django-oscar-cch Documentation

Release 2.2.4.post10

Craig Weber <craig@crgwbr.com>

Jul 21, 2017

Contents

This package handles integration between django-oscar based e-commerce sites and the CCH Sales Tax Office SOAP API.

Full Documentation: https://django-oscar-cch.readthedocs.io

CHAPTER 1

Contents

Installation

Caveats

1. You must fork the *order* application from Oscar to enable tax calculation as part of placing an order.
2. Persistence of tax details, while optional, requires that your project uses PostgreSQL. It relies on the HStore field.

Installation Guide

1. Install the `django-oscar-cch` package.:

```
$ pip install django-oscar-cch
```

2. Add `oscarcch` to your `INSTALLED_APPS`:

```
# myproject/settings.py
...
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.postgres',
    ...
    'oscarcch',
    ...
] + get_core_apps([
    ...
```

```
])
...
```

3. Add some attributes to `settings.py` to configure how the application should connect to CCH.:

```
# myproject/settings.py

# Add this is you need to connect to the SOAP API through an HTTP Proxy.
# See the instrumented-soap documentation for more details.
SOAP_PROXY_URL = ...

# Configure the CCH WSDL location, entity, and division code
CCH_WSDL = ...
CCH_ENTITY = ...
CCH_DIVISION = ...

# Provide either a product SKU or a product group and item to send to CCH when␣
↪calculating taxes
CCH_PRODUCT_SKU = ...
CCH_PRODUCT_GROUP = ...
CCH_PRODUCT_ITEM = ...
```

4. Alternative to setting `CCH_PRODUCT_SKU`, `CCH_PRODUCT_GROUP`, and `CCH_PRODUCT_ITEM` globally, you can set them per-product by creating ProductClass attributes with the same names (in lowercase).

5. Install the necessary extra fields on `order.models.Order` and `order.models.Line` (see also How to fork Oscar apps).:

```
# order/models.py

from oscarcch.mixins import CCHOrderMixin, CCHOrderLineMixin
from oscar.apps.order.abstract_models import AbstractOrder, AbstractLine

class Order(CCHOrderMixin, AbstractOrder):
    pass

class Line(CCHOrderLineMixin, AbstractLine):
    pass

from oscar.apps.order.models import *  # noqa
```

6. Create and run migrations for the *order* app.:

```
$ python manage.py makemigrations order
$ python manage.py migrate
```

For usage, continue to *Usage*.

Usage

Simple Integration

The library includes a mix-in class that can be added to *order.utils.OrderCreator* to enable tax calculation as part of the order placement process. Override `oscar.apps.order.utils.OrderCreator` in `order/utils.py` and add the mix-in directly before the super class:

```
from oscarcch.mixins import CCHOrderCreatorMixin
from oscar.apps.order import utils


class OrderCreator(CCHOrderCreatorMixin, utils.OrderCreator):
    pass
```

Custom Integration

For more complicated needs, you can interface with the tax calculation API directly. *CCHTaxCalculator* is used to apply taxes to a user's basket.:

```
from oscarcch.calculator import CCHTaxCalculator
from oscarcch.models import OrderTaxation


# Take a basket and the customer's shipping address and apply taxes to the basket. We␣
↪can optionally
# tolerate a failure to connect to the CCH server. In such a case, tax will be set to␣
↪0 and the method
# will return none. In normal cases, the method will return the details of the taxes␣
↪applied.
cch_response = CCHTaxCalculator().apply_taxes(basket, shipping_address, ignore_cch_
↪fail=True)
is_tax_known = (cch_response is not None)

# ...
# Do other things necessary to convert the basket into an order
# ...

# Take the tax details generated earlier and save them into the DB.
if is_tax_known:
    OrderTaxation.save_details(order, cch_response)
```

Django Settings

All settings in `oscarcch.settings` can be overridden by in your Django project's settings file.

Connection Settings

`oscarcch.settings.`**CCH_WSDL = 'file:///home/docs/checkouts/readthedocs.org/user_builds/django-oscar-cch/checkouts/lat**
Full URL of the CCH WSDL.

`oscarcch.settings.`**CCH_MAX_RETRIES = 2**
Max number of times to retry to calculate tax before giving up.

`oscarcch.settings.`**CCH_ENTITY = 'TESTSANDBOX'**
Default entity code to send to CCH.

`oscarcch.settings.`**CCH_DIVISION = '42'**
Default division code to send to CCH.

`oscarcch.settings.`**CCH_SOURCE_SYSTEM = 'Oscar'**
Name of the source system to send to CCH. Defaults to *Oscar*.

Transaction Settings

oscarcch.settings.**CCH_TEST_TRANSACTIONS = True**
Whether or not to set the test flag in CCH requests. Defaults to the same value as Django's DEBUG setting.

oscarcch.settings.**CCH_TRANSACTION_TYPE = '01'**
CCH Transaction Type Code. Defaults to 01.

oscarcch.settings.**CCH_CUSTOMER_TYPE = '08'**
CCH Customer Type. Defaults to 08.

oscarcch.settings.**CCH_PROVIDER_TYPE = '70'**
CCH Provider Type. Defaults to 70.

oscarcch.settings.**CCH_FINALIZE_TRANSACTION = False**
Whether or not to set the CCH finalize transaction flag. Defaults to False.

Product Taxation Settings

oscarcch.settings.**CCH_PRODUCT_SKU = 'ABC123'**
Default CCH Product SKU. Can be overridden by creating and setting a Product attribute called cch_product_sku.

oscarcch.settings.**CCH_PRODUCT_GROUP = ''**
Default CCH Product Group Code. Can be overridden by creating and setting a Product attribute called cch_product_group.

oscarcch.settings.**CCH_PRODUCT_ITEM = ''**
Default CCH Product Item Code. Can be overridden by creating and setting a Product attribute called cch_product_item.

Other Settings

oscarcch.settings.**CCH_TOLERATE_FAILURE_DURING_PLACE_ORDER = True**
When using the *Simple Integration*, this controls whether or not to allow placing an order when the call to CCH for tax calculation fails or times out. Defaults to True. When False and an error occurs, OrderCreator.place_order will raise an Exception.

oscarcch.settings.**CCH_PRECISION = Decimal('0.01')**
Construct a new Decimal object. 'value' can be an integer, string, tuple, or another Decimal object. If no value is given, return Decimal('0'). The context does not affect the conversion and is only passed to determine if the InvalidOperation trap is active.

oscarcch.settings.**CCH_POSTALCODE_LENGTH = 5**
Max length of postal-codes to send to CCH. Defaults to 5. All digits and characters after this limit will be clipped in the SOAP request.

oscarcch.settings.**CCH_TIME_ZONE = <DstTzInfo 'America/New_York' LMT-1 day, 19:04:00 STD>**

API Reference

PNP Installations

class oscarcch.calculator.**CCHTaxCalculator**
Simple interface between Python and the CCH Sales Tax Office SOAP API.

apply_taxes(*basket*, *shipping_address*, *ignore_cch_fail=False*)
Apply taxes to a Basket instance using the given shipping address.

Pass return value of this method to *OrderTaxation.save_details* to persist the taxation details, CCH transaction ID, etc in the database.

Parameters

- **basket** – Basket instance
- **shipping_address** – ShippingAddress instance
- **ignore_cch_fail** – When *True*, allows CCH to fail silently

Returns SOAP Response.

client
Lazy constructor for SOAP client

estimate_taxes(*basket*, *shipping_address*)
DEPRECATED. Use *CCHTaxCalculator.apply_taxes* instead.

Models

class oscarcch.models.**OrderTaxation**(**args*, ***kwargs*)
Persist top-level taxation data related to an Order.

messages
Message text returned by CCH

order
One-to-one foreign key to order.Order.

classmethod save_details(*order*, *taxes*)
Given an order and a SOAP response, persist the details.

Parameters

- **order** – Order instance
- **taxes** – Return value of *CCHTaxCalculator.apply_taxes*

total_tax_applied
Total Tax applied to the order

transaction_id
Transaction ID returned by CCH

transaction_status
Transaction Status returned by CCH

class oscarcch.models.**LineItemTaxation**(**args*, ***kwargs*)
Persist taxation details related to a single order line.

country_code
Country code used to calculate taxes

line_item
One-to-one foreign key to order.Line

state_code
State code used to calculate taxes

total_tax_applied
Total tax applied to the line

class oscarcch.models.**LineItemTaxationDetail**(**args*, ***kwargs*)
Represents a single type tax applied to a line.

data
HStore of data about the applied tax

taxation
Many-to-one foreign key to *LineItemTaxation*

Changelog

2.2.5

- Detect 9-digit ZIP codes in shipping a warehouse addresses and, instead of truncating the last 4 digits, send them in the *Plus4* field of the SOAP request.

2.2.4

- [Important] Fix bug causing order lines to get deleted is the corresponding basket or basket line is deleted.

2.2.3

- Handle bug occurring when a basket contained a zero-quantity line item.

2.2.2

- Upgrade dependencies.

2.2.1

- Simplified retry logic and fixed infinite loop issue.

2.2.0

- Improved documentation.
- **Added ability to retry CCH transactions when requests raises a ConnectionError, ConnectTimeout, or ReadTimeout.**
  - Added new setting, CCH_MAX_RETRIES, to control how many retries to attempt after an initial failure. Defaults to 2.

2.1.0

- Remove caching functionality from CCHTaxCalculator.estimate_taxes since miss rate was almost 100%.
- Fix bug in tax calculation causing taxes to be calculated based on pre-discounted prices instead of post-discounted prices.
- Add optional basket line quantity override by checking for property *BasketLine.cch_quantity*. Falls back to standard quantity if property doesn't exist.

2.0.0

- Renamed package to *oscarcch* for consistency. Set *db_table* option on models to prevent requiring table rename.
- Move tests inside *oscarcch* package.

1.1.1

- Fix bug where calculator could throw exception even when *ignore_cch_error* flag was set.

1.1.0

- Add the ability to set CCH product SKU, item, and group per-product in addition to globally.

1.0.5

- Add *CCH_TIME_ZONE* setting.
- Send time zone aware ISO format date as CalculateRequest InvoiceDate node. Formerly just sent the date.

1.0.4

- Truncate ZIP coes so that CCH doesn't choke when the user supplies a full 9-digit ZIP code.

1.0.3

- Improve unit tests by mocking all requests and responses. This allows running tests without a connection to an actual CCH server instance.
- Fixed bug where floats from SOAP response weren't properly converted into quantized decimals when saving *OrderTaxation* and *LineTaxation* models.

1.0.2

- Made *instrumented-soap* dependency optional.
- Moved gitlab testing from the shell executor to the docker executor.
- Added better usage documentation.

1.0.1

- Fixed an exception when *raven* isn't installed.

1.0.0

- Initial release.

Index