if not already exists.

get(*request*)
Render a form to collect user input about data sharing consent.

get_context_from_db(*consent_page*, *platform_name*, *item*, *context*)
Make set of variables(populated from db) that will be used in data sharing consent page.

get_course_or_program_context(*enterprise_customer*, *course_id=None*, *program_uuid=None*)
Return a dict having course or program specific keys for data sharing consent page.

get_default_context(*enterprise_customer*, *platform_name*)
Get the set of variables that will populate the template by default.

get_page_language_context_data(*course_id*, *enterprise_customer*, *success_url*, *failure_url*, *license_uuid*, *request*, *platform_name*)
Return a dict of data for the language on the page.

is_course_run_id(*course_id*)
Returns True if the course_id is in the correct format of a course_run_id, false otherwise.

Parameters **course_id** (`str`) – The course_key or course run id

Returns True or False

Return type (Boolean)

post(*request*)
Process the above form.

preview_mode = False

class enterprise.views.**HandleConsentEnrollment**(***kwargs*)
Bases: `django.views.generic.base.View`

Handle enterprise course enrollment at providing data sharing consent.

View handles the case for enterprise course enrollment after successful consent.

get(*request*, *enterprise_uuid*, *course_id*)
Handle the enrollment of enterprise learner in the provided course.

Based on *enterprise_uuid* in URL, the view will decide which enterprise customer's course enrollment record should be created.

Depending on the value of query parameter *course_mode* then learner will be either redirected to LMS dashboard for audit modes or redirected to ecommerce basket flow for payment of premium modes.

class enterprise.views.**NonAtomicView**(***kwargs*)
Bases: `django.views.generic.base.View`

A base class view for views that disable atomicity in requests.

dispatch(*request*, **args*, ***kwargs*)
Disable atomicity for the view.

Since we have settings.ATOMIC_REQUESTS enabled, Django wraps all view functions in an atomic transaction, so they can be rolled back if anything fails.

However, we need to be able to save data in the middle of get/post(), so that it's available for calls to external APIs. To allow this, we need to disable atomicity at the top dispatch level.

class enterprise.views.**ProgramEnrollmentView**(***kwargs*)
Bases: *`enterprise.views.NonAtomicView`*

Enterprise Program Enrollment landing page view.

This view will display information pertaining to program enrollment, including the Enterprise offering the program, its (reduced) price, the courses within it, and whether one is already enrolled in them, and other several pieces of Enterprise context.

static extend_course(*course*, *enterprise_customer*, *request*)
Extend a course with more details needed for the program landing page.

In particular, we add the following:

- *course_image_uri*
- *course_title*
- *course_level_type*
- *course_short_description*
- *course_full_description*
- *course_effort*
- *expected_learning_items*

- *staff*

get(*request*, *enterprise_uuid*, *program_uuid*)

Show Program Landing page for the Enterprise's Program.

Render the Enterprise's Program Enrollment page for a specific program. The Enterprise and Program are both selected by their respective UUIDs.

Unauthenticated learners will be redirected to enterprise-linked SSO.

A 404 will be raised if any of the following conditions are met:

- No enterprise customer UUID query parameter `enterprise_uuid` found in request.
- **No enterprise customer found against the enterprise customer** uuid `enterprise_uuid` in the request kwargs.
- **No Program can be found given `program_uuid` either at all or associated with** the Enterprise..

get_enterprise_program_enrollment_page(*request*, *enterprise_customer*, *program_details*)

Render Enterprise-specific program enrollment page.

get_program_details(*request*, *program_uuid*, *enterprise_customer*)

Retrieve fundamental details used by both POST and GET versions of this view.

Specifically:

- Take the program UUID and get specific details about the program.
- Determine whether the learner is enrolled in the program.
- Determine whether the learner is certificate eligible for the program.

post(*request*, *enterprise_uuid*, *program_uuid*)

Process a submitted track selection form for the enterprise.

class enterprise.views.**RouterView**(***kwargs*)

Bases: *enterprise.views.NonAtomicView*

A router or gateway view for managing Enterprise workflows.

COURSE_ENROLLMENT_VIEW_URL = '/enterprise/{}/course/{}/enroll/'

HANDLE_CONSENT_ENROLLMENT_VIEW_URL = '/enterprise/handle_consent_enrollment/{}/course/{}/'

PROGRAM_ENROLLMENT_VIEW_URL = '/enterprise/{}/program/{}/enroll/'

VIEWS = {'/enterprise/handle_consent_enrollment/{}/course/{}/': <class 'enterprise.views.HandleConsentEnrollment'>, '/enterprise/{}/course/{}/enroll/': <class 'enterprise.views.CourseEnrollmentView'>, '/enterprise/{}/program/{}/enroll/': <class 'enterprise.views.ProgramEnrollmentView'>}

eligible_for_direct_audit_enrollment(*request*, *enterprise_customer*, *resource_id*, *course_key=None*)

Return whether a request is eligible for direct audit enrollment for a particular enterprise customer.

'resource_id' can be either course_run_id or program_uuid. We check for the following criteria: - The *audit* query parameter. - The user's being routed to the course enrollment landing page. - The customer's catalog contains the course in question. - The audit track is an available mode for the course.

get(*request*, **args*, ***kwargs*)

Run some custom GET logic for Enterprise workflows before routing the user through existing views.

In particular, before routing to existing views:

- If the requested resource is a course, find the current course run for that course, and make that course run the requested resource instead.
- Look to see whether a request is eligible for direct audit enrollment, and if so, directly enroll the user.

static get_course_run_id(*user*, *enterprise_customer*, *course_key*)

User is requesting a course, we need to translate that into the current course run.

Parameters

- **user** –
- **enterprise_customer** –
- **course_key** –

Returns course_run_id

static get_path_variables(***kwargs*)

Get the base variables for any view to route to.

Currently gets: - *enterprise_uuid* - the UUID of the enterprise customer. - *course_run_id* - the ID of the course, if applicable. - *program_uuid* - the UUID of the program, if applicable.

post(*request*, **args*, ***kwargs*)

Run some custom POST logic for Enterprise workflows before routing the user through existing views.

redirect(*request*, **args*, ***kwargs*)

Redirects to the appropriate view depending on where the user came from.

exception enterprise.views.**VerifiedModeUnavailableException**

Bases: Exception

Exception that indicates the verified enrollment mode has expired or is otherwise unavaible for a course run.

enterprise.views.**add_reason_to_failure_url**(*base_failure_url*, *failure_reason*)

Adds a query param to the given base_failure_url indicating why an enrollment has failed.

enterprise.views.**get_global_context**(*request*, *enterprise_customer=None*)

Get the set of variables that are needed by default across views.

enterprise.views.**get_price_text**(*price*, *request*)

Return the localized converted price as string (ex. '$150 USD').

If the local_currency switch is enabled and the users location has been determined this will convert the given price based on conversion rate from the Catalog service and return a localized string

enterprise.views.**get_safe_redirect_url**(*url*, *requires_https=None*)

Ensure that the URL which is going to be used for redirection exists in whitelist.

enterprise.views.**render_page_with_error_code_message**(*request*, *context_data*, *error_code*, *exception=None*, ***kwargs*)

Return a 404 page with specified error_code after logging error and adding message to django messages.

enterprise.views.**should_upgrade_to_licensed_enrollment**(*consent_record*, *license_uuid*)

In the event that a learner did enroll into audit from B2C and they consented on the data sharing consent page, we want to upgrade their audit enrollment into verified when they view the course in the learner portal and hit the DSC GET endpoint again.

enterprise.views.**verify_edx_resources**()

Ensure that all necessary resources to render the view are present.

8.1.20 Module contents

Your project description goes here.

CHAPTER

NINE

CHANGE LOG

9.1 Unreleased

None

9.2 [3.56.10]

fix: adhering to urljoin patterns in integrated channels API views

9.3 [3.56.9]

fix: properly truncate payload to resolve missing CSOD deletes

9.4 [3.56.8]

feat: add debug logging to investigate missing CSOD deletes

9.5 [3.56.7]

feat: add debug logging to investigate missing CSOD deletes

9.6 [3.56.6]

feat: expand utility of CSOD deleted_at reset job

9.7 [3.56.5]

fix: properly pass SAP client status back to content transmission records

9.8 [3.56.4]

fix: open redirect url whitelisting for data sharing conseent and change enterprise page

9.9 [3.56.3]

fix: replace id with uuid in branding logo file path

9.10 [3.56.2]

fix: refactor the way we send cornerstone content metadata deletes

9.11 [3.56.1]

fix: accounting for integrated Canvas instances that have no root account Ids.

9.12 [3.56.0]

feat: refactor content metadata jobs to save api call status

9.13 [3.55.3]

fix: accurately selecting content key values when filtering for existing content metadata transmission audits.

9.14 [3.55.2]

fix: integrated channels properly handling customers with multiple catalogs that have overlapping content.

9.15 [3.55.1]

fix: properly removing update transmission payloads from SAP transmissions before saving completed records.

9.16 [3.55.0]

feat: add *enable_executive_education_2U_fulfillment* to *EnterpriseCustomer*

9.17 [3.54.2]

fix: follow-on to cornerstone learner records foreign keys job

9.18 [3.54.1]

fix: create cornerstone learner audit records with new foreign keys

9.19 [3.54.0]

feat: Remove progress, progress_v2 option from reporting configs and move all v1, v2 to v3.

9.20 [3.53.4]

fix: update course run selection logic for SAP content exporter

9.21 [3.53.3]

feat: mark Cornerstone session token fields read-only in admin

9.22 [3.53.2]

feat: update data sharing consent request language

9.23 [3.53.1]

feat: Django Admin tweaks for integrations

9.24 [3.53.0]

feat: Added management command for weekly nudge to dormant enterprise learners

9.25 [3.52.0]

feat: add *enable_portal_learner_credit_management_screen* to *EnterpriseCustomer*

9.26 [3.51.1]

feat: basic integrated channels task concurrency control

9.27 [3.51.0]

feat: Added command for monthly impact report for enterprise administrators

9.28 [3.50.0]

feat: add *enable_learner_portal_offers* to *EnterpriseCustomer*

9.29 [3.49.10]

fix: append backslash to ecommerce url and change *get* to *get_or_create* in *fulfill_pending_course_enrollments*

9.30 [3.49.9]

feat: add source query param in data sharing consent url

9.31 [3.49.8]

chore: add logs for dsc

9.32 [3.49.7]

fix: parsing improvements to incorrect learner completion completed-at dates

9.33 [3.49.6]

fix: incorrect learner completion completed-at dates

9.34 [3.49.5]

feat: add lms_user_id to serialized admin users

9.35 [3.49.4]

feat: add dry-run mode to integrated channels

9.36 [3.49.3]

fix: don't transmit schedule data to SAP if start or end date is empty

9.37 [3.49.2]

feat: improved channel logging

9.38 [3.49.1]

fix: add stricter validation on system wide role assignments

9.39 [3.49.0]

fix: Return None for context if a `SystemWideEnterpriseUserRoleAssignment` has no enterprise_customer and does not apply
We'll no longer fall back on granting context based on enterprise membership when there is no explicit context.

9.40 [3.48.0]

chore: add migration to remove is_active from role assignment model schema

9.41 [3.47.2]

chore: remove is_active field from role assignment model

9.42 [3.47.1]

chore: remove data-cleaning management commands. prepare for column-removal migration

9.43 [3.47.0]

temp: adding system wide role assignment field and management commands to clean data

9.44 [3.46.6]

fix: correctly handle multiple canvas and blackboard oauth configs

9.45 [3.46.5]

fix: degreed2 improperly tracking completion status

9.46 [3.46.4]

fix: Degreed2 estimated time to complete in hours

9.47 [3.46.3]

fix: update logic for parsing course price for SAP

9.48 [3.46.2]

fix: Degreed2 estimated time to complete in days

9.49 [3.46.1]

feat: admin view improvements

9.50 [3.46.0]

fix: modify signature of EmbargoApiClient.redirect_if_blocked Make this signature match and use the same signature that `openedx.core.djangoapps.embargo.api.redirect_if_blocked()` now uses.

9.51 [3.44.4]

fix: implement back-off and retry for degreed2 fix: drop tpa_hint param in redirects when no SSO

9.52 [3.44.3]

fix: Undoes revert of 3.44.0, while also ensuring that `SystemWideEnterpriseUserRoleAssignment.get_assignments()` can handle and respect any null values returned from `get_context()`.

9.53 [3.44.2]

- feat: configure django admin for degreed2 audit records
- fix: Moodle client should accept treat duplicate course id on create as a success

9.54 [3.44.1]

fix: no-op version bump (skipping 3.44.0) to account for a revert: https://github.com/openedx/edx-enterprise/pull/1534

9.55 [3.44.0]

fix: [REVERTED] override get_assignments() so that active enterprise uuids come first.

Overrides the SystemWideEnterpriseUserRoleAssignment.get_assignments() method to return a list of (role, context) assignments, where the first item in the list corresponds to the currently active enterprise for the user.

9.56 [3.43.1]

chore: replace enterprise customer drop-downs in django admin

9.57 [3.43.0]

feat: allow admins to remove learners from org

9.58 [3.42.5]

fix: improve guards on fk data backfill job

9.59 [3.42.4]

feat: updated logic for completions in integrated channels

9.60 [3.42.3]

feat: additional fk data backfill performance improvements

9.61 [3.42.2]

feat: speed up fk data backfill

9.62 [3.42.1]

feat: use new foreign keys on integrated channels audit models

9.63 [3.42.0]

feat: add admin_users to `EnterpriseCustomerSerializer`

9.64 [3.41.13]

fix: remove backfill managment command arguments

9.65 [3.41.12]

fix: Use enterprise customer uuid coming in request data

9.66 [3.41.11]

fix: Add unique_together constraint in SystemWideEnterpriseUserRoleAssignment

9.67 [3.41.10]

fix: Add management command to backfill missing audit record foreign keys.

9.68 [3.41.9]

fix: Squash SAP Success Factors migrations to remove reference to `PositiveIntegerField`.

9.69 [3.41.8]

fix: Alter *enterprise_course_enrollment_id* field from *PositiveIntegerField* to *IntegerField* in *BlackboardLearnerAssessmentDataTransmissionAudit* and *SapSuccessFactorsLearnerDataTransmissionAudit*. This change require to run migrations on mysql8.

9.70 [3.41.7]

fix: add foreign keys to integrated channels audit models

9.71 [3.41.6]

fix: making making degreed token base url optional

9.72 [3.41.5]

feat: add missing logging for grades api results in integrated channels exporter

9.73 [3.41.4]

feat: added html pages for oauth authorization success/failure

9.74 [3.41.3]

fix: allow null completed_timestamp field for integrated channels learner audit models

9.75 [3.41.2]

feat: setting customer identity provider config is_valid on first SSO login

9.76 [3.41.0]

feat: Allow partial_update on *EnterpriseCustomerViewSet*

9.77 [3.40.16]

fix: CSOD Learner Audit Django Admin Timeouts

9.78 [3.40.15]

fix: Use correct completions URL for Degreed2

9.79 [3.40.14]

fix: CornerstoneLearnerDataTransmissionAudit admin view timeout

9.80 [3.40.13]

fix: Degreed2 Missing Learner Data Audit Records

9.81 [3.40.12]

fix: Degreed2 Missing Learner Data Audit Records

9.82 [3.40.11]

feat: New integrated channels Blackboard api endpoint to fetch global config creds

9.83 [3.40.10]

feat: Add drafting functionality to save incorrect fields

9.84 [3.40.9]

feat: new integrated channels customer configs list view, new integrated channels config serializer *is_valid* field

9.85 [3.40.8]

feat: add enable_browse_and_request field to *EnterpriseCustomer*

9.86 [3.40.7]

fix: Broken Canvas oauth authorization url

9.87 [3.40.6]

feat: SAPSF content metadata transmission now also sends course schedule

9.88 [3.40.5]

feat: adding CornerstoneLearnerDataTransmissionAudit admin view feat: log if-modified-since + content metadata for CSOD

9.89 [3.40.4]

feat: support filtering by a list of user ids for *EnterpriseCustomerUserViewSet*

9.90 [3.40.3]

feat: show field show_course_price in SAPSF Django admin form

9.91 [3.40.2]

feat: override chunk size default to match channel capability

9.92 [3.40.1]

chore: squash migrations for blackboard and sap_success_factor apps.

9.93 [3.40.0]

fix: Alter *enterprise_course_enrollment_id* field from *PositiveIntegerField* to *IntegerField* in *BlackboardLearnerAssessmentDataTransmissionAudit* and *SapSuccessFactorsLearnerDataTransmissionAudit*. This change require to run migrations on mysql8.

9.94 [3.39.1]

fix: switching blackboard integrated channels from client based auth credentials to global creds

9.95 [3.39.0]

fix: ensure *active* field on EnterpriseCustomerUser objects are set to *False* appropriately feat: add management command to clean up *active* fields on EnterpriseCustomerUser objects

9.96 [3.38.7]

feat: customer configs draft saving that makes all variables optional feat: feat: Add field for Display name for LMS configs

[3.39.0] chore: dropped Django22, 30 and 31 support

9.97 [3.38.6]

feat: add created to enterprise course enrollment serializer fields

9.98 [3.38.5]

fix: update link_learners action to respond with error when payload is empty.

9.99 [3.38.4]

fix: bugfix for Cornerstone missing completion records

9.100 [3.38.3]

fix: more logging to debug missing completion records

9.101 [3.38.2]

fix: Django Admin bugfix

9.102 [3.38.1]

feat: New crud viewset for IC degreed2 configurations

9.103 [3.38.0]

feat: Adds toggle_universal_link endpoint

9.104 [3.37.0]

feat: Dependency upgrades

9.105 [3.36.13]

fix: check if instance is an iterable rather than a list in `EnterpriseCustomerUserReadOnlySerializer`

9.106 [3.36.12]

feat: add enterprise role assignments to `EnterpriseCustomerUserReadOnlySerializer`

9.107 [3.36.11]

fix: Integrated channels Degreed2 exporter now handles invalid start/end date in content metadata item

9.108 [3.36.10]

fix: add *basic_list* action to `EnterpriseCustomerInviteKeyViewSet` to return unpaginated set of invite keys.

9.109 [3.36.9]

feat: new oauth state for multi-lms-configuration

9.110 [3.36.8]

feat: allow more than 1 lms configuration per lms-kind

9.111 [3.36.7]

feat: update *enterprise_customer_invite_key* filter and serializer

9.112 [3.36.6]

feat: Show OAuth Auth link for Blackboard Admin

9.113 [3.36.5]

fix: add support for an `enterprise_customer_invite_key` UUID query parameter to be passed and handled by the `EnterpriseProxyLoginView`

9.114 [3.36.4]

feat: OAuth Auth link for Blackboard Admin

9.115 [3.36.3]

feat: Integrated channels, grade send logic only logs instead of raising when enterprise_customer_user record is inactive

9.116 [3.36.2]

feat: add is_active on enterprise customer invite key

9.117 [3.36.1]

feat: improved integrated channel log consistency

9.118 [3.36.0]

feat: added view to link learners from a enterprise customer key

9.119 [3.35.3]

fix: require expiry date on `EnterpriseCustomerInviteKey` model

9.120 [3.35.2]

feat: reformat integrated channels logging to be more splunk friendly

9.121 [3.35.1]

docs: Updating help_text for universal_link field on EnterpriseCustomer model

9.122 [3.35.0]

feat: Adding universal_link field to EnterpriseCustomer model

9.123 [3.34.2]

feat: add try catch block to skip unfound courses

9.124 [3.34.1]

feat: add enterprise customer invite key model and viewset

9.125 [3.34.0]

feat: New management command to revert enrollment data

9.126 [3.33.12]

feat: SAPSF integrated no longer considers grade change as a reason to retransmit completions.

9.127 [3.33.11]

feat: New management command to backfill end dates on Canvas

9.128 [3.33.10]

fix: incorrectly skipping completion transmissions

9.129 [3.33.9]

feat: allow filtering enterprise learners by enterprise uuid and enterprise role

9.130 [3.33.8]

fix: Moodle duplicate content metadata records detected

9.131 [3.33.7]

fix: allow for records to be saved for integrated channels' content across catalogs

9.132 [3.33.6]

fix: CSOD API session tokens bugfix

9.133 [3.33.5]

fix: CSOD API session tokens are now saved to the customer's configuration instead of individual transmission audits

9.134 [3.33.4]

feat: integrated channels only requests content metadata for courses that need updating

9.135 [3.33.3]

feat: Change Bulk Enrollment Assignment Logic for Pending learners

9.136 [3.33.2]

fix: no longer notify learners of already existing enrollments

9.137 [3.33.1]

fix: Rename model field from key to client_id: Degreed2

9.138 [3.33.0]

feat: New Integrated channel Degreed v2.

9.139 [3.32.0]

feat: Added management command to fix DSC records having spaces instead of +.

9.140 [3.31.1]

fix: pip-tools upgrade

9.141 [3.31.0]

feat: new integrated channels content metadata transmitter flow

9.142 [3.30.14]

fix: blackboard logging function was not returning desired string

9.143 [3.30.13]

fix: properly weight blackboard grades

9.144 [3.30.12]

- chore: update course enrollments through lms

9.145 [3.30.11]

- docs: added adr for zero state browsing with universal link

9.146 [3.30.10]

- fix: refactor moodle _post to use body params

9.147 [3.30.9]

- chore: Don't expire courses that have been modified after given date

9.148 [3.30.8]

- feat: Added a boolean in EnterpriseCustomer to specify whether labor market data should be available in learner portal

9.149 [3.30.7]

- update admin banner notification text field with following changes
- increase max_length from 255 to 512
- update help text
- use textarea widget in django admin

9.150 [3.30.6]

- maint: Integrated channels detection system of catalog changes needed is now disabled via override.

9.151 [3.30.5]

- fix: Integrated channels data transforming generates json serializable fields.

9.152 [3.30.4]

- fix: Blackboard integrated channel now correctly synchronizes the one-and-only valid refresh_token

9.153 [3.30.3]

- fix: content_filter in django admin was broken after the jsonfield upgrade, so this contains fix for that

9.154 [3.30.2]

- fix: switch is_revoked to True on LicensedEnterpriseCourseEnrollment after license expiration

9.155 [3.30.1]

- Allowing management commands to optionally run on inactive Integrated Channel configurations

9.156 [3.30.0]

- Switched back to `jsonfield` from `jsonfield2`

9.157 [3.29.0]

- Added api for fetching field choices from EnterpriseCustomerReportingConfiguration

9.158 [3.28.24]

- Integrated channels Canvas: now fills in Start/end dates in description, and uses Course participation type

9.159 [3.28.23]

- Fix cornerstone character limit bug with dict database table

9.160 [3.28.22]

- fix: Adding error handling for role assignment backfill management command

9.161 [3.28.21]

- bug: The exporter now properly handles instances when enterprise customer catalogs do no need updates.

9.162 [3.28.20]

- feat: Added enterprise_learner role assignment backfill management command

9.163 [3.28.19]

- fix: additional Moodle field changes

9.164 [3.28.18]

- Added customer config based learner data transmission feature flag

9.165 [3.28.17]

- Improve error logging in the Moodle integration

9.166 [3.28.16]

- Fix import error used by bulk enrollment in utils

9.167 [3.28.15]

- integrated channels: single learner assessment exporter logging is not helpful right now so improve it.

9.168 [3.28.14]

- logging improvement when calling integrated channels extract_integration_id

9.169 [3.28.13]

- fixes the way moodle queries for courses ENT-4806

9.170 [3.28.12]

- Integrated channels automatically fill in current date for audit completions if date not available.

9.171 [3.28.11]

- Create "enterprise_learner" role when `EnterpriseCustomerUser` records are re-linked.
- When `EnterpriseCustomerUser` records get deleted, also delete the "enterprise_admin" role specific to the relevant enterprise customer.

9.172 [3.28.10]

- Integrated channel transmitter completions routine now logs as error, any status codes greater than or equal to 400

9.173 [3.28.9]

- Include a `failure_reason=dsc_denied` to the DSC failure url when learner denies the DSC terms.

9.174 [3.28.8]

- SAP integrated channel django form gets missing idp id field

9.175 [3.28.7]

- Degreed integrated channel now uses idp_id explicitly when calling get_remote_id()

9.176 [3.28.6]

- SAP integrated channel now uses idp_id explicitly when calling get_remote_id()

9.177 [3.28.5]

- Fixed datetime issue in email_drip_for_missing_dsc_records.

9.178 [3.28.4]

- Integrated channels: audit track completion status now based on incomplete non-gated content

9.179 [3.28.3]

- Integrated channels: log response code and message if SAP post fails

9.180 [3.28.2]

- Add *progress_v3* report type for enterprise reporting.

9.181 [3.28.1]

- Inject a failure reason into the `failure_url` query params when a verified course mode is not available for DSC-based enrollments.

9.182 [3.28.0]

- Added support for Django 3.0, 3.1 and 3.2

9.183 [3.27.27]

- Adds enterprise catalog query title as an optional attribute to create/update post requests on the catalog service.

9.184 [3.27.26]

- Refactor data-sharing consent GET and POST handlers to not have too many statements, because readability matters.

9.185 [3.27.25]

- Blackboard Integrated channel oauth2 refresh token handling fixes.

9.186 [3.27.24]

- Adding a new EnterpriseEnrollmentSource to be used for bulk enrollment.

9.187 [3.27.23]

- Add logging of user id for troubleshooting in a couple of locations.
- Clean up pylint suppressions and rules using latest rules set by edx-lint.

9.188 [3.27.22]

- Prevent failures on integrated channels delete requests when courses are not found.

9.189 [3.27.21]

- Encode invalid course keys for CSOD customers

9.190 [3.27.20]

- Handle content_last_modified not provided by enterprise catalog

9.191 [3.27.19]

- Localize timezones on catalog modified min (not found) values

9.192 [3.27.18]

- Integrated channels util functions needed to base64 urlsafe encode/decode course keys for use with some LMS systems like Cornerstone.

9.193 [3.27.17]

- Integrated channels now checks and uses catalog modified times to determine if an update is needed before retrieving content metadata.

9.194 [3.27.16]

- Making bulk catalog query ID update params optional

9.195 [3.27.15]

- Added title field in `AdminNotification` table.

9.196 [3.27.14]

- Adding the ability to specify parameters in the bulk catalog query ID updated management command.

9.197 [3.27.13]

- Revert 'Start my course' links in bulk enrollment emails to courseware based links instead of learner portal.

9.198 [3.27.12]

- Prevent django admin deletions of catalog queries. Added management command to bulk update catalogs of their query IDs

9.199 [3.27.11]

- Avoid failure when an email send in the learners loop fails, for notify_enrolled_learners

9.200 [3.27.10]

- Use celery tasks for emails sent using EnterpriseCustomer's notify_enrolled_learners method

9.201 [3.27.9]

- Fix SAP Course Completion payload format again.

9.202 [3.27.8]

- Fix SAP Course Completion payload format.

9.203 [3.27.7]

- Replace EnrollmentApiClient calls from Bulk enrollment with a newly minted python api call (non-REST) from edx-platform

9.204 [3.27.6]

- Filter available IDPs for Enterprise Customers by new boolean flag on ProviderConfig model.

9.205 [3.27.5]

- Removing CSOD Integrated Channel from the list of supported channels for the content metadata transmission task.

9.206 [3.27.4]

- Add pagination handling to integrated channels Blackboard client

9.207 [3.27.3]

- Adds flag to SAP Success Factors customer configuration to switch SAP endpoints for learner completion calls.

9.208 [3.27.2]

- Ensure deletion and unlinking of a `EnterpriseCustomerUser` record only deletes the `enterprise_learner` system-wide role for that particular `EnterpriseCustomerUser`, as opposed to all `enterprise_learner` roles associated with the user.

9.209 [3.27.1]

- Updates bulk enrollment email template.

9.210 [3.27.0]

- Added enterprise uuid support in course enrollment. ERTE-5

9.211 [3.26.23]

- Fix the way that `page_size` is passed as a param to the `get_content_metadata` endpoint. Add a unit test for the `EnterpriseCatalogApiClient.get_content_metadata()` method, which was previously untested.

9.212 [3.26.22]

- Set the EnterpriseCatalogApiClient get_content_metadata request page_size parameter to 50; the enterprise-catalog service has a default page_size of 10. This change means that we'll make a smaller overall number of SELECTs against the enterprise-catalog database.

9.213 [3.26.21]

- Adds error handling and logging to the assignment deduplication management command.

9.214 [3.26.20]

- Updates requirements and style changes to match the latest Pylint.

9.215 [3.26.19]

- Updates to integrated channels catalogs to transmit help text.

9.216 [3.26.18]

- Overriding default chunk size for SAP and Canvas integrations.

9.217 [3.26.17]

- Adds Segment tracking for bulk enrollment method.

9.218 [3.26.16]

- Added history tables for EnterpriseCustomerUser and SystemWideEnterpriseUserRoleAssignment.

9.219 [3.26.15]

- Added management command to clean up duplicate transmitted assignments for the integrated channels.

9.220 [3.26.14]

- Fixed issue with API version in Tableau client.

9.221 [3.26.13]

- Fixed issue with CourseEnrollment receiver when learner has multiple enterprises.

9.222 [3.26.12]

- Canvas integrated channel now supports create_or_update pattern for courses. Detects/logs deleted courses.

9.223 [3.26.11]

- Removed `ENABLE_MULTIPLE_USER_ENTERPRISES_FEATURE` waffle switch

9.224 [3.26.10]

- Fix forward for parameter rename changing the signals API in 3.26.7

9.225 [3.26.9]

- Added support to use default idp in Enterprise slug login if there are multiple.

9.226 [3.26.8]

- added support for redirecting user to default IDP, in case multiple IDPs's attached

9.227 [3.26.7]

- developer-only facing updates to standardize LMS Integrated Channels logging.

9.228 [3.26.6]

- added an update api call to assign tableau user roles

9.229 [3.26.5]

- fix: Bypass slumber's getattr definition when requesting enrollments for usernames starting with '_' (because slumber will raise an AttributeError from getattr when requesting a resource that starts with '_').

9.230 [3.26.4]

- removed unnecessary call to ecom in bulk enrollment (process of assigning a license already accounts for this)

9.231 [3.26.3]

- added –skip-unlink param in unlink_enterprise_customer_learners command to just remove DSC records.

9.232 [3.26.2]

- Added logs for enterprise users created in tableau.

9.233 [3.26.1]

- Added check to configure reports only for Catalog over SFTP.

9.234 [3.26.0]

- Added support for admin scheduled banners that run from date x to date y.

9.235 [3.25.2]

- Log exception stack trace during DSC licensed-enrollment flow, so that we can look at log messages and understand what exactly is failing.

9.236 [3.25.1]

- bug fix, properly handle API response pagination from Canvas.

9.237 [3.25.0]

- added management command to unlink learners from their enterprise and deleting DSC and EnterpriseCourseEnrolment records.

9.238 [3.24.0]

- added `enable_compression` flag in EnterpriseCustomerReportConfiguration table.

9.239 [3.23.12]

- Database based template system for enrollment emails, including support for Admin and Self enroll modes. Admin mode for Bulk enrollment, existing enrollment emails still use the current template.

9.240 [3.23.11]

- Log more specific information about HTTP client errors that are caught when using the LMS enrollment API. Also send an exception event to the monitoring service when this happens, even though we handle the exception "gracefully".

9.241 [3.23.10]

- Send long dsc url in missing DSC email as individual params.

9.242 [3.23.9]

- Reduced the DSC url size to account for character limit in Segment event properties.

9.243 [3.23.8]

- Remove hardcoded admin permission constraints for ContentMetadataItemTransmission integrated channel model.

9.244 [3.23.7]

- Canvas integrated channel now 'concludes' course when sending deletion event, instead of 'delete'.

9.245 [3.23.6]

- Optimised handling of conditions defining the absence of a DSC.

9.246 [3.23.5]

- Added exception handling in consent missing email.

9.247 [3.23.4]

- Added a check for enterprise DSC configuration in missing DSC drip.

9.248 [3.23.3]

- Added a check for course access before sending Segment event for missing DSC.

9.249 [3.23.2]

- Added new field reply_to in enterprise customer where learner's reply to enterprise emails will be delivered.
- Removed migrations that have been merged into squashed migrations.

9.250 [3.23.1]

- Fix: filter out EnterpriseCourseEnrollments without corresponding CourseEnrollment records in learner portal view.

9.251 [3.23.0]

- Added support for `--enrollment-before` and `--no-commit` params in `email_drip_for_missing_dsc_records` command.

9.252 [3.22.16]

- Fixed Segment json string issue for DSC email drip

9.253 [3.22.15]

- Added additional Segment event properties for missing DSC drip email

9.254 [3.22.14]

- Fixed timezone issue in comparison of course start datetime

9.255 [3.22.13]

- Make enterprise customer uuid mandatory for *TableauAuthView*

9.256 [3.22.12]

- Change the verbose name and help text for the `enable_integrated_customer_learner_portal_search` field on the `EnterpriseCustomer` model.

9.257 [3.22.11]

- No longer call into the removed email_marketing platform djangoapp

9.258 [3.22.10]

- Use Braze for sending data sharing consent drop emails, add the DSC link inside the drip email.

9.259 [3.22.9]

- Expose enterprise catalog uuids associated with an Enterprise Customer in the `enterprise-customer` API endpoint.

9.260 [3.22.8]

- Add dashboard admin rbac role permission on tableau auth view so that only enterprise dashboard admins can access this view.
- Add support to generate tableau auth token based on incoming enterprise customer's uuid

9.261 [3.22.7]

- chore: upgrade edx-enterprise requirements

9.262 [3.22.6]

- Improves performance of enterprise role assignment admin page
- Deletes custom get_search_results() method, since `enterprise_customer__name` is now a viable search field
- Improves pagination by asking for an estimated row count from Mysql `INFORMATION_SCHEMA.TABLES`
- Turns 1 + N query into 1 query via proper use of `list_select_related`

9.263 [3.22.5]

- Fix: no longer stringifying *None* values passed to enterprise catalog creations calls

9.264 [3.22.4]

- Fix: learner_data exporter bug fix and refactor for cleaner enrollment filtering

9.265 [3.22.3]

- Feature: including EnterpriseCatalogQuery UUID field in request payload to enterprise-catalog on Enterprise-Catalog updates

9.266 [3.22.2]

- Feature: new UUID field on EnterpriseCatalogQuery model (and update to all existing query objects)

9.267 [3.22.1]

- Refactor: integrated channels learner exporter replace course api client

9.268 [3.22.0]

- Added a management command to send emails to learners with missing DSC

9.269 [3.21.4]

- allow searching of enterprise customer records with hyphenated uuid
- add typeahead search dropdown to imporve enterprise customer search on enterprise reporting configuration

9.270 [3.21.3]

- When a learner is linked from manage learners page, in-activate learner's other enterprises

9.271 [3.21.2]

- Added support of multiple identity_providers in enterprise.models.get_remote_id.

9.272 [3.21.1]

- Added multiple identity_providers in EnterpriseCustomerApi

9.273 [3.21.0]

- Added the ability to link/unlink enterprise customer catalogs with enterprise reporting configuration via its API endpoint.

9.274 [3.20.5]

- Integrated channels learner_data module refactored to avoid making some LMS REST API calls

9.275 [3.20.4]

- Refactored code in *proxied_get()* to clean up duplicate logic.

9.276 [3.20.3]

- Removing unused and out of date endpoints for Bulk Enrollment

9.277 [3.20.2]

- Allow licensed audit enrollment to have a path to upgrade into verified

9.278 [3.20.1]

- update edx-rbac to 1.4.2, plus a bunch of other version bumps.

9.279 [3.20.0]

- feat: add support for enterprise admins to create pending enterprise users

9.280 [3.19.0]

- feat: add support for creating multiple pending enterprise users

9.281 [3.18.7]

- Refactored bulk enrollment serializer and bug fixes to the bulk enrollment endpoint.

9.282 [3.18.6]

- fix: The update_role_assignments_with_customers command no longer updates records. It only creates new records, which helps de-risk the operation.

9.283 [3.18.5]

- fix: do not include unpublished courses when enrollment link resolves course_runs

9.284 [3.18.4]

- fix: The update_role_assignments_with_customers command no longer deletes open assignments. Allowing it to do so left us prone to error when an explicit enterprise_customer_uuid arg is provided. We should modify this command in the future to perform deletions of open assignments as its only action, and it should only be invoked this way after we have verified that all backfilled enterprise_customer fields on the assignments have been set correctly.

9.285 [3.18.3]

- Adds the catalog admin role to `roles_api.roles_by_name()`.

9.286 [3.18.2]

- Removes course mode as a required parameter to the bulk subscription enrollment endpoint.

9.287 [3.18.1]

- Adds bulk enterprise learner in bulk courses enrollment endpoint with pending user support.

9.288 [3.18.0]

- Adds a management command to update all `SystemWideEnterpriseUserRoleAssignment` records in a way that makes them more explicitly defined.

9.289 [3.17.47]

- Bug fix to remove a deprecated parameter that was causing bulk enrollments to fail.

9.290 [3.17.46]

- Made help text of sender_alias more generic.

9.291 [3.17.45]

- Fix bulk enrollment endpoint to process email_csv and email as well

9.292 [3.17.44]

- Replaced an LMS Enrollment API call with direct call the DB to avoid LMS rate limiting during integrated channels bulk jobs.

9.293 [3.17.43]

- Updated the default IDP priority of enterprises for social auth.

9.294 [3.17.42]

- Change canvas_course_id to BigInteger: Integrated Channels

9.295 [3.17.41]

- Upgrade django-ipware to version 3.0.2

9.296 [3.17.40]

- Read CSV files using *utf-8-sig* encoding to handle Byte Order Mark

9.297 [3.17.39]

- Rename *Owners* field to *Partners* for Cornerstone Integration

[3.17.38]

- Omitting assessment level reporting from integrated Canvas learners final grade to not have redundant reported points between final grades and subsection grades.

9.298 [3.17.37]

- Refactor to only create an `EnterpriseCourseEnrollment` if we successfully create/update a `CourseEnrollment` record

9.299 [3.17.36]

- Properly filtering integrated channels that support assessment level reporting.

9.300 [3.17.35]

- Map "estimated_hours" to "credit_hours" in addition to "total_hours" in SAP.

9.301 [3.17.34]

- Removing temporary logs from integrated channels.

9.302 [3.17.33]

- Enable manually adding learners to multiple enterprises

9.303 [3.17.32]

- Adding the logic to select default provider in case an enterprise has multiple identity providers attached.

9.304 [3.17.31]

- Change moodle course title in exporter, to include edX text.

9.305 [3.17.30]

- Investigatory logging to track down Integrated Channels transmission issues.

9.306 [3.17.29]

- Prevent NoneType string concatenation when handling multiple enterprises logistration without redirects.

9.307 [3.17.28]

- Adds default field in enterprise customer identity provider table to select default IDP if there are more than one IDPs attached with enterprise.

9.308 [3.17.27]

- Adding Logging to single learner assessment level reporting task.

9.309 [3.17.26]

- Updating docs to reflect method behaviors.

9.310 [3.17.25]

- Making failed SAP user remote ID retrievals log relevant context data.

9.311 [3.17.24]

- Making sure Canvas Integrated Channel properly url encodes user identifier fields.

9.312 [3.17.23]

- Fixing assessment level reporting audit retrieval.

9.313 [3.17.22]

- Adds content metadata item transmission table to Django Admin.

9.314 [3.17.21]

- Introduce and use a `roles_api` module and use the roles API in signal receivers that need to create or delete role assignments.
- For created or updated learner and admin enterprise users, associate their user-role with the `enterprise_customer` to which that user is linked.
- Install django-cache-memoize.

9.315 [3.17.20]

- Adds better exception handling to the SAP integrated channels.
- Adds better logging to the base transmission process in the integrated channels.

9.316 [3.17.19]

- Removes the sync_enterprise_catalog_query boolean field from the EnterpriseCustomerCatalog model.
- Adds migration to remove the sync_enterprise_catalog_query boolean field.

9.317 [3.17.18]

- Removes all references to the sync_enterprise_catalog_query boolean field from the EnterpriseCustomerCatalog model.
- Updates all conditional use of the sync_enterprise_catalog_query field to be True.
- A second PR will follow to remove the model field and perform the db migration (blue/green deployment safe).

9.318 [3.17.17]

- Added a catch all exception block to ensure login flow is not interrupted by analytics user sync.

9.319 [3.17.16]

- Include course mode for the user's `student.CourseEnrollment` in the `EnterpriseCourseEnrollmentSerializer`.

9.320 [3.17.15]

- In `SystemWideEnterpriseUserRoleAssignment`, Use either `applies_to_all_contexts` or `enterprise_customer` if they are True or non-null, respectively, in determining the result of `get_context()`, but continue to return list of all linked enterprise customer UUIDs if not, (which is the current behavior). This is a small step on our journey to explicitly defining user-role assignments.

9.321 [3.17.14]

- On the `SystemWideEnterpriseUserRoleAssignment` model, adds an `enterprise_customer` FK (nullable) and an `applies_to_all` boolean field (defaults to False) that indicates if the user has wildcard permissions.
- Updates the admin to show the "effective" customer in the detail view, and the explicit value in the list view. The effective value is the deprecated way we currently determine role assignment - by implicitly assigning the role on every customer to which the user is linked.
- In the detail view/form, the "Enterprise customer" dropdown contains only customers to which the user is currently linked.

9.322 [3.17.13]

- added check to make sure enterprise user can only use linked IdP with their enterprise customer.

9.323 [3.17.12]

- Conditionally allows the deletion of individual `EnterpriseCourseEnrollment` and related `LicensedEnterpriseCourseEnrollment` records via the Django Admin site, so that site admins can manually delete enterprise enrollments that were created in error. This is only allowed if a Django settings feature flag is set to `True`.

9.324 [3.17.11]

- Apply edx-rbac migration to add `applies_to_all_contexts` field to `SystemWideEnterpriseUserRoleAssignment`.
- Added endpoints for Cornerstone integrated channel.

9.325 [3.17.10]

- added home page logo for EnterpriseSelectionView and EnterpriseLoginView

9.326 [3.17.9]

- Fix deprecation warning: `third_party_auth` should be imported as `common.djangoapps.third_party_auth`.

9.327 [3.17.8]

- Added new API endpoints for Degreed integrated channel.

9.328 [3.17.7]

- Added new field `sender_alias` in enterprise customer which will be used in emails except of default alias.

9.329 [3.17.6]

- Non-effectual code cleanup / refactor to remove some final pieces of duplication (canvas, blackboard).

9.330 [3.17.5]

- Ensure enterprise course enrollments return valid course run statuses such that when a learner earns a passing certificate, the `enterprise_course_enrollments` API endpoint deems the course is complete even though the course itself may not have ended yet per the configured dates.

9.331 [3.17.4]

- Add some info to the `EnterpriseCourseEnrollment` docstring, add `is_active` property to same.

9.332 [3.17.3]

- Fixed unnessary integrated channel signal transmission on course completion to inactive customers by adding guard condition.

9.333 [3.17.2]

- Stop listening for `student.CourseEnrollment` unenrollment signal, as introduced in 3.17.0

9.334 [3.17.1]

- Add management command to process expired subscriptions and field on subscriptions to persist that the subscription expiration has been processed

9.335 [3.17.0]

- Listen for `student.CourseEnrollment` unenrollment signal and delete associated `EnterpriseCourseEnrollment` record if one exists (we will have a historical record of the deletion).

9.336 [3.16.11]

- Retrieve `EnterpriseCustomerUser` by both user_id and enterprise_customer to handle users who are pending for more than 1 enterprise.

9.337 [3.16.10]

- Forcing embedded enrollment links within integrated Blackboard courses to open new windows to avoid security alert prompt.

9.338 [3.16.9]

- Upgrade celery to 5.0.4

9.339 [3.16.8]

- Added ClientError exception handling for SAPSuccessFactorsAPIClient.

9.340 [3.16.7]

- Modify the learner portal enterprise_course_enrollments endpoint to include an `is_enrollment_active` key that indicates the status of the enterprise enrollment's related `student.CourseEnrollment`. Allow the endpoint to optionally accept an ``?is_active` query param, so that clients may request only active enrollments from it.

9.341 [3.16.6]

- Improved error handling for SAP Success Factors OAuth2 response.

9.342 [3.16.5]

- Refactoring title content metadata in integrated course creation within the Blackboard integrated channel.

9.343 [3.16.4]

- Add SuccessFactors Customer Configuration API endpoint.

9.344 [3.16.3]

- Update unique constraints for pending Enterprise learners/admins to support users who may be pending for more than 1 Enterprise.
- Fix `handle_user_post_save` to account for the potential of being a pending learner/admin for more than 1 Enterprise.

9.345 [3.16.2]

- Refactor `handle_user_post_save` to be responsible for linking PendingEnterpriseCustomerUser records and granting admin permissions.

9.346 [3.16.1]

- Adding backend support for admin portal Blackboard configuration.

9.347 [3.16.0]

- Added the ability to enable multiple Identity Providers for a single enterprise customer.

9.348 [3.15.0]

- Converted relation between enterprise customer and identity provider to a one-to-many.

9.349 [3.14.1]

- Adds new API for Canvas LMS configurations.

9.350 [3.14.0]

- Rebranding update: Change fonts and colors, change mobile layout

9.351 [3.13.12]

- Adding decorators to missed integrated channel tasks.

9.352 [3.13.11]

- Add new API for external LMS configurations.

9.353 [3.13.10]

- Use logo from `get_platform_logo_url` in the legacy Django templates

9.354 [3.13.9]

- Adding Blackboard support for assessment level reporting in the integrated channels.

9.355 [3.13.8]

- Bug fix with course key lookup in the Canvas assessment level grade reporting flow.

9.356 [3.13.7]

- Rebranding update: move to more robust `get_platform_logo_url` and update default branding colors.

9.357 [3.13.6]

- Add log for enterprise enrollment page.

9.358 [3.13.5]

- Fixed deprecation warnings related with drf methods (detail_route, list_route).

9.359 [3.13.4]

- Empty sequence bugfix in catalog api.

9.360 [3.13.3]

- Course end date bugfix.

9.361 [3.13.2]

- Add course end date to course level metadata.

9.362 [3.13.1]

- Base implementation of assessment level reporting for Integrated Channels.

9.363 [3.13.0]

- Use full paths for edx-platform/common/djangoapps imports, as described in edx-platform ADR #7.

9.364 [3.12.4]

- Fix silent exception in catalog api call.

9.365 [3.12.3]

- Add code_owner custom attribute for celery tasks.

9.366 [3.12.2]

- Refresh catalog metadata on create and update

9.367 [3.12.1]

- added support for grade, completion and course_structure type reports in enterprise report configurations. Added validation to allow these reports for Pearson enterprises only.

9.368 [3.12.0]

- Support uploading a `course_id` column in the "Manage Learners" CSV bulk upload to allow manual enrollments in multiple courses at once.

9.369 [3.11.1]

- Fixes the issue where user preference value can not be null.

9.370 [3.11.0]

- Added spanish translations for data sharing consent page.

9.371 [3.10.5]

- Update Moodle integration to single transmission to handle responses properly.

9.372 [3.10.4]

- Remove hyphens from enterprise_customer_uuid for admin user creation and tableau authentication.

9.373 [3.10.3]

- Fix timout on update.

9.374 [3.10.2]

- Updated the logic to clear enterprise learner language in a way that db lock does not happen.

9.375 [3.10.1]

- change username with enterprise_customer_uuid for tableau trusted authentication and tableau user creation.

9.376 [3.10.0]

- Tests only: upgrade to pytest 6+ and factoryboy 3+ to bring up to date with edx-platform.

9.377 [3.9.13]

- Adding Blackboard customization to integrated channel content metadata creation.

9.378 [3.9.12]

- change username with user_id for tableau trusted authentication and tableau user creation.

9.379 [3.9.11]

- add logs to know if data sharing consent is failing because catalog does not contain the course

9.380 [3.9.10]

- added POST enterprise-customer/<uuid>/enterprise_learner endpoint to mimic Manage Learners admin form functionality

9.381 [3.9.9]

- upgrade version to create new release on pypi.

9.382 [3.9.8]

- added error_codes in the logging/error messages for the CourseEnrollmentView for better debugging capability.

9.383 [3.9.7]

- Unset learners language so that default_language from enterprise customer may take effect.

9.384 [3.9.6]

- Fix DSC tests to verify enrolling a learner with a license_uuid

9.385 [3.9.5]

- ENT-2450: Add action to kick off jobs to refresh enterprise catalogs so changes will be immediately visible

9.386 [3.9.4]

- Style/UX changes for Moodle integration.

9.387 [3.9.3]

- Adding integrated course customization for Blackboard courses.

9.388 [3.9.2]

- Re-add check for license uuid when enrolling learners into a course

9.389 [3.9.1]

- Added the EnterpriseAnalyticsUser model and tableau integration functions.

9.390 [3.9.0]

- Enable enterprise to have a default language configuration for its learners.

9.391 [3.8.43]

- ENT-3557: Improve blackboard view logging to better report root cause of auth failure.

9.392 [3.8.42]

- ENT-3460: Adding properties to safely use branding config.

9.393 [3.8.41]

- Embedded enterprise in the username was removed for tableau trusted authentication.

9.394 [3.8.40]

- Bug fix: SAML stripping for unlinking was not properly removing saml prefix.

9.395 [3.8.39]

- Blackboard client update/delete and unit tests.

9.396 [3.8.38]

- Reverting changes to EnterpriseCustomerBrandingConfig.

9.397 [3.8.37]

- Using python properties for EnterpriseCustomerBrandingConfiguration colors.

9.398 [3.8.36]

- Authenticate user with Tableau.

9.399 [3.8.35]

- Add default branding config object to the Customer record if null.

9.400 [3.8.34]

- Implementing Blackboard completion data tranmission.

9.401 [3.8.33]

- During license revocation, if no audit track exists for the course, attempt to unenroll the learer from it.

9.402 [3.8.32]

- Catches/Handles error occurring with Moodle integrated channel.

9.403 [3.8.31]

- Refactors the revoke endpoint into smaller parts, so that implementing new logic is easier to manage.

9.404 [3.8.30]

- Moodle client bug fix

9.405 [3.8.29]

- Make email field optional for sftp delivery for enterprise reporting config

9.406 [3.8.28]

- Blackboard exporter

9.407 [3.8.27]

- Update `get_service_usernames()` to read from a list variable (that may not exist).

9.408 [3.8.26]

- Moodle completion data implementation

9.409 [3.8.25]

- Blackboard client Oauth2 implementation

9.410 [3.8.24] 2020-10-02

- Allow learners to enroll with their license in courses when DSC is disabled.

9.411 [3.8.23] 2020-10-01

- Added Audit grade for Audit mode enrollments in integrated channels.

9.412 [3.8.22]

- Updated seed_enterprise_devstack_data to enable the test customer's subscription management screen

9.413 [3.8.21] 2020-09-28

- Add functionality to save logo file at only one location when saving EnterpriseCustomerBrandingConfiguration instance

9.414 [3.8.20] 2020-09-24

- Better exception handling for integrated channels.

9.415 [3.8.19] 2020-09-24

- Copy test from edx-platform over to enterprise to test migrations early.

9.416 [3.8.18] 2020-09-23

- Initial setup for Blackboard Integrated Channel.

9.417 [3.8.17] 2020-09-23

- Update logo name and path after the instance is saved to replace None with instance id.

9.418 [3.8.16] 2020-09-22

- Token expiration handling in canvas client.

9.419 [3.8.15] 2020-09-22

- Update Data Sharing Consent language.

9.420 [3.8.14] 2020-09-21

- Add Moodle integration to integrated_channels.

9.421 [3.8.13] 2020-09-20

- Fix issue with canvas channel not finding a course, by using search endpoint

9.422 [3.8.12] 2020-09-21

- Fix column width issue for DSC and other pages

9.423 [3.8.11] 2020-09-18

- Upgrading celery version to 4.4.7 for python 3.8 support

9.424 [3.8.10] 2020-09-17

- Reverting PR #952.

9.425 [3.8.9] 2020-09-16

- Standardizing log format in integrated channels learner data export.

9.426 [3.8.8] 2020-09-15

- Fixing the construction of the next param in the proxy login view for SSO.

9.427 [3.8.7] 2020-09-15

- Adding more informative logs to the integrated channels.

9.428 [3.8.6] 2020-09-15

- Using viewname in reverse as part of args to prevent IndexOutOfRange exception

9.429 [3.8.5] 2020-09-14

- Add a field to EnterpriseCustomer to disable main menu navigation for integrated channel customer users.

9.430 [3.8.4] 2020-09-14

- Add a field for enabling analytics screen in the admin portal for an EnterpriseCustomer.

9.431 [3.8.3] 2020-09-14

- Add management command to create DSC records.

9.432 [3.8.2] 2020-09-11

- Course and Course Run enrollment_url now points to learner portal course page if LP enabled.

9.433 [3.8.1] 2020-09-10

- Canvas channel discovery improvements assorted changes.

9.434 [3.8.0] 2020-09-09

- Assign "enterprise_admin" system-wide role to pending admin users when registering their user account.

9.435 [3.7.8] 2020-09-09

- Fixes migration mismatch for Canvas models.

9.436 [3.7.7] 2020-09-04

- The `seed_enterprise_devstack_data` management command now accepts an enterprise name when creating an enterprise, and the learner portal is activated by default.

9.437 [3.7.6] 2020-09-09

- Adds the learner data exporter and transmitter to the Canvas integrated channel.

9.438 [3.7.5] 2020-09-08

- Celery version is now upgraded to latest one

9.439 [3.7.4] 2020-09-04

- Adds support to capture contract discounts from the Enrollment API by adding `default_contract_discount` to the `EnterpriseCustomer` model and passing it to ecommerce when creating orders

9.440 [3.7.3] 2020-09-01

- Override the `EnterpriseContentCatalog.save()` method to sync the `content_filter` from an associated `EnterpriseCatalogQuery`, if appropriate.
- Add some logging to the `update_enterprise_catalog_query` signal.

9.441 [3.7.2] 2020-09-01

- The `seed_enterprise_devstack_data` management command is now idempotent when creating an enterprise, and creates users and operator roles for the license-manager and enterprise-catalog workers.

9.442 [3.7.1] 2020-08-28

- Also send course image_url to Canvas when creating course.

9.443 [3.7.0] 2020-08-27

- Fixed Duplicate Calls to OCN API.

9.444 [3.6.9] 2020-08-26

- Return requested user's linked enterprises only. For staff user return all enterprises.

9.445 [3.6.8] 2020-08-26

- Added course update and deletion capabilities to the canvas integrated channel.

9.446 [3.6.7] 2020-08-26

- Changed strings in Manage Learners DSC view.

9.447 [3.6.6] 2020-08-24

- Added a fix for "Manual Order Not Fulfilled" bug.

9.448 [3.6.5] 2020-08-24

- Added course mode in ecommerce manual enrollment API.

9.449 [3.6.4] 2020-08-18

- Canvas transmitter implementation for course creation

9.450 [3.6.3] 2020-08-19

- Adding Django admin forms for Canvas integration config and cleanup on models.

9.451 [3.6.2] 2020-08-17

- Adding Canvas integrated channels API endpoint for the oauth process completion

9.452 [3.6.1] 2020-08-17

- Added logging in enrollment endpoint for test purposes.

9.453 [3.6.0] 2020-08-12

- ENT-2939: removing waffle flag and utility function used in enterprise-catalog transition

9.454 [3.5.4] 2020-08-12

- Fixed date format in Cornerstone catalog sync call

9.455 [3.5.3] 2020-08-11

- Fix permissions issue with license_revoke endpoint in LicensedEnterpriseCourseEnrollmentViewSet.

9.456 [3.5.2] 2020-08-11

- Add Content Metadata Exporter for Canvas Integration.

9.457 [3.5.1] 2020-08-11

- Add client instantiation and oauth validation for Canvas integration.

9.458 [3.5.0] 2020-08-10

- Add *update_course_enrollment_mode_for_user* method to the EnrollmentApiClient.
- Create new API endpoint to update the mode for a user's licensed enterprise course enrollments when their enterprise license is revoked.
- Introduce new course run status for *saved_for_later*.
- On revocation of an enterprise license, mark the user's licensed course enrollments as *saved_for_later* and *is_revoked*.

9.459 [3.4.40] 2020-08-05

- Create fresh migrations from scratch for Canvas since this app is yet to run migrations in platform.

9.460 [3.4.39] 2020-08-04

- Remove field 'key' from a canvas integrated_channel model (but not migration yet), step 2/3

9.461 [3.4.38] 2020-08-04

- Migration to remove `banner_border_color` and `banner_background_color` branding config fields.

9.462 [3.4.37] 2020-08-04

- Add new field client_id to canvas model for removing older key field (step 1/3)

9.463 [3.4.36] 2020-08-04

- Remove references to deprecated `banner_border_color` and `banner_background_color` branding config fields.

9.464 [3.4.35] 2020-08-04

- Add postman collection for Canvas integrated channel

9.465 [3.4.34] 2020-08-03

- Migration to copy old color field values to new field.

9.466 [3.4.33] 2020-08-03

- Add BrandingConfiguration primary/secondary/tertiary color fields.

9.467 [3.4.32] 2020-07-31

- Add Canvas integrated_channel first cut.

9.468 [3.4.31] 2020-07-30

- The PendingEnterpriseCustomerUser create action will create an EnterpriseCustomerUser if an `auth.User` record with the given user_email already exists.

9.469 [3.4.30] 2020-07-29

- Add flag to sync updates in an EnterpriseCatalogQuery with its associated EnterpriseCustomerCatalogs.
- Create a post_save signal to overwrite the content_filter with the update.
- Changes should also be sent to the Enterprise Catalog service.

9.470 [3.4.29] 2020-07-29

- Added new view for requesting the DSC for learners for specific course.

9.471 [3.4.28] 2020-07-24

- Add query params to proxy login redirect for new welcome template to be rendered.
- Fixing proxy_login SSO redirect, adding default next param from proxy_login.

9.472 [3.4.27] 2020-07-23

- Adds hide_course_original_price field to the serializer for the EnterpriseCustomer endpoint.

9.473 [3.4.26] 2020-07-20

- Adds proxy login view to allow unauthenticated enterprise learners to login via existing flow from the learner portal.

9.474 [3.3.26] 2020-07-17

- Uses correct course mode slugs during enrollment from GrantDataSharingPermissions.

9.475 [3.3.25] 2020-07-16

- Use the GrantDataSharingPermissions view to enroll licensed users in courses

9.476 [3.3.24] 2020-07-15

- Remove get_due_dates and always return an empty list for due_dates

9.477 [3.3.23] 2020-07-13

- Remove unnecessary data migration

9.478 [3.3.22] 2020-07-13

- Final removal of marked_done field

9.479 [3.3.21] - 2020-07-10

- Gracefully handle when list of subjects for content metadata contains either a list of strings and list of dictionaries

9.480 [3.3.20] - 2020-07-09

- Added new SAML Config option to EnterpriseCustomer in Django admin.

9.481 [3.3.19] - 2020-07-08

- Remove database references to marked_done.

9.482 [3.3.18] - 2020-07-07

- Admin dashboard rules predicates now pass an object into the edx-rbac utility functions.

9.483 [3.3.17] - 2020-07-07

- Created LicensedEnterpriseCourseEnrollment.

9.484 [3.3.16] - 2020-07-02

- Change marked_done on EnterpriseCourseEnrollment mode nullable.

9.485 [3.3.15] - 2020-06-30

- Added health checks for enterprise service.

9.486 [3.3.14] - 2020-06-30

- Added saved_for_later field to the EnterpriseCourseEnrollment model. This will eventually replace the marked_done field.

9.487 [3.3.13] - 2020-06-29

- Changed GrantDataSharingPermission to redirect to the intended course instead of dashboard, if consent is not required

9.488 [3.3.12] - 2020-06-27

- Repair invalid key references in Discovery API Client method.

9.489 [3.3.11] - 2020-06-25

- Restore EnterpriseCatalogQuery functionality to previous state.

9.490 [3.3.10] - 2020-06-24

- xAPI: Include course UUID in activity extensions collection

9.491 [3.3.9] - 2020-06-24

- Remove verbose names from EnterpriseCourseEnrollment model Meta class

9.492 [3.3.8] - 2020-06-23

- Add support to override enrollment attributes for learners

9.493 [3.3.7] - 2020-06-19

- Bug fix: Added missing migration for content_filter validation changes.

9.494 [3.3.6] - 2020-06-17

- Add validation for content_filter subfields in EnterpriseCatalogQuery and EnterpriseCustomerCatalog

9.495 [3.3.5] - 2020-06-17

- Update processing of marked_done field slightly for cleaner boolean usage in client

9.496 [3.3.4] - 2020-06-15

- Update GrantDataSharingPermissionView to accept both; course_run_id as well as course_key

9.497 [3.3.3] - 2020-06-12

- Exclude unpublished course runs when determining available/enrollable status

9.498 [3.3.2] - 2020-06-10

- Added status key to default content filter for EnterpriseCustomerCatalog.

9.499 [3.3.1] - 2020-06-10

- Added marked_done field in /enterprise_course_enrollments/ response

9.500 [3.3.0] - 2020-06-09

- xAPI Integrated Reporting Channel, Version 2

9.501 [3.2.22] - 2020-06-09

- Added rollback for EnterpriseCourseEnrollment enroll

9.502 [3.2.21] - 2020-06-03

- Downgrade an error log to a warning to reduce alert noise

9.503 [3.2.20] - 2020-06-01

- Suppress the 404 exception in get_enterprise_catalog when we expect it
- Add enterprise_customer_uuid to an error message to be more informative
- Delete "enterprise_learner" role assignment when an EnterpriseCustomerUser record is soft deleted (i.e., *linked* attribute is False)
- Update seed_enterprise_devstack_data command to include name on user profiles when creating enterprise users

9.504 [3.2.19] - 2020-06-01

- Updating the catalog preview URL to use the Catalog Service

9.505 [3.2.18] - 2020-05-28

- Added the enterprise slug login functionality.

9.506 [3.2.17] - 2020-05-27

- Improve xAPI enrollment/completion event filtering, transmitting, and recording

9.507 [3.2.16] - 2020-05-27

- Removing caniusepython3 as it is no longer needed since python3 upgrade.

9.508 [3.2.15] - 2020-05-26

- Improve EnterpriseRoleAssigment exception messaging

9.509 [3.2.14] - 2020-05-19

- Converting UUID fields to string for use in can_use_enterprise_catalog

9.510 [3.2.13] - 2020-05-15

- Added can_use_enterprise_catalog utility function to exclude enterprises from the transition to enterprise-catalog

9.511 [3.2.12] - 2020-05-13

- Created migration to *update_or_create* a system-wide enterprise role named *enterprise_catalog_admin*

9.512 [3.2.11] - 2020-05-12

- Moving the post model save logic for Enterprise Catalog to signals.py.

9.513 [3.2.10] - 2020-05-08

- Updated EnterpriseCustomerCatalogAdmin save hook to check if a corresponding catalog exists in the enterprise-catalog service. If it does, the save hook will update the existing catalog; otherwise, a new catalog will be created.
- Added extra logging when syncing Enterprise Catalog data to the Enterprise Catalog Service.

9.514 [3.2.9] - 2020-05-08

- Added a flag to enable the slug login for an enterprise customer.

9.515 [3.2.8] - 2020-05-07

- Makes the data sharing consent template guard against empty/null branding configuration logo values.

9.516 [3.2.7] - 2020-05-07

- Added extra logging in 'create_enterprise_course_enrollments' management command.

9.517 [3.2.6] - 2020-05-06

- Added use of traverse_pagination for get_content_metadata in the enterprise_catalog api client.

9.518 [3.2.5] - 2020-05-06

- Pass enterprise customer's name to enterprise-catalog service during create/update of enterprise catalogs
- Refactor *migrate_enterprise_catalogs* management command to check if a catalog already exists in the enterprise-catalog service. If a catalog already exists, it will be updated with a PUT request; otherwise, a new catalog will be created with a POST request.

9.519 [3.2.4] - 2020-05-06

- Specified python3.5 version for PyPI release

9.520 [3.2.3] - 2020-05-06

- Removed support for Django<2.2 & Python3.6
- Added support for python3.8.
- Changes to use catalog query content filter if defined instead of catalog content filter.

9.521 [3.2.2] - 2020-05-05

- Made enrollment reason optional when linking learners without enrollment.

9.522 [3.2.1] - 2020-05-04

- Added extra logging in 'create_enterprise_course_enrollments' management command.

9.523 [3.2.0] - 2020-04-23

- Squashed the sap_success_factors and integrated_channel app migrations.

9.524 [3.1.3] - 2020-04-23

- Revised "end date" window for determinine course active/inactive status in catalog API responses.

9.525 [3.1.2] - 2020-04-21

- Added extra exception handling in *create_enterprise_course_enrollments* management command.

9.526 [3.1.1] - 2020-04-20

- removed get_cache_key and using it from edx-django-utils.

9.527 [3.1.0] - 2020-04-14

- Squashed the enterprise app migrations.

9.528 [3.0.15] - 2020-04-14

- Fixed HTML tags bug from short course description in enterprise course enrollment page

9.529 [3.0.14] - 2020-04-10

- Fixing the traversal of results in get_content_metadata for the enterprise-catalog API client

9.530 [3.0.13] - 2020-04-10

- Switch catalog_contains_course method to use enterprise catalog service behind waffle sample

9.531 [3.0.12] - 2020-04-10

- Add USE_ENTERPRISE_CATALOG waffle sample, and remove USE_ENTERPRISE_CATALOG waffle flag
- Switch the use of waffle.flag_is_active to waffle.sample_is_active
- Updates the EnterpriseCatalogApiClient to make the user argument optional. If the user argument is not provided, it will use the "enterprise_worker" user instead
- No longer passes user to the EnterpriseCatalogApiClient during initialization in places where a request and/or user object doesn't already exist

9.532 [3.0.11] - 2020-04-10

- Fix issue with matching urls for redirect to enterprise selection page

9.533 [3.0.10] - 2020-04-08

- Use the USE_ENTERPRISE_CATALOG waffle flag for transitioning integrated channels to using the enterprise-catalog service

9.534 [3.0.9] - 2020-04-08

- Add USE_ENTERPRISE_CATALOG waffle flag
- Switch get_course, get_course_run, get_program, and get_course_and_course_run methods to use enterprise catalog service behind waffle flag

9.535 [3.0.8] - 2020-04-08

- Converted the EnrollmentApiClient to JWT client.

9.536 [3.0.7] - 2020-04-07

- Additional xAPI transmission workflow logging

9.537 [3.0.6] - 2020-04-06

- Added support for bypassing enterprise selection page for enrollment url triggered login

9.538 [3.0.5] - 2020-03-31

- Added "active" key in enterprise_catalog API for "course" content_type if the "course" has "course_run" available for enrollment.

9.539 [3.0.4] - 2020-03-31

- Removed the 'EDX_API_KEY' from CourseApiClient.

9.540 [3.0.3] - 2020-03-27

- Updated enterprise-catalog endpoint urls to match rename

9.541 [3.0.2] - 2020-03-26

- Improved xApi logging to include statement and LRS endpoint'

9.542 [3.0.1] - 2020-03-18

- Updated xApi integrated channel to use the updated CourseOverview method 'get_from_ids()'

9.543 [3.0.0] - 2020-03-16

- Removed use of Bearer Authentication

9.544 [2.5.5] - 2020-03-13

- Add field for enabling subscription managment screen in the admin portal to EnterpriseCustomer.

9.545 [2.5.4] - 2020-03-12

- Reset authentication cookies on enterprise selection to update JWT cookie with user's enterprise

9.546 [2.5.3] - 2020-03-11

- Added the salesforce opportunity_id in manage learner django admin.

9.547 [2.5.2] - 2020-03-10

- Fixed formatting on JSON fields in django admin forms

9.548 [2.5.1] - 2020-03-05

- Added new data type for enterprise report configurations

9.549 [2.5.0] - 2020-03-03

- Removing enterprise_learner_portal_hostname from ent cust model (including api)

9.550 [2.4.2] - 2020-02-27

- Removed the code for enrolling the program from manage learner django admin panel.

9.551 [2.4.1] - 2020-02-26

- Update log level from INFO to DEBUG for transmit_content_metadata management command

9.552 [2.4.0] - 2020-02-25

- Restricted PendingEnterpriseCustomerUser to be linked with only one EnterpriseCustomer at a time

9.553 [2.3.9] - 2020-02-17

- Added discount percentage support in pending enrollment use case.

9.554 [2.3.8] - 2020-02-10

- Added totalHours field for successfactors completion event

9.555 [2.3.7] - 2020-02-07

- Learner attached to multiple enterprises, logging in via SSO should be taken to Enterprise selection page

9.556 [2.3.6] - 2020-02-06

- Fixed learner data transmission command when grades API return *user_not_enrolled* error

9.557 [2.3.4] - 2020-02-04

- Remove totalHours field from content metadata export

9.558 [2.3.3] - 2020-02-03

- Added exception handling for enrollment api calls during manual enrollment

9.559 [2.3.2] - 2020-01-31

- Adding contact_email to enterprisecustomer admin form

9.560 [2.3.1] - 2020-01-29

- Updated calls to *manual enrollments api* to include enterprise customer info

9.561 [2.3.0] - 2020-01-29

- Add soft deletion support for EnterpriseCustomerUser model

9.562 [2.2.0] - 2020-01-28

- Adding new fields to EnterpriseCustomer and EnterpriseCustomerBrandingConfiguration models

9.563 [2.1.7] - 2020-01-28

- Revert Edx-Api-Key-replacement-changes

9.564 [2.1.6] - 2020-01-27

- Updating enterprise catalog migration management command

9.565 [2.1.5] - 2020-01-27

- Added totalHours field for successfactors export

9.566 [2.1.4] - 2020-01-24

- add boolean field to track linked/unlinked EnterpriseCustomerUser records

9.567 [2.1.03] - 2020-01-24

- Code refactor and ability to send learner completion if grade is changed

9.568 [2.1.01] - 2020-01-21

- Initialized EnrollmentApiClient with enterprise service worker user

9.569 [2.1.0] - 2020-01-16

- Added hooks to sync EnterpriseCustomerCatalog creation, deletion, and model updates in Django Admin to the new enterprise-catalog service

9.570 [2.0.50] - 2020-01-16

- Replaced EnrollmentApiClientJwt name back to original client's name.

9.571 [2.0.49] - 2020-01-15

- Added management command to reset SAPSF completion data.

9.572 [2.0.48] - 2020-01-14

- Updated enterprise catalog client json formatting.

9.573 [2.0.47] - 2020-01-13

- Replaced Edx-Api-Key in the remaining endpoints of EnrollmentApiClient

9.574 [2.0.46] - 2020-01-10

- Introduced management command to migrate enterprise catalog data to new service.

9.575 [2.0.45] - 2020-01-09

- ENT-2489 | Extracting JSON from discovery service response to calculate size

9.576 [2.0.43] - 2020-01-08

- Replaced Edx-Api-Key in the ThirdPartyAuthApiClient
- Changed the client in one endpoint of ThirdPartyAuthApiClient
- Endpoint name: model-EnterpriseCustomerUser

9.577 [2.0.42] - 2020-01-07

- Updated context for user with multiple linked enterprises

9.578 [2.0.41] - 2020-01-06

- Added enterprise discount percentage in a manual enrollment

9.579 [2.0.40] - 2020-01-06

- Replaced Edx-Api-Key in the EnrollmentApiClient
- Changed the client in one endpoint of EnrollmentApiClient
- Endpoint name: admin-views-EnterpriseCustomerManageLearnersView

9.580 [2.0.39] - 2020-01-06

- Replaced Edx-Api-Key in the CourseApiClient
- Changed the client in one endpoint of CourseApiClient
- Endpoint name: exporters-learnerdata

9.581 [2.0.38] - 2020-01-02

- Changed logging of response size from 2.0.37 (ENT-2489) to use size of response in bytes

9.582 [2.0.37] - 2020-01-02

- Added logging of response size when requests are made to discovery service for data not in cache

9.583 [2.0.36] - 2019-12-30

- Use *edx-tincan-py35* PYPI package instead of downloading via git

9.584 [2.0.35] - 2019-12-30

- Version upgrade for edx-rbac

9.585 [2.0.34] - 2019-12-24

- Disabled the manual enrollment orders for audit mode enterprise learners.

9.586 [2.0.33] - 2019-12-23

- Added ability to include or exclude date from the report configuration file name.

9.587 [2.0.32] - 2019-12-17

- Aligned xAPI statement formats with TinCan/Rustici standards
- While uploading bulk users in 'manager learners' from django admin, better handling if invalid encoding found.

9.588 [2.0.31] - 2019-12-11

- Added ADR for Multiple User Enterprises.

9.589 [2.0.30] - 2019-12-04

- Get the enterprise_customer linked with SAML and mark it active.

9.590 [2.0.29] - 2019-12-04

- Update the enterprise customer in the session in case of customer with multiple linked enterprises

9.591 [2.0.28] - 2019-12-3

- Added logic to set the EnterpriseCourseEnrollmentSource for the Enterprise Enrollments through offers and management task.

9.592 [2.0.27] - 2019-11-26

- Make the SAML enterprise active at login and de-activate other enterprises learner is linked to.

9.593 [2.0.26] - 2019-11-26

- Updated xapi exports with an active enterprise setting for users with multiple linked enterprises.

9.594 [2.0.25] - 2019-11-22

- Added logic to set the EnterpriseCourseEnrollmentSource for the Enterprise Enrollments background task.

9.595 [2.0.24] - 2019-11-21

- Added logic to set the EnterpriseCourseEnrollmentSource for Enterprise Enrollments by URL.

9.596 [2.0.23] - 2019-11-20

- Display enterprise course enrollments separate from non-enterprise course enrollments in the "Enterprise Customer Learner" Django admin form

9.597 [2.0.22] - 2019-11-18

- Custom get function in EnterpriseCustomerUserManager to enable multiple user enterprises.

9.598 [2.0.21] - 2019-11-14

- Remove success url validation for select enterprise page.

9.599 [2.0.20] - 2019-11-13

- Added Source to Enterprise API Enrollments.

9.600 [2.0.19] - 2019-13-08

- Add manual enrollment audit creation for enrollments created in Manage Learners form.

9.601 [2.0.19] - 2019-11-13

- Sorted results of enterprise-learner API by active flag in descending order so active enterprises are on the top

[2.0.18] - 2019-11-13

- Better handling when Integrated Channels return unexpected results

9.602 [2.0.17] - 2019-11-08

- Added in models to track enterprise enrollment source and updated the Enterprise Course Enrollments and PendingEnrollments to track that source.

9.603 [2.0.16] - 2019-11-07

- Address defect ENT-2463. Add protection within EnterpriseCustomerUser model in enroll method during coure enrollments.

9.604 [2.0.15] - 2019-11-07

- Added missing migration for EnterpriseCustomerUser

9.605 [2.0.14] - 2019-11-07

- Add Enterprise selection page to allow a learner to select one of linked enterprises

9.606 [2.0.13] - 2019-11-07

- Add manual order creation to enterprise manual enrollment admin form

9.607 [2.0.12] - 2019-11-06

- Update 'EnterpriseCustomerUser' model. Add 'create_order_for_enrollment'. Called during 'enroll'. Will create an ecommerce order for pending course enrollments.

9.608 [2.0.11] - 2019-11-06

- Add management command to populate sample enterprise data in the LMS within devstack

9.609 [2.0.10] - 2019-10-29

- Add method to Ecommerce API client to call the manual enrollment order API

9.610 [2.0.9] - 2019-10-28

- Updated image url field in content metadata export for cornerstone and degreed

9.611 [2.0.8] - 2019-10-22

- Adding logging to search/all/ endpoint in discovery api client

9.612 [2.0.7] - 2019-10-21

- Added certificate and grades api calls for transmitting learner export to integrated channels

9.613 [2.0.6] - 2019-10-18

- Add query_param to remove expired course runs from /enterprise/api/v1/enterprise_catalogs/UUID/ endpoint

9.614 [2.0.5] - 2019-10-15

- Adding migration file to remove EnterpriseCustomerEntitlement from table schema

9.615 [2.0.4] - 2019-10-10

- Added preview button for EnterpriseCustomerCatalogs in EnterpriseCustomer admin page

9.616 [2.0.3] - 2019-10-09

- Add message box to code management page and admin portal

9.617 [2.0.2] - 2019-10-07

- Updating create_enterprise_course_enrollment task to accept object ids instead of python objects to play nicely with async.
- Also converts course_id to str before handing it to task to play nicely with async.

9.618 [2.0.1] - 2019-10-07

- Commenting out code while troubleshooting signal issue in the LMS

9.619 [2.0.0] - 2019-10-02

- Removing EnterpriseCustomerEntitlement code

9.620 [1.11.0] - 2019-10-02

- Adding post-save receiver to spin off EnterpriseCourseEnrollment creation tasks on CourseEnrollment creation signals

9.621 [1.10.8] - 2019-10-01

- Resolved issue with content_metadata image_url.

9.622 [1.10.7] - 2019-09-25

- Added support to transmit single learner data.

9.623 [1.10.6] - 2019-09-25

- Added ability set supported languages in Cornerstone Global Config.

9.624 [1.10.5] - 2019-09-23

- Updating enterprise_learner_portal LMS API calls to refer to new function locations in the LMS.

9.625 [1.10.4] - 2019-09-05

- Added new endpoint basic_list to EnterpriseEnrollment.

9.626 [1.10.3] - 2019-09-19

- Add enable_portal_reoprting_config_screen field to EnterpriseCustomer model.
- Add enable_portal_reporting_config_screen to EnterpriseCustomerSerializer.

9.627 [1.10.2] - 2019-09-18

- Added ability to set password on reporting configuration.

9.628 [1.10.1] - 2019-09-16

- Upgrading requirements.

9.629 [1.10.0] - 2019-09-16

- Add learner portal configuration fields to EnterpriseCustomer model.

9.630 [1.9.12] - 2019-09-06

- Implement "move to completed" functionality for Enterprise Enrollments.

9.631 [1.9.11] - 2019-09-05

- Add new field 'marked_done' to EnterpriseCourseEnrollment.

9.632 [1.9.10] - 2019-09-04

- Improved enterprise enrollment workflow logging.

9.633 [1.9.9] - 2019-08-29

- Updated learner portal enrollments endpoint to require an enterprise id.

9.634 [1.9.8] - 2019-08-29

- Corrected missing db migration data for the EnterpriseCustomerReportingConfigurations model

9.635 [1.9.7] - 2019-08-28

- Added API endpoints for EnterpriseCustomerReportingConfigurations and updated permissions to use Feature role based auth.

9.636 [1.9.6] - 2019-08-23

- Added XAPILearnerDataTransmissionAudit model for xapi integrated channel.

9.637 [1.9.5] - 2019-08-21

- Preventing another error in enterprise_learner_portal serializer when certificate info is None.

9.638 [1.9.4] - 2019-08-20

- Adding type check to enterprise_learner_portal serializer.
- Adding enterprise_learner_portal to quality check commands.

9.639 [1.9.3] - 2019-08-20

- Fix for include course run dates and pacing type in the course description sent to SAP. Prior release (1.9.2) did not include bumping the version in __init__.py.

9.640 [1.9.2] - 2019-08-20

- Include course run dates and pacing type in the course description sent to SAP.

9.641 [1.9.1] - 2019-08-19

- Added enterprise_learner_portal to MANIFEST.in file to recursively grab files app on build
- Minor fixes to typos and an image link

9.642 [1.9.0] - 2019-08-12

- Adding enterprise_learner_portal app to support data needs of frontend enterprise learner portal app

9.643 [1.8.9] - 2019-08-15

- Remove tincan from src directory

9.644 [1.8.8] - 2019-08-01

- For CornerstoneCourseListAPI handled corner cases for default values.

9.645 [1.8.7] - 2019-07-31

- Added history models for PendingEnrollment and PendingEnterpriseCustomerUser.
- Sending default values for required fields in Cornerstone Course List API

9.646 [1.8.6] - 2019-07-25

- Add/Update logs for GrantDataSharingPermissions and DataSharingConsentView views to improve monitoring.

9.647 [1.8.5] - 2019-07-25

- Change coupon code request email from address.

9.648 [1.8.4] - 2019-07-24

- Introduce enterprise catalog queries.

9.649 [1.8.3] - 2019-07-24

- Upgrade python requirements.

9.650 [1.8.2] - 2019-07-23

- Log success of coupon code request email send.

9.651 [1.8.1] - 2019-07-22

- Show linked enterprise customer on *Enterprise Customer Learners* and *System wide Enterprise User Role Assignments* admin screen

9.652 [1.8.0] - 2019-07-22

- Replace edx-rbac jwt utils with edx-drf-extensions jwt utils

9.653 [1.7.3] - 2019-07-19

- Change the way we declare dependencies so we can avoid breaking make upgrade in edx-platform.

9.654 [1.7.2] - 2019-07-18

- Added ability to send user's progress to cornerstone

9.655 [1.7.1] - 2019-07-15

- Reverted page size of SAPSF inactive user results from 1000 to 500

9.656 [1.7.0] - 2019-07-15

- Pin certain constraints from edx-platform so that edx-enterprise will install properly there.

9.657 [1.6.23] - 2019-07-15

- Upgrade python requirements

9.658 [1.6.22] - 2019-07-11

- Revert changes made in 1.6.20

9.659 [1.6.21] - 2019-07-11

- Added additional logging for enterprise api

9.660 [1.6.20] - 2019-07-10

- Updated catalog preview URL on enterprise customer catalog admin list display

9.661 [1.6.19] - 2019-07-09

- Added ability to skip keys if their value is None for content exporter

9.662 [1.6.18] - 2019-06-24

- Changed page size of SAPSF inactive user results from 500 to 1000

9.663 [1.6.17] - 2019-06-20

- Fixed Server Error on enterprise course enroll url caused by week_to_complete None value

9.664 [1.6.16] - 2019-06-20

- Capture user attributes sent by cornerstone

9.665 [1.6.15] - 2019-06-18

- Fix error where the search/all/ endpoint in discovery is called with course_key=None

9.666 [1.6.14] - 2019-06-18

- Pass language code instead of language name in languages field of course-list API for cornerstone

9.667 [1.6.13] - 2019-06-17

- Improved logging of *unlink_inactive_sap_learners* command and matching social auth user by *uid* field

9.668 [1.6.12] - 2019-06-14

- Updated discovery clients to always call the enterprise customer site if available

9.669 [1.6.11] - 2019-06-14

- Update the format of course_duration in xAPI payload data.

9.670 [1.6.10] - 2019-06-13

- Remove old catalog model field.

9.671 [1.6.9] - 2019-06-12

- Install django-filter so this app is compatible with newer DRF packages.

9.672 [1.6.8] - 2019-06-11

- Fix error in enrollment flow caused by the way course keys were parsed.

9.673 [1.6.7] - 2019-06-11

- added enable_audit_data_reporting in EnterpriseCustomerSerializer

9.674 [1.6.6] - 2019-06-10

- Use OAuth2AuthenticationAllowInactiveUser as oauth2 authentication instead of BearerAuthentication for course-list API.

9.675 [1.6.5] - 2019-06-06

- Use edx-rbac functions and pin edx-rbac so that we can continue to release edx-enterprise.

9.676 [1.6.4] - 2019-06-05

- Upgrade packages to get latest edx-drf-extensions version.

9.677 [1.6.3] - 2019-06-04

- Remove RBAC waffle switch

9.678 [1.6.2] - 2019-05-31

- Remove old style catalogs

9.679 [1.6.1] - 2019-05-30

- Fallback to request.auth if JWT cookies are not found.

9.680 [1.6.0] - 2019-05-29

- Added new integrated channel *cornerstone* with course-list API.

9.681 [1.5.9] - 2019-05-27

- Reverting changes from 1.5.6.

9.682 [1.5.8] - 2019-05-24

- Bumping version to 1.5.8. 1.5.7 was tagged and released without actually bumping the version

9.683 [1.5.7] - 2019-05-24

- Updating get_paginated_content ent catalog method to use count value given from discovery service

9.684 [1.5.6] - 2019-05-24

- Fix the way a course identifier is found for a given course run.

9.685 [1.5.5] - 2019-05-21

- Clean up rbac authorization related waffle switches and logic

9.686 [1.5.4] - 2019-05-20

- Updating test packages to be inline with edx-platform. Specifically Bleach >2.1.3

9.687 [1.5.3] - 2019-05-16

- Add total number of weeks to view from data consent screen

9.688 [1.5.2] - 2019-05-15

- Remove usages of get_decoded_jwt_from_request from rbac in favor of get_decoded_jwt from edx-drf-extensions

9.689 [1.5.1] - 2019-05-09

- Updating consent granted view to redirect to dashboard if consent is not required

9.690 [1.5.0] - 2019-05-08

- Add sync_learner_profile_data flag to data returned by enterprise-learner endpoint

9.691 [1.4.10] - 2019-05-08

- Add enterprise customer column in the list_display admin interface for *SystemWideEnterpriseUserRoleAssignment*
- Update *SystemWideEnterpriseUserRoleAssignment* admin interface search to support search by enterprise customer

9.692 [1.4.9] - 2019-05-02

- Upgrade edx-rbac version

9.693 [1.4.8] - 2019-04-26

- Reduce course mode match exception log level

9.694 [1.4.7] - 2019-04-17

- Fix invalid object attribute references in exception message

9.695 [1.4.6] - 2019-04-17

- Stop masking discovery call failures from the client for enterprise catalog endpoint calls.

9.696 [1.4.5] - 2019-04-12

- Revise course mode match exception message in CourseEnrollmentView.

9.697 [1.4.4] - 2019-04-11

- Revise course load exception message in CourseEnrollmentView.

9.698 [1.4.3] - 2019-04-11

- Added *availability* key to default content filter for ECC.

9.699 [1.4.2] - 2019-04-11

- Update *assign_enterprise_user_roles* management command to also assign catalog and enrollment api admin roles.

9.700 [1.4.1] - 2019-04-10

- Update *RouterView* if user is already enrolled in course run of a course then user will land on that course_run.

9.701 [1.4.0] - 2019-04-08

- Add new rbac permission checks to enterprise api endpoints.

9.702 [1.3.11] - 2019-04-07

- Update context for *enterprise-openedx-operator* role.

9.703 [1.3.10] - 2019-04-03

- Provide ability to add ECE even if course is closed from manage learners admin interface.

9.704 [1.3.9] - 2019-03-29

- Update role metadata for *edx-openedx-operator* role.

9.705 [1.3.8] - 2019-03-29

- Update *assign_enterprise_user_roles* management command to also assign enterprise operator role.

9.706 [1.3.7] - 2019-03-28

- Add data migration for adding edx enterprise operator role.

9.707 [1.3.6] - 2019-03-27

- Introduce rbac models for feature specific roles within edx-enterprise.

9.708 [1.3.5] - 2019-03-22

- Assign an enterprise learner role to new EnterpriseCustomerUser.

9.709 [1.3.4] - 2019-03-21

- Management command to assign enterprise roles to users.

9.710 [1.3.3] - 2019-03-21

- Fixed error in enrollment flow when audit track is selected and no DSC required.

9.711 [1.3.2] - 2019-03-18

- Adding django admin for SystemWideEnterpriseUserRoleAssignments.

9.712 [1.3.1] - 2019-03-13

- Optimizations around unlinking of SAP Success factor inactive users

9.713 [1.3.0] - 2019-03-07

- Introducing Enterprise System Wide Roles and edx-rbac.

9.714 [1.2.12] - 2019-02-15

- Updating enterprise views with new logging
- Updating enterprise views to render new error page in a number of circumstances

9.715 [1.2.11] - 2019-02-07

- Allow admins with enterprise permissions to edit Data Sharing Consent Records

9.716 [1.2.10] - 2019-01-30

- Include Enterprise Catalog UUID in Enterprise Customer django admin inline.

9.717 [1.2.9] - 2019-01-23

- Upgrade requirements, and add code-annotations.
- Add PII annotations to all apps in this repo.
- Enable PII checking during CI.

9.718 [1.2.8] - 2019-01-22

- Revert 1.2.4 to restore DSC functionality.

9.719 [1.2.7] - 2019-01-18

- Replace error level log with info level log when enterprise user is not enrolled in course yet and the *transmit_learner_data* command is run

9.720 [1.2.5] - 2019-01-16

- Updating launch_points data in SapSuccessFactorsContentMetadataExporter so SuccessFactors can be mobile ready

9.721 [1.2.4] - 2019-01-16

- Remove HandleConsentEnrollment view and replaced with a function inside GrantDataSharingPermissions view. Removed GET side effect

9.722 [1.2.3] - 2019-01-10

- Add management command "unlink_inactive_sap_learners" to unlink inactive SAP learners from the related enterprises

9.723 [1.2.2] - 2019-01-09

- Update styling for future courses start date visibility

9.724 [1.2.1] - 2018-12-21

- Handle /search/all/ endpoint large catalog queries to discovery through HTTP POST

9.725 [1.2.0] - 2018-12-19

- Updating the course grade api url called in lms api

9.726 [1.1.4] - 2018-12-19

- Upgrade django-simple-history required version

9.727 [1.1.3] - 2018-12-18

- Add option on EnterpriseCustomer for displaying code management in portal

9.728 [1.1.2] - 2018-12-12

- Update EnterpriseCustomer model to introduce customer type field

9.729 [1.1.1] - 2018-12-11

- Use LMS-defined segment track() method

9.730 [1.1.0] - 2018-12-06

- Updating EnterpriseCustomerReportingConfiguration model. ManyToMany relationship with EnterpriseCustomerCatalog
- Updating EnterpriseCustomerReportingConfigurationAdminForm validation
- Updating EnterpriseCustomerReportingConfigurationSerializer

9.731 [1.0.6] - 2018-11-28

- Added username and user email in EnterpriseCustomerUserAdmin list display.
- Added search by username and user email in EnterpriseCustomerUserAdmin.

9.732 [1.0.5] - 2018-11-14

- Added enterprise api for requesting additional coupon codes.

9.733 [1.0.4] - 2018-11-07

- Make HTTP POST request to get catalog results from discovery.

9.734 [1.0.3] - 2018-11-02

- Fix translations for enterprise pages.

9.735 [1.0.2] - 2018-10-25

- Updated EnterpriseCustomerReportingConfiguration model with PGP key

9.736 [1.0.1] - 2018-10-24

- Made autocohorting API availability based on a configuration option.

9.737 [1.0.0] - 2018-10-16

- Upgrade edx-drf-extensions with refactored imports.
- Remove Hawthorn testing for upcoming backward incompatible change.

9.738 [0.73.6] - 2018-10-04

- SuccessFactors: Submit batch/chunk of OCN items to tenants until error status

9.739 [0.73.5] - 2018-09-21

- Added ability to query enterprises by slug on the with_access_to endpoint

9.740 [0.73.4] - 2018-09-17

- Added ability to assign cohort upon enrollment.
- Added ability to unenroll in enrollment API.

9.741 [0.73.3] - 2018-09-14

- Added Country field to the EnterpriseCustomer model.

9.742 [0.73.2] - 2018-09-11

- Fixed 500 error on enterprise customer admin screen.

9.743 [0.73.1] - 2018-08-30

- Remove the SailThru flags for enterprise learner when un-linking it from enterprise.

9.744 [0.73.0] - 2018-08-21

- Changed permission logic and added filtering options for the enterprise with_access_to endpoint.

9.745 [0.72.7] - 2018-08-20

- Added preview field that takes user to Discovery with elastic search results for the catalog

9.746 [0.72.6] - 2018-08-17

- Added management command to send course enrollment and course completion info for enterprise customers.

9.747 [0.72.5] - 2018-08-09

- Revise management command query to include all potentially-applicable enrollment records

9.748 [0.72.4] - 2018-08-08

- Move some fields from Global Degreed Configuration to Enterprise Degreed Configuration.

9.749 [0.72.3] - 2018-08-08

- Added LearnerInfoSerializer and CourseInfoSerializer for serializing xAPI payload data.

9.750 [0.72.2] - 2018-07-27

- Added endpoint to check a user's authorization to Enterprises based on membership in a given django group.

9.751 [0.72.1] - 2018-07-26

- Added missing migrations for xAPI LRS Configuration model

9.752 [0.72.0] - 2018-07-24

- Implemented reporting channel of course completion via X-API

9.753 [0.71.2] - 2018-07-23

- Add thumbnail images in exported metadata content by content type.

9.754 [0.71.1] - 2018-07-23

- Updated message for invalid Enterprise Customer Catalog references in B2B enrollment workflow.

9.755 [0.71.0] - 2018-07-20

- Updated TinCanPython package to support python 3
- Updated UUID field to nowrap in admin interface of enterprise customer catalog model.

9.756 [0.70.8] - 2018-07-13

- Display customer catalog content filter's default value on enterprise customer admin.

9.757 [0.70.7] - 2018-07-12

- Make customer catalog content filter's default value configurable.

9.758 [0.70.6] - 2018-07-09

- Pass catalog value only when provided on enterprise course enrollment page.

9.759 [0.70.5] - 2018-07-06

- Send learner data transmissions to integrated channels by course key and course run id.

9.760 [0.70.4] - 2018-07-03

- Use query param "catalog" instead of "enterprise_customer_catalog_uuid" for catalog based enterprise discounts.

9.761 [0.70.3] - 2018-06-29

- Apply enterprise catalog conditional offer by the provided enterprise catalog UUID.

9.762 [0.70.2] - 2018-06-28

- Modify enterprise branding config API to use enterprise slug as the lookup_field.

9.763 [0.70.1] - 2018-06-27

- Paginate linked learners list on manage learners Django admin view.

9.764 [0.70.0] - 2018-06-26

- Add unique slug field to EnterpriseCustomer.

9.765 [0.69.6] - 2018-06-25

- Update requirements to fix pip install issues and to keep in line with edx-platform.

9.766 [0.69.5] - 2018-06-25

- Fix the Direct-to-Audit enrollment issue in case of course instead of course run.

9.767 [0.69.4] - 2018-06-20

- Strip locale values.

9.768 [0.69.3] - 2018-06-20

- Add and transmit customer specific locales so that SuccessFactors show course title and description.

9.769 [0.69.2] - 2018-06-18

- Fix the Direct-to-Audit enrollment issue in case of course.

9.770 [0.69.1] - 2018-06-07

- 500 error when attempting to enroll using course-level URL.

9.771 [0.69.0] - 2018-05-31

- Add a *progress_v2* option in the reporting config to be used for data API fetching.

9.772 [0.68.9] - 2018-05-31

- Increased character limit from 20 to 255 for field title in EnterpriseCustomerCatalog model
- Reorder list display for EnterpriseCustomerCatalogAdmin
- Add sorting order for EnterpriseCustomerCatalogAdmin

9.773 [0.68.8] - 2018-05-30

- Mark ECU as inactive internally if SAPSF says the ECU is inactive on their side.

9.774 [0.68.7] - 2018-05-24

- Admin tooling enterprise customer reporting configuration enhancement - Order by Enterprise Customer Name.

9.775 [0.68.6] - 2018-05-22

- Update DSC to show notification interstitial communicating to enterprise learner they are leaving company's site.

9.776 [0.68.5] - 2018-05-17

- Configuration to show/hide original price on enterprise course landing page.

9.777 [0.68.4] - 2018-05-16

- Remove constraints on the reporting config.

9.778 [0.68.3] - 2018-05-11

- Update enrollment api authorization to check group permissions.

9.779 [0.68.2] - 2018-05-10

- Dropped sap_success_factors_historicalsapsuccessfactorsenterprisecus80ad table.

9.780 [0.68.1] - 2018-05-09

- Add *json* report type.

9.781 [0.68.0] - 2018-05-09

- Allow reporting configs to work for arbitrary data and report types.

9.782 [0.67.8] - 2018-05-04

- Added ordering to resolve warnings of probable invalid pagination data.

9.783 [0.67.7] - 2018-04-23

- Update the messages when an enterprise learner leave an organization.

9.784 [0.67.6] - 2018-04-20

- Update user session when they become an Enterprise learner.

9.785 [0.67.5] - 2018-04-18

- Added ability to specify data sharing consent wording on a per enterprise basis.

9.786 [0.67.4] - 2018-04-12

- Add configuration to allow replacing potentially sensitive SSO usernames.

9.787 [0.67.3] - 2018-04-05

- Improved integrated channel logging.

9.788 [0.67.2] - 2018-04-05

- Fix the enterprise manage learner django admin tool is loading correctly for chrome users.

9.789 [0.67.1] - 2018-04-04

- Integrated channel refactoring cleanup.

9.790 [0.67.0] - 2018-03-26

- Refactored integrated channel code to allow for greater flexibility when transmitting content metadata.

9.791 [0.66.2] - 2018-03-26

- Update isort version and sort imports after making consent and integrated_channels first party apps.

9.792 [0.66.1] - 2018-03-23

- Temporarily disable linked learners list on manage learners Django admin view until paging can be added.

9.793 [0.66.0] - 2018-03-05

- Add EnterpriseCustomerCatalog course detail endpoint.

9.794 [0.65.8] - 2018-02-23

- Add "Enrollment Closed" in course title if the course is no longer open for enrollment.

9.795 [0.65.7] - 2018-02-14

- Support multiple emails in EnterpriseCustomerReportingConfiguration.
- Only require email(s) in EnterpriseCustomerReportingConfiguration if the selected delivery method is email.

9.796 [0.65.6] - 2018-02-13

- Remove the renderer.py file.

9.797 [0.65.5] - 2018-02-13

- Add functionality in enterprise django admin for transmitting courses metadata related to a specific enterprise.

9.798 [0.65.4] - 2018-02-09

- Indicate when a course is no longer open for enrollment by updating course title for transmit courses metadata.

9.799 [0.65.3] - 2018-02-06

- Decreased SuccessFactors course metadata chunk size from 1000 to 500, per SAP's recommendation.

9.800 [0.65.2] - 2018-02-05

- Updated the "Data Sharing Policy" language.

9.801 [0.65.1] - 2018-02-02

- Provide an option for enterprise to pull enterprise catalog API in XML format not just JSON.

9.802 [0.65.0] - 2018-01-30

- Add migration for removing old password fields from the database.

9.803 [0.64.0] - 2018-01-29

- Removed code references to old password fields.

9.804 [0.63.0] - 2018-01-25

- Improved handling of password fields on database models.

9.805 [0.62.0] - 2018-01-18

- Exclude credit course mode option from course enrollment page.

9.806 [0.61.6] - 2018-01-18

- Group Name, Active, Site, and Logo together.
- Rename "Provider id" form label to "Identity Provider"
- Rename "Entitlement id" form label to "Seat Entitlement"
- Rename "Coupon URL" form label to "Seat Entitlement URL"
- Add a "View details" hyperlink next to identity provider drop-down.
- Add a "Create a new catalog" link under the Catalog drop-down.
- Add a "View details" hyperlink next to catalog field, if catalog is selected.
- Add a "Create a new identity provider" link under the Identity Provider drop-down.

9.807 [0.61.5] - 2018-01-18

- Include start date in all course runs title when pushing to Integrated Channels.

9.808 [0.61.4] - 2018-01-12

- Add localized currency to enterprise landing page.

9.809 [0.61.3] - 2018-01-11

- Fix enterprise logo stretching issue in enterprise sidebar on course/program enrollment pages.

9.810 [0.61.2] - 2018-01-09

- Add missing migrations for sap_success_factors and degreed.

9.811 [0.61.1] - 2018-01-09

- Update django admin list view for enterprise customer model.

9.812 [0.61.0] - 2018-01-09

- SuccessFactors Admin Update: Enterprise Customer Configuration.

9.813 [0.60.0] - 2018-01-03

- Add sftp configuration options for EnterpriseCustomerReportingConfiguration.

9.814 [0.59.0] - 2017-12-28

- Add check for active companies when getting list of channels

9.815 [0.58.0] - 2017-12-22

- Add save_enterprise_customer_users command.

9.816 [0.57.0] - 2017-12-21

- Remove references to SSO IdP config drop_existing_session flag.

9.817 [0.56.5] - 2017-12-20

- Fix templates to use new bootstrap bundle library.

9.818 [0.56.4] - 2017-12-19

- Fix syntax error in template-embedded Javascript.

9.819 [0.56.3] - 2017-12-14

- Make sure root url has a fallback for proxy enrollment email links.

9.820 [0.56.2] - 2017-12-13

- Add course_enrollments API endpoint to swagger specification.

9.821 [0.56.1] - 2017-12-13

- Add publish_audit_enrollment_url flag to EnterpriseCustomerCatalog.

9.822 [0.56.0] - 2017-12-13

- Update create_enterprise_course_enrollment command.

9.823 [0.55.7] - 2017-12-13

- Ensure that proxy enrollment email links trigger SSO.

9.824 [0.55.6] - 2017-12-12

- Check site configuration for from email address first

9.825 [0.55.5] - 2017-12-11

- Added course start date to title string for instructor-led courses

9.826 [0.55.4] - 2017-12-06

- Redirect to embargo restriction message page if user is blocked from accessing course.

9.827 [0.55.3] - 2017-12-05

- Add integrated channel configuration info to course metadata push task logging.

9.828 [0.55.2] - 2017-12-04

- Include additional context for learner data transmission job exceptions.

9.829 [0.55.1] - 2017-11-30

- Track enterprise course enrollment events.

9.830 [0.55.0] - 2017-11-29

- Add Degreed as new integrated channel.

9.831 [0.54.1] - 2017-11-29

- Increase font size on data sharing consent page.

9.832 [0.54.0] - 2017-11-28

- Introduce the bulk enrollment/upgrade api endpoint for Enterprise Customers.

9.833 [0.53.19] - 2017-11-28

- Do not change EnterpriseCustomerReportingConfiguration.password on update.

9.834 [0.53.18] - 2017-11-28

- Add Identity Provider's ID to enterprise customer API response.

9.835 [0.53.17] - 2017-11-27

- Remove inaccurate landing page audit track language.

9.836 [0.53.16] - 2017-11-22

- Use LMS_INTERNAL_ROOT_URL instead of LMS_ROOT_URL for API base.

9.837 [0.53.15] - 2017-11-16

- Use the cryptography package instead of the unmaintained pycrypto.

9.838 [0.53.14] - 2017-11-14

- Link learner to enterprise customer directly using "tpa_hint" URL parameter.

9.839 [0.53.13] - 2017-11-14

- Update DSC policy to match legal requirements.

9.840 [0.53.12] - 2017-11-09

- Remove "Discount provided by…" text on the program landing page.

9.841 [0.53.11] - 2017-11-06

- Removing SAP_USE_ENTERPRISE_ENROLLMENT_PAGE switch via django waffle and use landing page URL instead of track selection page.

9.842 [0.53.10] - 2017-11-02

- Move data sharing policy to its own partial to improve theming of the data sharing consent page

9.843 [0.53.9] - 2017-11-02

- Apply appropriate content filtering to the EnterpriseCustomerCatalog detail endpoints.

9.844 [0.53.8] - 2017-11-02

- Show generic info message on enterprise course enrollment page.

9.845 [0.53.7] - 2017-10-30

- Added inline admin form to EnterpriseCustomer admin for EnterpriseCustomerCatalog.

9.846 [0.53.6] - 2017-10-30

- Fix error for empty course start date on DSC page.

9.847 [0.53.5] - 2017-10-26

- Fetch catalog courses in large chunks to avoid API limit.

9.848 [0.53.4] - 2017-10-26

- Preserve catalog querystring on declining DSC.

9.849 [0.53.3] - 2017-10-26

- Fixing logo size on themed enterprise pages

9.850 [0.53.2] - 2017-10-24

- Remove unused dependency on django-extensions

9.851 [0.53.1] - 2017-10-24

- Fix alteration in querystring parameters for decorator "enterprise_login_required".

9.852 [0.53.0] - 2017-10-24

- Get rid of the *EnterpriseIntegratedChannel* model and any other related but unused code.

9.853 [0.52.10] - 2017-10-23

- Fix migration issue for *enabled-course-modes* field of EnterpriseCustomerCatalog

9.854 [0.52.9] - 2017-10-20

- Update the call level to enrollment uls from EnterpriseCustomer to EnterpriseCustomerCatalog.

9.855 [0.52.8] - 2017-10-20

- Update EnterpriseApiClient.get_enterprise_courses to account for EnterpriseCustomerCatalogs.

9.856 [0.52.7] - 2017-10-20

- Update course enrollment view for enterprise enabled course modes.

9.857 [0.52.6] - 2017-10-19

- Update the EnterpriseCustomerCatalog migration.

9.858 [0.52.5] - 2017-10-19

- Add EnterpriseCustomerCatalog UUID as query parameter "catalog" in enterprise course and program enrollment URL's.

9.859 [0.52.4] - 2017-10-18

- Upgrade django-simple-history to 1.9.0. Add needed migrations.

9.860 [0.52.3] - 2017-10-18

- Introducing EnterpriseCustomerReportingConfig model for enterprise_reporting.

9.861 [0.52.2] - 2017-10-18

- If a course is unenrollable, the program and course enrollment landing pages will display only a subset of information.

9.862 [0.52.1] - 2017-10-15

- Change a log level from *error* to *info* in our LMS API Client, as it wasn't really an error.

9.863 [0.52.0] - 2017-10-14

- Implement a direct-audit-enrollment pathway for course enrollment.
- Implement a RouterView that the enrollment URLs have to go through before redirection to a downstream view.

9.864 [0.51.5] - 2017-10-11

- Added enabled_course_modes JSONField to EnterpriseCustomerCatalog model

9.865 [0.51.4] - 2017-10-11

- Added UTM parameters to marketing, track selection, and course/program enrollment URLs returned by Enterprise API.

9.866 [0.51.3] - 2017-10-10

- Fix bug related to EnterpriseCustomer creation form introduced with 0.51.0.

9.867 [0.51.2] - 2017-10-10

- Modify EnterpriseCustomer.catalog_contains_course to check EnterpriseCustomerCatalogs.

9.868 [0.51.1] - 2017-10-06

- Refactor user-facing DSC view's logic.

9.869 [0.51.0] - 2017-10-05

- Make discovery-service lookups site-aware

9.870 [0.50.1] - 2017-10-03

- Improved robustness for *force_fresh_session* decorator in conjunction with *enterprise_login_required*
- Consciously avoid attempting to sync back details for SAPSF users who aren't linked via SSO

9.871 [0.50.0] - 2017-10-03

- Add contains_content_items endpoint to EnterpriseCustomerViewSet and EnterpriseCustomerCatalogViewSet.

9.872 [0.49.0] - 2017-10-02

- Rewrite all of our CSS in SASS/SCSS.
- Use Bootstrap for our modals.
- Fix existing course modal UI issues using Bootstrap & SASS/SCSS.

9.873 [0.48.2] - 2017-09-29

- Step 2 in making enrollment email template linked to enterprise. Remove site from model. No migration.

9.874 [0.48.1] - 2017-09-25

- Step 1 in making enrollment email template linked to enterprise. Make 'site' nullable, add 'enterprise_customer'.

9.875 [0.48.0] - 2017-09-25

- Add extra details to the program enrollment landing page.

9.876 [0.47.1] - 2017-09-25

- Add proper permissions/filtering schemes for all of our endpoints.

9.877 [0.47.0] - 2017-09-21

- Step 3 in safe deployment of removing old consent models: make migrations to delete the outstanding fields/models.

9.878 [0.46.8] - 2017-09-21

- Step 2 in safe deployment of removing old consent models: remove *require_account_level_consent*, but no migration.

9.879 [0.46.7] - 2017-09-21

- Step 1 in safe deployment of removing old consent models: make *require_account_level_consent* nullable.

9.880 [0.46.6] - 2017-09-21

- Added some log messages to trace possible 404 issue.

9.881 [0.46.5] - 2017-09-21

- Remove old account-level consent features as well as consent from EnterpriseCourseEnrollment.

9.882 [0.46.4] - 2017-09-20

- Abstract away usage of *configuration_helpers*.

9.883 [0.46.3] - 2017-09-19

- Make bulk enrollment emails more intelligent

9.884 [0.46.2] - 2017-09-19

- Add exception handling for transmit course metadata task.

9.885 [0.46.1] - 2017-09-18

- Remove the *auth-user* endpoint completely.

9.886 [0.46.0] - 2017-09-15

- Allow multi-course enrollment for enterprise users in admin.

9.887 [0.45.0] - 2017-09-14

- Modified enterprise-learner API endpoint to include the new DataSharingConsent model data.

9.888 [0.44.0] - 2017-09-08

- Added MVP version of the Programs Enrollment Landing Page.

9.889 [0.43.5] - 2017-09-08

- Wrapped API error handling into the clients themselves.

9.890 [0.43.4] - 2017-09-07

- Removed the text if there is no discount on the course enrollment landing page.

9.891 [0.43.3] - 2017-09-06

- Ensure that segment is loaded and firing page events for all user facing enterprise views.

9.892 [0.43.2] - 2017-09-06

- Display the enterprise discounted text on the course enrollment landing page.

9.893 [0.43.1] - 2017-09-05

- Remove support for writing consent_granted in enterprise-course-enrollment api.

9.894 [0.43.0] - 2017-08-31

- Add architecture for program-scoped data sharing consent.

9.895 [0.42.0] - 2017-08-24

- Do not create baskets and orders for audit enrollments.

9.896 [0.41.0] - 2017-08-24

- Migrate the codebase to the new *consent.models.DataSharingConsent* model for when dealing with consent.

9.897 [0.40.7] - 2017-08-23

- Fix bug causing 500 error on course enrollment page when the course does not have a course image configured.

9.898 [0.40.6] - 2017-08-23

- Update Consent API to use Discovery worker user for auth, rather than request user.

9.899 [0.40.5] - 2017-08-23

- Update SAP course export to use enterprise courses API.

9.900 [0.40.4] - 2017-08-23

- Fix 500 server error on enterprise course enrollment page.

9.901 [0.40.3] - 2017-08-21

- Change landing page course modal to use discovery api for populating course details.

9.902 [0.40.2] - 2017-08-16

- Increase capability and compatibility of Consent API.

9.903 [0.40.1] - 2017-08-11

- Add new unified DataSharingConsent model to the *consent* app.

9.904 [0.40.0] - 2017-08-08

- Add Enterprise API Gateway for new Enterprise Catalogs and Programs endpoints.
- Add /enterprise/api/v1/enterprise-catalogs/ endpoint.
- Add /enterprise/api/v1/enterprise-catalogs/{uuid}/ endpoint.
- Add /enterprise/api/v1/programs/{uuid}/ endpoint.

9.905 [0.39.9] - 2017-08-08

- Added management command "create_enterprise_course_enrollments" for missing enterprise course enrollments.

9.906 [0.39.8] - 2017-08-04

- Fixed session reset decorator bug.

9.907 [0.39.7] - 2017-08-04

- Make whether Enterprise Customers get data for audit track enrollments configurable.

9.908 [0.39.6] - 2017-08-02

- Fixed the text cutoff in the bottom of the course info overlay.

9.909 [0.39.5] - 2017-08-02

- Only send one completion status per enrollment for SAP SuccessFactors.

9.910 [0.39.4] - 2017-08-01

- Create Audit enrollment in E-Commerce system when user enrolls in the audit mode in enterprise landing page.

9.911 [0.39.3] - 2017-07-28

- Remove Macro use from swagger api config as it is not supported by AWS.

9.912 [0.39.2] - 2017-07-27

- Introduce new endpoint to the Enterprise API to query for courses by enterprise id.

9.913 [0.39.1] - 2017-07-27

- Ensure catalog courses API endpoint users are associated with an EnterpriseCustomer.

9.914 [0.39.0] - 2017-07-24

- Officially include Consent application by ensuring it is installable.

9.915 [0.38.7] - 2017-07-22

- Add a new Consent application.
- Add initial implementation of a generic Consent API.

9.916 [0.38.6] - 2017-07-21

- Remove SSO-related consent capabilities

9.917 [0.38.5] - 2017-07-19

- Add page_size in querystring and data mapping template to fix "next" and "previous" urls in API response.

9.918 [0.38.4] - 2017-07-18

- Fix DSC Policy Language Needs

9.919 [0.38.3] - 2017-07-14

- Fix dependency installation process in setup.py.

9.920 [0.38.2] - 2017-07-14

- Add consent declined message to course enrollment landing page.

9.921 [0.38.1] - 2017-07-13

- Remove requirement on too-new django-simple-history version
- Require slightly older django-config-models version

9.922 [0.38.0] - 2017-07-11

- Move to edx-platform release-focused testing
- Add Django 1.11 support in Hawthorn testing branch

9.923 [0.37.1] - 2017-07-11

- Update Enterprise landing page styling/language

9.924 [0.37.0] - 2017-07-06

- Update enterprise catalog api endpoint so that api returns paginated catalogs.

9.925 [0.36.11] - 2017-06-29

- Update DSC page language.

9.926 [0.36.10] - 2017-06-29

- Introducing SAP_USE_ENTERPRISE_ENROLLMENT_PAGE switch via django waffle.

9.927 [0.36.9] - 2017-06-28

- Refactor of automatic session termination logic.

9.928 [0.36.8] - 2017-06-28

- Enforce data sharing consent at login for SSO users only if data sharing consent is requested at login.

9.929 [0.36.7] - 2017-06-25

- UI tweaks to the enterprise landing page and course overview modal.

9.930 [0.36.6] - 2017-06-25

- Disable atomic transactions for CourseEnrollmentView to ensure that new EnterpriseCustomerUser records are saved to the database in time for ecommerce API calls.

9.931 [0.36.5] - 2017-06-23

- Apply automatic session termination logic to enterprise landing page based on enterprise customer configuration.

9.932 [0.36.4] - 2017-06-21

- Sort course modes in landing page.

9.933 [0.36.3] - 2017-06-21

- Fix for being unable to create course catalog clients due to upstream removal of the library.

9.934 [0.36.2] - 2017-06-21

- Add the ability to pass limit, offset and page_size parameters to enterprise catalog courses.

9.935 [0.36.1] - 2017-06-20

- Properly bump PyPI to latest changes from v0.36.0.

9.936 [0.36.0] - 2017-06-20

- Migrate from old, monolithic python-social-auth to latest, split version.
- Rework the NotConnectedToOpenEdX exception to be just one, and to say which method/dependency is missing.

9.937 [0.35.2] - 2017-06-20

- Fix Next and Previous page urls for enterprise catalog courses.

9.938 [0.35.1] - 2017-06-15

- Displayed course run price with entitlement on landing page and course information overlay

9.939 [0.35.0] - 2017-06-15

- Allow account-level data sharing consent in a course-specific context

9.940 [0.34.7] - 2017-06-14

- Enable "Continue" button flows on enterprise landing page

9.941 [0.34.6] - 2017-06-14

- Fixed layout of data sharing consent decline modal on mobile view

9.942 [0.34.5] - 2017-06-09

- Add Django 1.10 support back

9.943 [0.34.4] - 2017-06-09

- Added course information overlay

9.944 [0.34.3] - 2017-06-07

- Make enterprise landing page url available in the enterprise api and SAP course export.

9.945 [0.34.2] - 2017-06-06

- Fix UI issues (unexpected html escape) on enterprise landing page.

9.946 [0.34.1] - 2017-06-06

- Bug fix for Data sharing consent pop up page.

9.947 [0.34.0] - 2017-06-05

- Update data backing and behavior of enterprise landing page
- Fix template prioritization bug
- Fix URL rendering in enterprise login decorator

9.948 [0.33.24] - 2017-06-02

- UI updates for data sharing consent page.

9.949 [0.33.23] - 2017-06-02

- Fix a bug with unexpected image data in SAP course export job.

9.950 [0.33.22] - 2017-06-02

- Add an *EnterpriseApiClient* method for getting enrollment data about a single user+course pair
- Add logic to enterprise landing page that redirects users to the course when already registered

9.951 [0.33.21] - 2017-06-01

- UI updates for course mode selection in enterprise landing page.

9.952 [0.33.20] - 2017-05-23

- Migrate from mako templates to django templates

9.953 [0.33.19] - 2017-05-18

- Display account created/linked messages on enterprise landing page

9.954 [0.33.18] - 2017-05-17

- Add Enable audit enrollment flag

9.955 [0.33.17] - 2017-05-16

- Add django admin for enterprise course enrollment models

9.956 [0.33.16] - 2017-05-15

- Bug fixes for SAP learner completion data passback.

9.957 [0.33.15] - 2017-05-10

- Additional minor UI updates for enterprise landing page.

9.958 [0.33.14] - 2017-05-10

- Add new externally managed consent option for enterprise customers.

9.959 [0.33.13] - 2017-05-09

- Fix invalid API Gateway URIs

9.960 [0.33.12] - 2017-05-03

- Add enterprise landing page

9.961 [0.33.11] - 2017-05-02

- Add tpa hint if available for launchURLs for SAP Course metadata push.

9.962 [0.33.10] - 2017-05-02

- Fix bug with inactivating SAP courses that are no longer in the catalog.

9.963 [0.33.9] - 2017-04-26

- Fix enterprise logo validation message for max image size limit

9.964 [0.33.8] - 2017-04-26

- Updated calls to get_edx_api_data as its signature has changed in openedx.

9.965 [0.33.7] - 2017-04-24

- Redirect to login instead of raising Http404 if EnterpriseCustomer missing.
- Add confirmation_alert_prompt_warning to context of account-level consent view.

9.966 [0.33.6] - 2017-04-21

- Increase max size limit for enterprise logo

9.967 [0.33.5] - 2017-04-20

- Added vertical hanging indent mode to isort settings and adjusted current imports

9.968 [0.33.4] - 2017-04-18

- Enforce login for course-specific data sharing consent views.

9.969 [0.33.3] - 2017-04-18

- Fixed the CSS for the expand arrow in the data sharing consent page.

9.970 [0.33.2] - 2017-04-17

- Update Data Sharing Consent message.

9.971 [0.33.1] - 2017-04-17

- Order enterprise customers by name on enterprise customer django admin

9.972 [0.33.0] - 2017-04-11

- Improve accounting for inactive courses for SAP course export.

9.973 [0.32.1] - 2017-04-06

- Bug Fix: Added Handling for user enrollment to courses that do not have a start date.

9.974 [0.32.0] - 2017-04-06

- Refine SAP course export parameters

9.975 [0.31.4] - 2017-04-05

- Added missing migration file for recent string updates

9.976 [0.31.3] - 2017-04-04

- Modified SAP completion status data to correctly indicate a failing grade to SAP systems.

9.977 [0.31.2] - 2017-04-03

- Bugfix: Resolve IntegrityError getting raised while linking existing enterprise users when data sharing consent is disabled for the related enterprise.

9.978 [0.31.1] - 2017-03-31

- Bugfix: Allow unlinking of enterprise learners with plus sign or certain other characters in email address.

9.979 [0.31.0] - 2017-03-30

- Edited UI and error strings.

9.980 [0.30.0] - 2017-03-27

- Fully implements sap_success_factors transmitters and client to communicate with the SAP SuccessFactors API, and to handle auditing and other business logic for both catalog and learner data calls.

9.981 [0.29.1] - 2017-03-27

- Support for segment.io events on data sharing consent flow

9.982 [0.29.0] - 2017-03-23

- Updates integrated_channels management command *transmit_learner_data* to support sending completion data for self-paced courses, and to use the Certificates API for instructor-paced courses.

9.983 [0.28.0] - 2017-03-23

- New data sharing consent view supporting failure_url parameter

9.984 [0.27.6] - 2017-03-21

- Removed OAuth2Authentication class from API viewset definitions

9.985 [0.27.5] - 2017-03-17

- Updated api.yaml to resolve swagger configuration issues.

9.986 [0.27.4] - 2017-03-17

- Allows enterprise enrollments to be made on servers that sit behind a load balancer.

9.987 [0.27.3] - 2017-03-16

- Added integrated_channels management command to transmit courseware metadata to SAP SuccessFactors.

9.988 [0.27.2] - 2017-03-10

- Added integrated_channels management command to transmit learner completion data to SAP SuccessFactors.

9.989 [0.27.1] - 2017-03-13

- Added api.yaml and api-compact.yaml files to introduce api endpoints for catalog api-manager.

9.990 [0.27.0] - 2017-03-02

- Added API endpoint for fetching catalogs and catalog courses.

9.991 [0.26.3] - 2017-03-02

- Added integrated_channels to MANIFEST.in to properly include migrations for the new packages.

9.992 [0.26.2] - 2017-03-02

- Fixed package listing in setup.py to avoid import errors when using as a library

9.993 [0.26.1] - 2017-02-28

- Added support for retrieving access token from SAP SuccessFactors
- Added indicator in Sap SuccessFactors admin tool for checking the configuration's access to SuccessFactors.

9.994 [0.26.0] - 2017-02-28

- Formally introducing new integrated_channels apps
- Adding new models and admin interfaces for integrated_channel and sap_success_factors

9.995 [0.25.0] - 2017-02-28

- Refactor _enroll_users() method to pay down technical debt
- Improve admin messaging around enrollment actions

9.996 [0.24.0] - 2017-02-27

- API for SSO pipeline is simplified to a single element.
- SSO users are linked to relevant Enterprise Customer when data sharing consent is disabled.

9.997 [0.23.2] - 2017-02-22

- SSO users are not created as EnterpriseCustomerUsers until all consent requirements have been fulfilled.

9.998 [0.22.1] - 2017-02-20

- Course Catalog API degrades gracefully in absence of Course Catalog service.

9.999 [0.22.0] - 2017-02-14

- Added API endpoint for fetching entitlements available to an enterprise learner

9.1000 [0.21.2] - 2017-02-07

- Add id in EnterpriseCustomerUserSerializer fields

9.1001 [0.21.0] - 2017-01-30

- Add UI handling for course-specific data sharing consent

9.1002 [0.20.0] - 2017-01-30

- Add ability to select existing learners to be enrolled in courses from admin

9.1003 [0.19.1] - 2017-01-30

- Resolved conflicting urls for User API endpoint.

9.1004 [0.19.0] - 2017-01-30

- Added read-only enterprise API endpoint for IDAs.
- Moved utility functions from api.py to utils.py

9.1005 [0.18.0] - 2017-01-27

- Add the ability to notify manually-enrolled learners via email.

9.1006 [0.17.0] - 2017-01-25

- Add the EnterpriseCourseEnrollment model and related methods

9.1007 [0.16.0] - 2017-01-25

- Fix a bug preventing a course catalog from being unlinked from an EnterpriseCustomer

9.1008 [0.15.0] - 2017-01-25

- Enroll users in a program.

9.1009 [0.14.0] - 2017-01-20

- Added view of seat entitlements on enterprise admin screen

9.1010 [0.13.0] - 2017-01-06

- Dynamically fetch available course modes in the Manage learners admin

9.1011 [0.12.0] - 2017-01-05

- Create pending enrollment for users who don't yet have an account.

9.1012 [0.11.0] - 2017-01-05

- Added links from the Manage Learners admin panel to individual learners.

9.1013 [0.10.0] - 2017-01-04

- Added the ability to search the Manage Learners admin panel by username and email address.

9.1014 [0.9.0] - 2016-12-29

- In django admin page for enterprise customer added alphabetical ordering for catalog drop down and displayed catalog details link next to selected catalog.

9.1015 [0.8.0] - 2016-12-08

- added the branding information api methods to return the enterprise customer logo on the basis of provider_id or uuid.
- Updated the logo image validator to take an image of size maximum of 4kb.

9.1016 [0.7.0] - 2016-12-07

- Added a feature to enroll users in a course while linking them to an enterprise customer.

9.1017 [0.6.0] - 2016-12-04

- Fixed EnterpriseCustomer form to make Catalog field optional
- Added user bulk linking option
- Added Data Sharing Consent feature

9.1018 [0.5.0] - 2016-11-28

- Added checks to make sure enterprise customer and identity provider has one-to-one relation.
- Added a helper method to retrieve enterprise customer branding information

9.1019 [0.4.1] - 2016-11-24

- Fixed User.post_save handler causing initial migrations to fail

9.1020 [0.4.0] - 2016-11-21

- Set up logic to call course catalog API to retrieve catalog listing to attach to EnterpriseCustomer.

9.1021 [0.3.1] - 2016-11-21

- Fixed missing migration.

9.1022 [0.3.0] - 2016-11-16

9.1022.1 Added

- Added Pending Enterprise Customer User model - keeps track of user email linked to Enterprise Customer, but not yet used by any user.
- Added custom "Manage Learners" admin view.

9.1022.2 Technical features

- Added sphinx-napoleon plugin to support rendering Google Style docstrings into documentation properly (i.e. make it recognize function arguments, returns etc.)
- Added translation files

9.1023 [0.2.0] - 2016-11-15

- Linked EnterpriseCustomer model to Identity Provider model

9.1024 [0.1.2] - 2016-11-04

- Linked EnterpriseCustomer model to django Site model

9.1025 [0.1.1] - 2016-11-03

- Enterprise Customer Branding Model and Django admin integration

9.1026 [0.1.0] - 2016-10-13

- First release on PyPI.
- Models and Django admin integration

CHAPTER

TEN

MULTIPLE ENTERPRISE SUPPORT

10.1 Status

Accepted

10.2 Context

There is a business need to be able to support a learner being linked to multiple enterprises. Today, edX workflows only support a 1-to-1 linkage and our system does not prevent linking multiple Enterprises to a single learner. Hence not all workflows work as expected.

Some examples of these errant workflows are:

1. When a learner is linked with an enterprise and tries to link with another enterprise(through SSO), the learner is redirected to the login page instead of the registration page for the second enterprise. This is because the system detects that these credentials are already in the system.
2. Enrollment, Progression, Completion, Consent records become ambiguous if a learner completes a course from enterprise A and then attempts the same course through enterprise B.
3. Incorrect discounts are applied when an employee linked with multiple enterprises redeems the benefit applicable only to one of the enterprises.

10.3 Decisions

Short term updates:

As part of the login flow the user would select one of the linked enterprises. This selection would only happen once, after user authentication in the login flow.

Long term updates:

An account/profile switcher like we have in New Relic, Google, etc. with a flow similar to the following could be Implemented.

1. User flow will allow the learner to login without any enterprise related barriers.
2. After login, user is reminded that they are linked to multiple Enterprises and that they need to select the "profile" they intend to use during this logged-in session before executing any enrollment or other transaction in the system.
3. The enterprise "profile" would be specified using a "switch enterprise" page.

4. If a learner logs in via SSO with account-A, but intends to proceed using account-B, they would be logged out so that they can then re-login via account-B.
5. The text on the new "switch enterprise" page makes the above user-flow and selection as clear as possible for the learner.

10.4 Consequences

1. Discounts applied correctly for users connected to multiple enterprises.
2. Unambiguous transactional data history for enterprise users.

CHAPTER

ELEVEN

NEW ENTERPRISE CATALOG IDA

11.1 Status

Draft

11.2 Context

We are interested in improving the reliability and performance of our system at large, and we have identified that the enterprise catalog api and related functionality are a central dependency that often causes issues, such as users experiencing random failures during enrollment or other workflows. Almost every feature that our users interact with depends on enterprise catalogs: Enrollment, Code Redemption, Data Sharing Consent, Data transmission to external LMS systems, and API access given to select customers.

Our main goals are as follows:

1. Get the response time of the enterprise customer catalog endpoints to be below 1 second for 99% of calls.

2. Since many features do not hit the enterprise catalog api endpoints but perform the same calculations directly in code, we would also like to improve the performance of those code paths and drastically reduce the calls to the discovery service. Examples include the data sharing consent workflow, the enterprise enrollment api, and the task that "catches" enterprise enrollments from the B2C site.

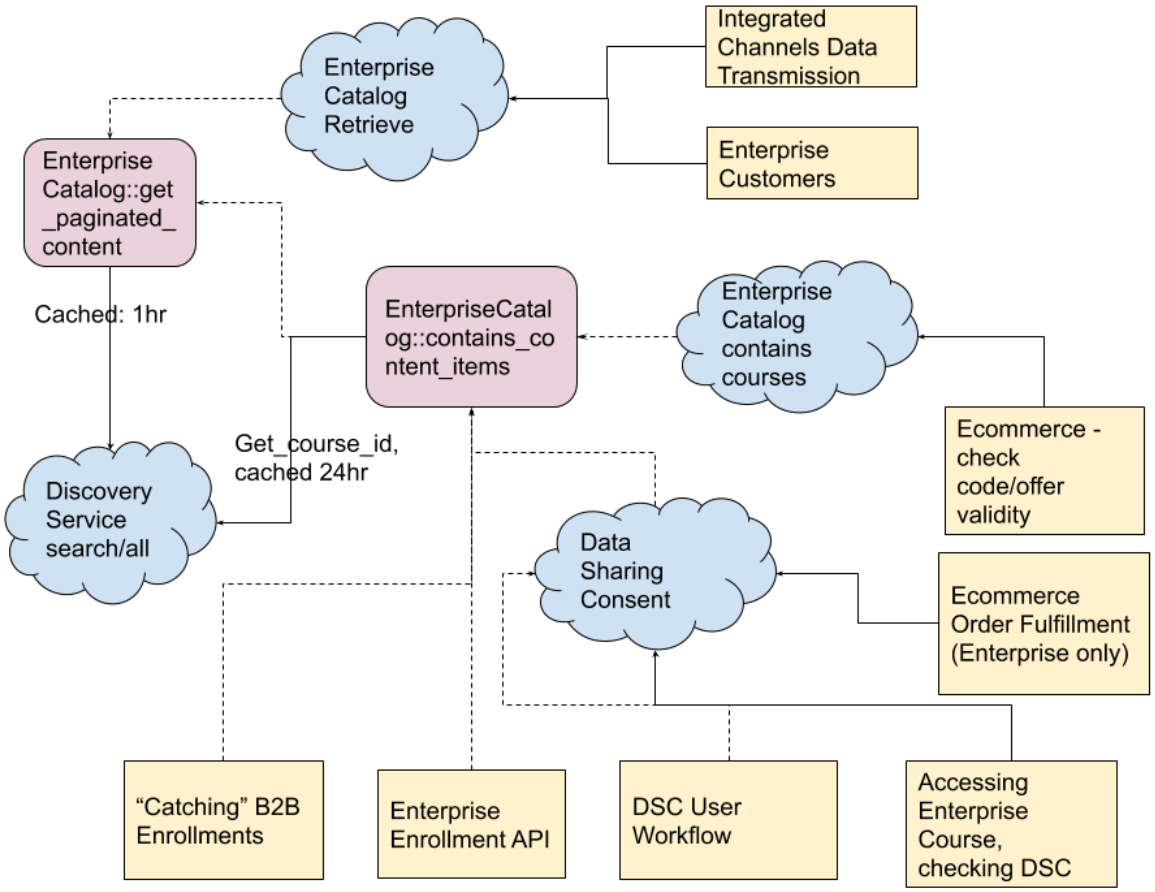
11.2.1 What is Enterprise Catalog?

Enterprise catalogs are essentially saved elasticsearch queries attached to an Enterprise Customer. There can be many catalogs configured for a given customer, which enables different sets of learners from the same Enterprise to be given access to different sets of courses, and allows different discounts to be configured for different sets of courses.

We expose two apis that are frequently used by various enterprise features: one to retrieve all of the content metadata that would be returned by the configured catalog query, and one to determine if a given course id or program id is part of the set of content returned by the catalog query. These apis exist both as python apis and REST apis that can be hit from within our edX ecosystem and externally from customers. They are wrappers around the search/all endpoint in the discovery service. Although there is caching in place, the calls to discovery are implemented synchronously, and we've found that half the time requests do not hit the cache, and thus the most of the time is spent waiting on the discovery service to return results.

11.2.2 Who's Calling What now?

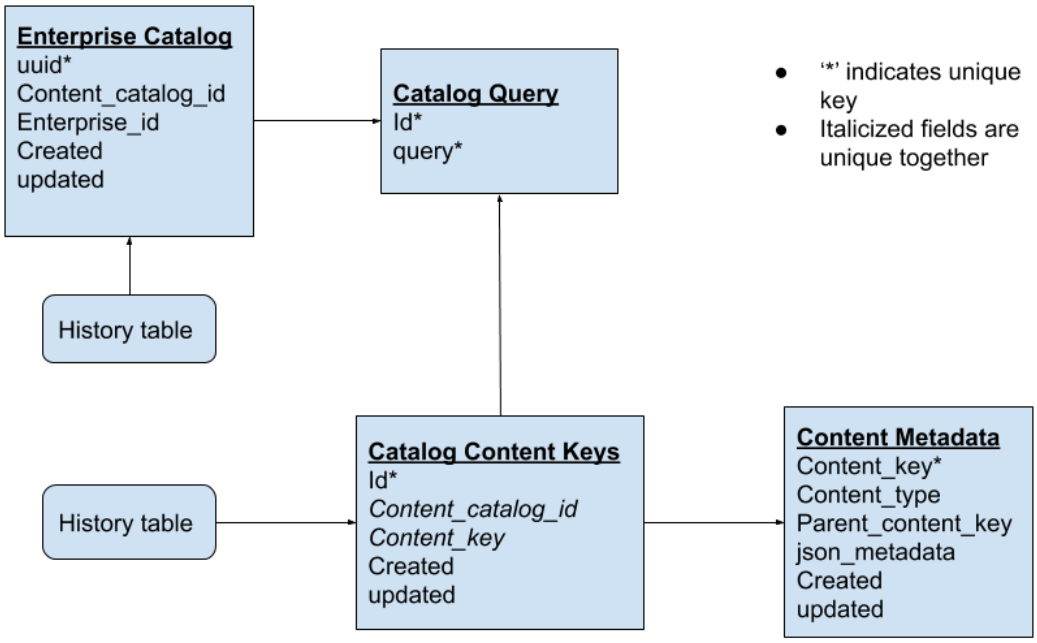
Here is a diagram showing a simplified view of the calls to these apis. The yellow boxes represent the features or user workflows that trigger calls to the various apis. The blue clouds represent REST apis, and the purple rounded boxes represent python code apis. Solid lines are synchronous api calls and dotted lines are code apis (not hitting the network).



11.3 Decisions

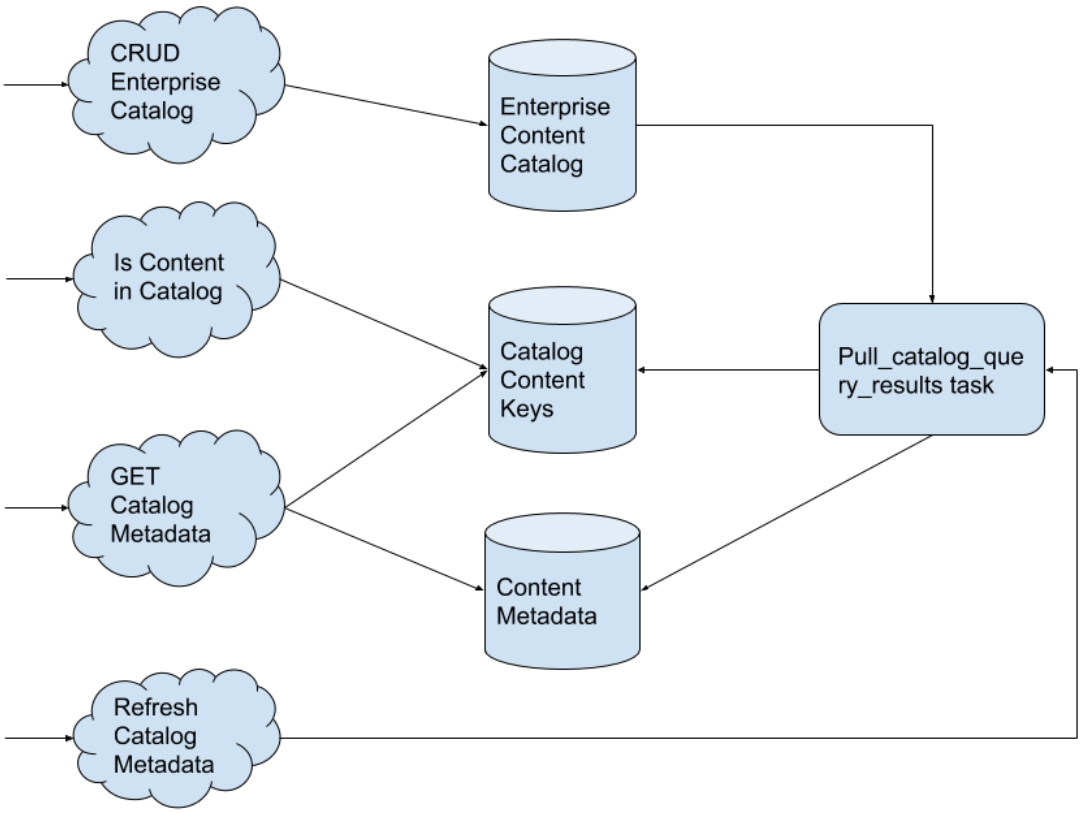
In order to address these performance issues and to more effectively scale and build new features related to enterprise catalogs in the future, we will create a new IDA (Independently Deployable Application) to replace the existing enteprise catalog functionality. This IDA will support the existing API contracts established in the edx-enterprise API, but will be a new implementation separate from what we do today.

The microservice will be supported by the following relational data model:



The microservice will expose the following APIs for use:

- CRUD Operations for Enterprise Catalog
- is content id in catalog: Given a course, course run, program, or other content id, and an enterprise catalog, returns true if the content id is contained within the catalog's current results.
- get_catalog_metadata: Given an enterprise catalog, return the full content metadata results for content currently in the catalog.
- pull_catalog_query_results: Given an enterprise catalog, kick off a task to get the freshest content metadata for that catalog.



All metadata will be fetched from the discovery service through asynchronous tasks that are entirely decoupled from when a call is made to get metadata for a particular catalog. We will guarantee that catalog data is fresh within the past 24 hours. Catalog refreshes will be performed through asynchronous tasks whenever a catalog is created or updated, when it is manually refreshed, or on a daily cadence. We will also gain efficiencies by deduping elasticsearch queries across different enterprise catalogs so that catalogs with the same query only fetch data from discovery once.

We will maintain the existing api signature so that the only update current api clients will have to make is to the url; request and response data will maintain the same format.

We will transition all internal api calls (both code and network calls) to use this new service. Once this transition is complete, we will deprecate and remove the original functionality contained in this repo.

11.4 Consequences

Transitioning to this new service will require careful consideration, since many internal systems as well as external clients rely on this API. Network calls should be simple enough to swap out, but within edx-enterprise there are many code api calls to enterprise catalogs, and these will be tougher to transition. We will need to be mindful of how we are caching data from this new service throughout our system. Hopefully, it can be kept to a minimum because this api should be much more performant.

This will also be the first enterprise specific IDA, which will require us to gain an understanding and ownership of a service from end to end. This should include monitoring and alerting SLAs that we expect the service to maintain. Hopefully this will allow us to move towards extracting more functionality in this repo out to microservices.

CHAPTER

TWELVE

CANVAS INTEGRATION VIA CLASSIC INTEGRATION MODEL VS LTI

12.1 Status

Accepted

12.2 Context

We would like to capture points about the decision to choose the Class integration model over LTI model, in order to make edX courses available in Canvas LMS for partner institutions.

12.3 Decision

For Canvas integration, these following use cases are prioritized:

- Students can easily discover courses within their institution's online learning experience
- Faculty can easily view student performance data within their institution's online learning experience

There is another use case that is deemed not applicable w.r.t Canvas integration:

- Students can consume course content from within their institution's online learning experience

The classic integration model nicely supports the former use cases and allows students to take courses on the edX platform once discovered in Canvas.

Furthermore, LTI does not support course discovery, which is essential to independent and facilitated learning use-cases. hence we made a decision to choose the classic integration model.

12.4 Consequences

Choosing classic over LTI does mean that the "Digital Textbook" use case will be harder/clunkier to implement. LTI does very well as this. This factor may be re-considered in the future.

CHAPTER

THIRTEEN

ENTERPRISE ROLE ASSIGNMENTS WILL RECORD A CUSTOMER UUID

13.1 Status

Accepted - January 2021

13.2 Context

The `SystemWideEnterpriseUserRoleAssignment` model is used to grant role-based permissions to enterprise users. These permissions generally control the users' access to data and ability to perform certain functions within an enterprise customer. This assignment model does *not* currently refer to any specific customer - the "context" to which a role is applied is inferred from a user (an `auth.User` record) having been "linked" to a customer via the `EnterpriseCustomerUser` model.

However, a single `auth.User` may be linked to multiple `EnterpriseCustomers`, and if that user is assigned a role (such as `enterprise_admin`), it is inferred that the role assignment applies to **every** customer to which the user is linked. This is undesirable; we generally do not want an admin of *Customer A* to also have admin permissions for *Customer B* (to which the user may be linked as a mere `learner`). Instead, the `SystemWideEnterpriseUserRoleAssignment` model should explicitly record *which* customer the assigned user has the given role in.

13.3 Decisions (interspersed with Consequences)

13.3.1 We'll add an `enterprise_customer` field to the assignment model

- It will be an optional field. It should be a foreign key on `EnterpriseCustomer.uuid`.
- Via an edx-rbac change, we will also add a boolean `applies_to_all_contexts` field, which is a wildcard that, if true, indicates the user has the role assigned for every customer. It should default to `false`.
- The semantics of `applies_to_all_contexts` imply that `enterprise_customer` will sometimes be null. For example, a user assigned the `enterprise_openedx_operator` role will have `applies_to_all_contexts` set to `true` and an `enterprise_customer` as `null`.
- We will add a customer validator to enforce that the `enterprise_customer` field can be null only when `applies_to_all_contexts` is true.

13.3.2 We'll backfill the values of this field

- We'll make the current, implicit assignments (that is, if a role is assigned to a user, that role applies to the user in every customer to which the user is linked) explicit by filling in the UUID value of the linked enterprise customer.
- For `enterprise_openedx_operator` roles, the `applies_to_all_contexts` field will be marked true.

13.3.3 We'll do a hand-audit of enterprise_admin role assignments

- For every user who is effectively an admin of two different enterprise customers, we'll have to manually determine (with help from our support team) which of the two customers they are actually an admin of, and which they are simply a learner within (unless they're truly an admin of both).

13.3.4 The context of a system-wide enterprise assignment is determined from `enterprise_customer`

- This is true for every role.
- If `applies_to_all` is true, `get_context()` should return the wildcard context token "*".
- The code that populates the user JWTs with (role name, customer UUID) pairs shouldn't have to change - that code should only be looking at the `get_context()` method of the role assignment class.

13.3.5 Admin forms must record a value for `enterprise_customer`

- It would be nice to "auto-populate" it somehow if the user is linked to only one enterprise customer.

13.3.6 This field will be populated by signal handlers related to enterprise user CRUD operations

- Every user will get an `enterprise_learner` role assignment for the given enterprise.
- Pending admin "conversion" will result in an `enterprise_admin` role assignment for the given enterprise.
- We'll delete the role assignment when an `EnterpriseCustomerUser` is deactivated.

CHAPTER

FOURTEEN

BULK ENROLLMENT ENDPOINT FOR SUBSCRIPTION LEARNERS

14.1 Status

Accepted

14.2 Context

Business need: The admin portal feature to enroll multiple learners in multiple subscription-courses needs a suitable backend.

Various enrollment apis exist, but they do not cater to or optimize for multiple enrollments.

There is an *EnrollmentApiClient* which has an *enroll_user_in_course()* method used by:

- models::EnterpriseCustomerUser::enroll()
- utils::enroll_user() (handles multiple courses)
- views::CourseEnrollmentView::post()
- views::HandleConsentEnrollment::get()
- views::GrantDataSharingPermission::_enroll_learner_in_course()

There is also a method to create pending user enrollments for non-edX users at: *models::EnterpriseCustomer::enroll_user_pending_registration()*

This method also handles multiple courses.

Due to the scale and volume considerations, an api endpoint is needed that caters to the bulk use case without hitting rate limiting issues or server overload.

14.3 Decisions

- We will add a new endpoint to edx-enterprise to handle bulk enrollment in a single request.
- For existing edX users, we will reuse the *utils::enroll_user()* method. This method handles multiple courses already, but still calls the *EnrollmentApiClient* for now. There is a cleanup effort anticipated to replace the HttpClient calls embedded in the *utils::enroll_user()* method with direct database call utilties in the edx-platform Django application. Such a cleanup is not in scope for this feature work.
- For emails not matching existing users in edX LMS Django application, we will use the *models::EnterpriseCustomer::enroll_user_pending_registration()* method

- We will create an ecommerce order for each successful enrollment of an existing edX user. For this we will use the *EcommerceApiClient::create_manual_enrollment_orders()* method which creates an order for each enrollment passed.
- Any failures in individual enrollments will not cause failures of the entire batch. In other words, the endpoint will be non transactional. This avoid wasteful and confusing workflow for the client of the api, by performing a 'best effort' enrollment and reporting the failures.
- We will not have MFEs call this endpoint directly. Rather they will call a license-manager endpoint which will call this endpoint. There are several reasons for this: #. Only licensed learners should be enrolled in the subscription's courses. This design will guard against any frontend requests containing non-desired learners. #. Calling license-manager from edx-enterprise requires edx-enterprise to know about license-manager which is counter to the archiectural design for edx-enterprise #. Finally, semantically it makes sense to request a license management service to authorize license based enrollments.
- The edx-enterprise endpoint will return a response with these pieces of information: *{'successes': [], 'pending': [], 'failures': []}*. Note, pending enrollments are supported. This will allow the license-manager to inform the user if any enrollments failed, while successfully (meaning: success or pending) enrolling the rest, in a single request cycle.
- The endpoint will be safe to invoke with pre-existing enrollment pairs (meaning learner email + course_id). These enrollments will also be added to 'successes' in the response.
- The endpoint will also support a boolean input to notify learners of enrollments (or not) by email

14.4 Consequences

1. MFEs can make a single reliable request for bulk enrollment
2. Spurious enrollments of non-eligible learners is avoided
3. Partially enrolling 'what it can' and recording the rest in a response, allows for a better user experience for admins and the use of a single coherent endpoint.

CHAPTER

FIFTEEN

ADDING ENTERPRISE_UUID IN COURSE ENROLLMENT EVENT

15.1 Status

Accepted

15.2 Context

`event_routing_backends` application needs `enterprise_uuid` in order to send transformed enrollment events to the relevant enterprise customers. At present, `enterprise_uuid` is not included in the course enrollment events generated by edx enterprise.

15.3 Decision

`enterprise_uuid` will be added in the course enrollment event generated by edx enterprise and will be added to the context of `EVENT_NAME_ENROLLMENT_ACTIVATED` emitted by edx platform.

15.4 Consequences

`enterprise_uuid` will be present in the context of `EVENT_NAME_ENROLLMENT_ACTIVATED` only if the user has provided an enterprise url during enrollment.

CHAPTER

SIXTEEN

TRANSMITTING ONLY UPDATES IDA

16.1 Status

Draft

16.2 Context

The transmit_content_metadata jenkins job (along with the underlying integrated_channels.transmit_content_metadata() task and export() method) makes an uncomfortably large number of calls to the the enterprise-catalog service's /api/v1/enterprise-catalog/<uuid>/get_content_metadata endpoint with a relatively low number of customers which results in an unacceptably high database load.

At the same time the data being requested by this frequently-running job is often unchanged between runs - content metadata tends to change in enterprise-catalog roughly once/day.

Our main goals are as follows:

1. Reduce the overall number of calls required by the Integrated Channels in order to retrieve the content metadata for any particular transmission.

2. Prevent the Integrated Channels from making an abundance of requests to the enterprise catalog if there is no update needed (ie if nothing in the catalog has changed)

16.2.1 What is the transmit_content_metadata job and underlying export method?

transmit_content_metadata is one of the three core tasks of the Integrated Channels, the other two being transmit_learner_data and transmit_subsection_learner_data. Specifically transmit_content_metadata's job is to gather course metadata contained within the customer's enterprise catalog and transmit create, update and deletes of course entities, through customer configurations, to the external LMS'.

The *export()* method is the main method responsible for fetching said content metadata. It will take all listed *catalogs to transmit* under the customer's configurations (otherwise will default to all the customer's catalogs if *catalogs to transmit* is not provided) and request content metadata from the enterprise catalog. These catalogs can range from "all possible content" to a specific, single course.

16.3 Decisions

In order to address these performance issues and to more effectively handle onboarding more customers to the Integrated Channels, we have decided to address requests to the enterprise catalog on two fronts:

1. Increase the page size of requests going to */api/v1/enterprise-catalog/<uuid>/get_content_metadata* from a base of 10 to 100. This will reduce the number of calls necessary to retrieve the entire catalog from 150~ to about 15 (as of 07/2021).

2. Add an additional request/check to the Integrated Channel's exporter that will retrieve the last time either a catalog or the associated content metadata of a catalog has been changed (whichever is most recent). This timestamp will be compared to the *content_last_changed* field of the ContentMetadataItemTransmission object, which is the *catalog last modified* value retrieved from the customer's most recent successful transmission. If there is no updated needed, then the exporter will refrain from querying enterprise catalog for the catalog's metadata.

16.4 Exceptions

1. The Cornerstone (CSOD) LMS Integration will not use the 'transmit only updates' rule."
2. CSOD related transmissions will record ContentMetadataItemTransmission objects, same as all other channels.

16.5 Consequences

The transmit-content-metadata jenkins job can continue to run on a frequent schedule without having to worry about unacceptably large database loads, as it will record when updates are needed and will only request data when necessary. If any metadata associated with a catalog has changed between job run N and N+1, then we'll transmit the full catalog to the integrated channel during run N+1. It should be noted that there could be potential further improvements to the entire export process, as currently we are requesting the entire catalog and not only updated content. Reducing requests to only the updated metadata could further reduce database loads.

CHAPTER

SEVENTEEN

DETERMINING COURSE COMPLETION FOR VARIOUS COURSE MODES

17.1 Status

Draft

17.2 Context

LMS integrations have a learner_data module that reports out (to external LMS systems), a field (boolean) called *course_completed*, along with any available grading data.

For Non Audit-mode enrollments: the value *completed_date* is used as an indication of course completion. This is deterministic for courses that issue certificates or grading info

However, for Audit mode enrollments, there is a case where the learner has already finished all non-gated content. But in this state, *course_completed* will never be true for these learners who never upgrade to verified, for example. Also there is no more content learner can do anything with. So in this sense, they are done with the course.

Also, these learners don't always have access to all the content. So we cannot rely on certificate or grade information to detect if they are 'done' with a course.

In fact there is no one deterministic definition of course_completed for audit track.

Therefore, for LMS integration customers to get a sense of how audit learners are doing, with completion, we need to offer a best approximation sense of *course_completed*.

17.3 Decisions

Integrated channels will determine the *course_completed* for each enrollment with the following logic:

```
if audit_enrollment:
    Transmit course as complete for this learner, if no non-gated content is remaining␣
↪to be finished
else:
    Transmit course as complete if certificate or grading data indicates completion_date␣
↪is available.
```

17.4 Consequences

We will have more accurate reporting of completion of course (in some sense) for audit track. There is no precise way to define that a learner is done with a course unless we are issuing a certificate. Therefore this best approximation is being used for LMS integrations.

CHAPTER

EIGHTEEN

ADDING COURSE KEY - UUID MAPPING FOR CORNERSTONE

18.1 Status

Draft

18.2 Context

At one point we were having transmitted courses to CSOD tenants fail with invalid chars contained within course metadata. To combat this, we decided to encode the course keys. This has resulted in some of the courses failing validation with a "course key too long" failures. There's a 50 char limit to course keys on the CSOD lms, so we need to limit the size of our encoded course keys within the content metadata.

Simply truncating it would not allow for backwards compatibility or ensuring uniqueness. We also considered encoding and then compress, but while this would most likely work for every course key (the largest length found in prod was 54 chars), it can't properly guarantee the key to be under 50 chars, and we would have to worry with hashing the fact that it is one way.

18.3 Decisions

Using uuids as the course key as they are guaranteed to be 36 characters and don't have to be url encoded. We will be creating a dictionary-like table to map the edx course key to a key that will be used as the course key in Cornerstone.

Since we already have many courses in Cornerstone that the keys are under 50 characters, we will not be creating new uuids in the dictionary, because that could lead to a course being duplicated on their system. The logic that is currently implemented (using url encoded course key) will remain, unless the encoding is over 50 characters. The uuid will be generated when we are transmitting course content metadata, and also when they report learner data back to us.

We also started an exploration of whether there are existing uuids we can use to map to course keys. We found that there are uuids in Discovery for courses: https://github.com/edx/course-discovery/blob/master/course_discovery/apps/course_metadata/models.py#L768 and courseruns: https://github.com/edx/course-discovery/blob/master/course_discovery/apps/course_metadata/models.py#L1277 but we are still in the stages of finding out whether they are stable throughout their lifespan to be used. Turns out only courses with 'published' state in discovery have stable uuids. Because there are some unknowns there, this fix will allow us a more immediate remediation.

18.4 Consequences

As always, adding an additional table leads to more memory storage, and have a customized approach to Cornerstone might lead to some confusion down the line. Code will be properly documented and lead back to this ADR to alleviate some of this.

CHAPTER

NINETEEN

ZERO STATE BROWSING WITH UNIVERSAL LINK

19.1 Status

DRAFT

19.2 Context

Today, the only way to access our learner portal is by explicit, named invitation from an administrator sent from the admin portal. The invitation can only be sent while assigning a subsidy (code or license). Our cost-conscious admins need a way to publicize their catalog and gauge interest before assigning a subsidy (spend).

We want to allow admins to generate a universal link that can be shared with their enterprise. Users visiting the universal link would be able to browse the enterprise catalog and request course access.

We will take a two-phase approach for the implementation of universal links.

19.2.1 Phase 1

Users visiting the link would be able to logistrate and be automatically associated with the enterprise. Users linked to an enterprise through a universal link will have no subsidies assigned and be in the "zero state". Users in the "zero state" are shown an aggregate of all the catalogs for an enterprise on the search page and have access to all the tools that do not require a subsidy. They would then be able to browse the offerings and request course access.

19.2.2 Phase 2

We want to make enterprise catalogs publicly accessible through a universal link so that users who don't want to logistrate can still view a catalog. Users visiting this link would be able to browse the course offerings and request access. Upon course request, users will logistrate and be redirected to the learner portal course access request flow. Note that a user must be authenticated and associated with an enterprise to request course access.

19.3 Decision

19.4 Phase 1 - Browsing via the learner portal search page

Currently a `PendingEnterpriseCustomerUser` is created whenever a subsidy is assigned that associates a user to an enterprise through email. Upon logistration, we check for for any `PendingEnterpriseCustomerUser` and links a user by replacing `PendingEnterpriseCustomerUser` with an `EnterpriseCustomerUser` and assigning the *enterprise_learner* role.

A user has to be associated with an enterprise to have the correct permissions when accessing our APIs. We have to link a user to an enterprise in order to support browsing without a subsidy. This can be accomplished by exposing a secured endpoint that links a user to an enterprise which gets called by the learner portal. A user linked this way will be in the "zero state" by default and be able to browse the learner portal freely.

A new endpoint `POST /link-enterprise/` that links a user and an enterprise will be created. The payload will contain `enterprise_customer_uuid`, `user_id`, and `enterprise_customer_key` - a UUID generated for the enterprise A new model `EnterpriseCustomerInviteKey` will be created to to stores these keys.

`EnterpriseCustomerInviteKey` contains the following fields * uuid - key used to generate the universal link/proof that a user had the universal link * enterprise_customer_uuid - uuid of the enterprise * usage_limit - the number of times a key can be used to link users * expiry_date - defaults to the end date of the enterprise's subscription plan * timestamps

CRUD endpoints for creating/deleteing an `EnterpriseCustomerInviteKey` model will also be created.

A universal link can be generated by adding the key as a query param to the url that points to the learner portal for an enterprise, i.e. `/?enterprise_customer_key=key`. An enterprise admin will be able to generate/view links through the admin portal by calling `POST /EnterpriseCustomerInviteKey/` to generate a new key. Note that only enterprise admins should be able to generate a key, and only for enterprises they are an admin of. The generated link could also include the enterprise slug, but relying on the `enterprise_customer_key` instead to direct the user to the correct enterprise means that the link would still work if an enterprise changes its slug.

A user will be prompted to logistrate when visiting the link if not already logged in and be redirected to the learner portal afterwards. Currently, a not found page is shown on the learner portal if the user is not linked. If the user is not linked and an `enterprise_customer_key` is found in the query params, the portal will call the new endpoint to link the user. If the call succeeds, the portal will reload and the user will be able to browse as usual. If the call fails, appropriate messaging regarding the failure will be shown to the user. This will also link users that were already logged in before visiting the link.

19.4.1 Limiting logistration through universal link

Any authenticated user with a universal link that contains a valid key is able to link themselves with an enterprise. We want to take necessary precautions to prevent abuse of this feature. *EnterpriseCustomerInviteKey* will be expirable and have a limit on the number of times it can be used. We will validate the key (not expired and usage limit has not been reached) when linking a user to an enterprise. If the key is invalid, the user will not be linked and be informed that the link is invalid. We will also validate that the `enterprise_customer_uuid` in the request payload matches with the `enterprise_customer_uuid` associated with the key.

19.4.2 Limiting number of course requests to prevent email spamming

In the future we want to limit the number of course requests to prevent admins from being spammed with request notifications. This should be factored in when implementing course request and will not be addressed here.

19.4.3 Expiring a universal link

A universal link can have a lifetime dictated by the expiry date set on the `EnterpriseCustomerInviteKey` used in the link. A key will not be valid if expired. An enterprise admin should be able to manually expire a key to revoke a link.

19.4.4 Analytics

Segment events will be fired when * an `enterprise_customer_key` is created - i.e. 'edx.bi.user.enterprise.key.created' * a user is linked to an enterprise using the universal link, include key used in the event - 'edx.bi.user.enterprise.key.used' * a key expires or reaches its usage cap - 'edx.bi.user.enterprise.key.invalidated' * a user attempts to use a bad key - 'edx.bi.user.enterprise.key.attempted'

We will also keep track of which user were linked using a universal link in the model (possibily a new field on `EnterpriseCustomerUser`) and include that information in future tracking events, i.e. search events in the learner portal.

We also want to track how many users land on the logistration page but abandons the page. This could possibly be done through google analytics.

19.4.5 Consequences

- After a user requests course access and is granted a subsidy(ies), the learner portal will only display offerings associated with the subsidy(ies).
- The JWT token will need to be refreshed to include the `enterprise_learner` role after a user has been linked. The learner portal has logic to refresh the token if there are no roles included.
- The generated link would point to the learner portal search by default but having the `enterprise_customer_key` should allow a user to be linked from any page. We might want to redirect a user back to the search afterwards and strip the key from the url.
- Although key usage can be limited, users might still get unintended access to an `enterprise_customer_key` and link themselves to an enterprise. The admin portal will add a feature to list all of the learners associated with an enterprise and allow admins to manually remove them.

19.5 Phase 2 - Browsing anonymously

To support anonymous browsing, we will create a public page on the learner portal. Anyone with a universal link will be able to browse an enterprise catalog and request course access on this page. The learner portal will make calls to new/modified endpoints to fetch data and display the catalog. The phase 1 implementation enables users to be associated with an enterprise which is required for requesting course access. Note that the anonymous browsing component could be built before the phase 1 implementation if it doesn't the include course access request.

The process of generating the universal link will remain largely the same as in phase 1. The link would point to this new page rather than the search page, ie. `/catalog?enterprise_customer_key=key`. We will also validate the `enterprise_customer_key` in the query params before rendering the page. Revoking a link will also be the same process as in phase 1.

Since the learner portal makes calls to protected endpoints, we have to add/modify them to support the new public catalog page. The following are the APIs that the learner portal interacts with to display the enterprise catalog/course information:

- edx-enterprise `GET /enterprise-customer/`
  - Fetches enterprise customer data such as uuid, enterprise_customer_catalogs, etc. by the enterprise slug.
  - This is a protected endpoint.
  - There is a lite version `GET /enterprise-customer/basic_list` that currently returns only id and name of an enterprise. We will modify this to also include the enterprise_customer_catalogs and that will be the minimal information we need to render the page.
- Algolia search API
  - Queries Algolia for catalog data.
  - This only requires an api key which the learner portal already has access to.
- course-discovery `GET /courses/{key}/`
  - Queries course-discovery to get course information once a user clicks on a course.
  - This is a protected endpoint.
  - Enterprise catalog also hosts course data. We could expose `enterprise-catalogs/get_content_metadata/{key}` as a public endpoint for the learner portal to query course data without going through course-discovery. The enterprise-catalog is synced daily with course-discovery.
- enterprise-catalog `GET /contains_content_items/`
  - Checks whether or not the specified content is available to the EnterpriseCustomer.
  - This is a protected endpoint.
  - We will make this endpoint public.

Authenticated users would not be able to view this page and instead be redirected to the normal search page since it's outside of the normal learner portal flow.

19.5.1 Consequences

- We have to expose data through public endpoints which leads to concerns. For any endpoints we add/modify, we have to keep the amount of information returned to the minimum.
- An `enterprise_customer_key` remains valid until it's expired or the usage limit has been reached. However the usage limit only refers to the number of users that can be linked using the key. If we want to limit the number of times the key can be used to view the public catalog, we can keep track the number of unique visits and add a constraint.
- Having a public catalog could potentially increase the load on our system. This is not a huge concern and we will monitor the number of calls made.

CHAPTER

TWENTY

MOVING THE RESPONSIBILITY OF CATALOG DIFFING TO THE ENTERPRISE-CATALOG SERVICE

20.1 Status

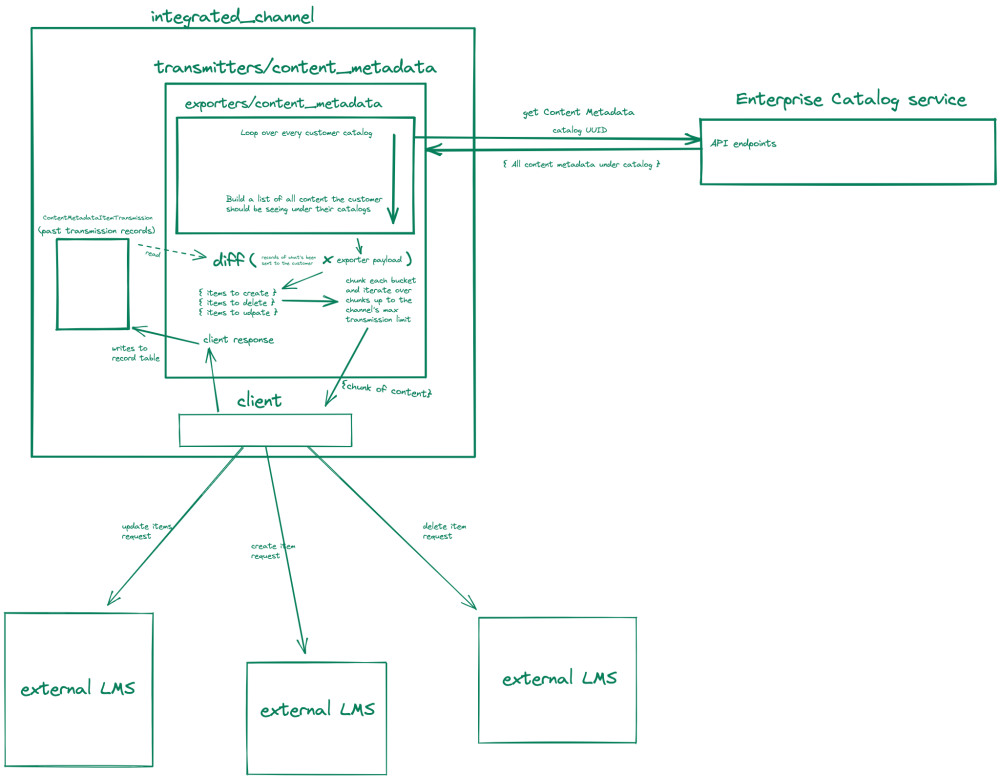
Accepted

20.2 Context

Recent, now reverted, attempts to make efficiency improvements to the integrated channels requesting data from the enterprise catalogs have highlighted short comings of the current architecture in terms of the capabilities of the Integrated Channels to accurately report the last time a customer's full catalog has been successfully updated. Because the Integrated Channels are responsible for chunking transmissions (only sending as much as the individual channel's API allows at a given time) such that not all content under a customer's catalog is guaranteed to be transmitted in a single scheduled run, we cannot accurately determine if all content under a customer's catalog has been updated since the catalog has last updated without first using the enterprise catalog's currently available API *get_content_metadata* endpoint.

- **How do the Integrated Channels currently handle customer catalog data?**

Existing flow chart:



Currently on a scheduled job, the Integrated Channels' content metadata exporter iterates over each customer's catalog and builds up a dict of content representing the current state of the customer's catalog content. Take away: as it stands now, the Integrated Channels assumes that the content metadata exporter will report the entirety of the customer's purchased content, then pass that payload to the content metadata transmitter which compares the payload with all saved *ContentMetadataItemTransmission* items under the customer to determine which items it (the transmitter) needs to create, update, and delete. It then chunks up each of the buckets, and sends those chunks (up to a limited number of times) to the client to send to the customer's LMS and saves the updated *ContentMetadataItemTransmission* entry.

- **Why does this matter?**

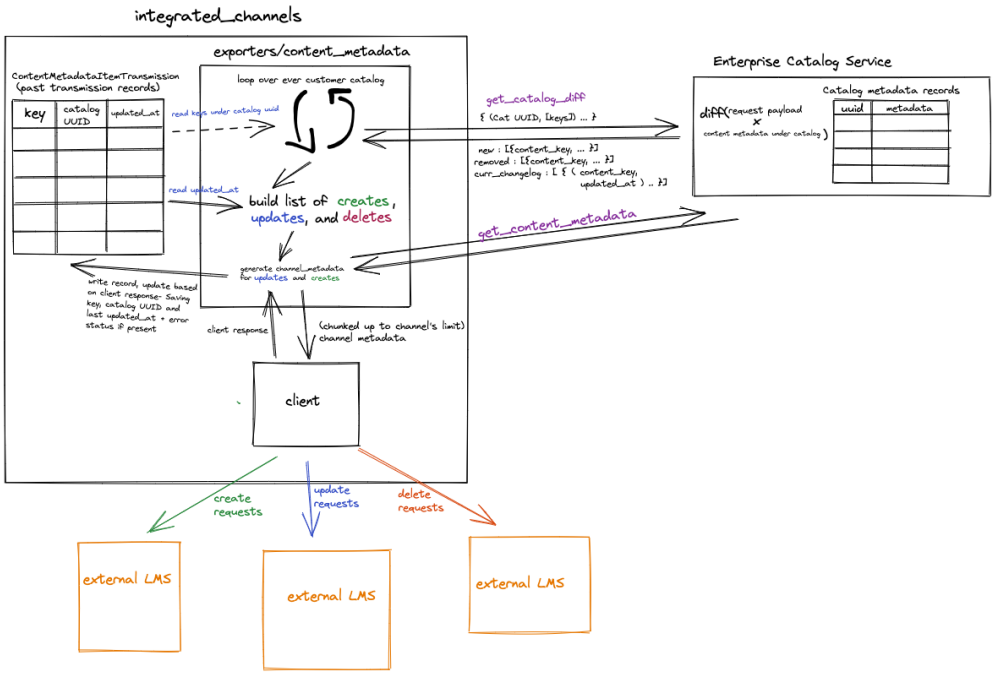
querying all content from the catalog service is expensive- Asking for all content under the catalog, for every customer, for every catalog, on every transmission is not only an undesirably large request to be sending, but also consumes much of the enterprise catalog's DB load. We need a smarter solution to how the Integrated Channels requests content from the catalog service.

putting the responsibility of determining differences between what's already been sent to the customers and what's currently in customer's catalogs adds much complexity to the integrated channels which can make it unapproachable from an onboarding perspective but also from a debugging point of view. Log noise and number of potential error sources means tracking down issues within the Integrated Channels are time consuming and thus, costly.

20.3 Decisions

- **How the Integrated Channels could better determine if customer metadata needs updating**

Proposed flow chart:



When we build an endpoint that takes a catalog UUID and a set of content keys linked to the uuid and returns three buckets of data:

1) What the last *updated_at* times of the content keys are for the content keys provided that are under the specified catalog

2) What content keys exist under the specified catalog that were not provided in the list of content keys

3) What content keys don't exist under the specified catalog that were provided in the list of content keys

We would be able to move the responsibility of diffing what we believe the customer has with what the customer should have, to the enterprise-catalog service via a new API endpoint. This would mean that the only job of the Integrated Channels would be to compare the last updated times of the content that already exist to get which content need updates, creates and deletes would already determined by the first endpoint.

20.4 Consequences

Adding the new diffing enterprise-catalog service API endpoint would allow us to accurately determine if catalog updates are needed for any particular iteration of the content metadata transmissions. We would be able to limit the number of calls needed to the enterprise-catalog service, saving time and DB load since the content_metadata endpoint is a hefty one. It would also allow us to easily implement a filtered content retrieval endpoint in the future such that we won't have to fetch any extraneous content whatsoever. Additionally these changes cut back on the complexity of the integrated channels, making it easier to approach and accurately identify future issues. Lastly, it moves the source of truth of catalog content information to the enterprise-catalog service and out of the Integrated Channels which should not be handling that responsibility in the first place.

20.5 Further Improvements

There are two additional improvements to explore with the Integrated Channels' metadata transmission flow. Firstly, we remove records from our transmission audit table when a delete request is issued. This can result in *lost* nodes of content (as we've actively seen from customers) where content can be assumed to be deleted on our end but have it exist still on the customer's external LMS. The fix here is pretty easy- if we choose to not delete records, but rather create a new, nullable *deleted_at* or equivalent field in the transmission record table. Then we would be able to mark and exclude any deleted records from appropriate look ups, but still have something to help us identify past courses that were sent to customers.

As stated earlier, currently the only method to retrieve content under a customer's catalog is the singular, bulk *get_content_metadata* endpoint. This returns all content metadata belonging to an enterprise catalog. If we were to build out the existing endpoint where individual content key's worth of metadata could be specified in the request body, then we would be able to further reduce the amount the Integrated Channels request unused data.

PYTHON MODULE INDEX

e

INDEX

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

V

W

X