# django-oscar-paypal Documentation

*Release 1.0.0*

**David Winterbottom**

**May 30, 2018**

# Contents

This package provides integration between django-oscar and two of PayPal's payment options:

- *PayPal Express* - Involves redirecting customer's over to PayPal's site where they can choose shipping options and confirm payment using their PayPal account. The customer is then redirected back to the merchant site where they confirm the order.
- *PayPal PayFlow Pro* - Allows you to accept customer payments on your site without requiring a redirect to PayPal. This allows the customer to pay with a normal bankcard rather than their PayPal account.

It's possible to use both of these options individually or at the same time. Further, it's possible to use either without Oscar.

CHAPTER 1

# Installation

Whichever payment option you wish to use, the package installation instructions are the same.

Install:

```
pip install django-oscar-paypal
```

By default, this won't install Oscar as well. To install Oscar, run:

```
pip install "django-oscar-paypal[oscar]"
```

Finally, add `paypal` to your `INSTALLED_APPS`, and run:

```
python manage.py syncdb
```

CHAPTER 2

Table of contents

## 2.1 Express checkout

*PayPal Express* is an API for integrating PayPal payments into an ecommerce site. A typical implementation involves redirecting the user to PayPal's site where they enter their shipping and billing information before arriving back on the merchant site to confirm the order. It can also be used purely for payment, with shipping details being collected on the merchant site.

Oscar also supports a dashboard for PayPal Express transactions, which integrates with Oscar's dashboard.

See the PDF documentation for the gory details of PayPal Express.

### 2.1.1 Getting started

You need to create a PayPal sandbox account which is different from your normal paypal account then use this account to create two 'test' users: a buyer and a seller. Once the seller is created, you will have access to a username, password and 'signature' which are used to authenticate API requests.

Add the following settings using the details from your sandbox buyer account:

```
PAYPAL_API_USERNAME = 'test_xxxx.gmail.com'
PAYPAL_API_PASSWORD = '123456789'
PAYPAL_API_SIGNATURE = '...'
```

Next, you need to add the PayPal URLs to your URL config. This can be done as follows:

```
from django.contrib import admin
from oscar.app import shop

from paypal.express.dashboard.app import application

urlpatterns = [
    url(r'^admin/', admin.site.urls),
```

(continues on next page)

(continued from previous page)

```
    url(r'^checkout/paypal/', include('paypal.express.urls')),
    # Optional
    url(r'^dashboard/paypal/express/', application.urls),
    url(r'', shop.urls),
]
```

If you are using the dashboard views, extend the dashboard navigation to include the appropriate links:

```
from django.utils.translation import ugettext_lazy as _
OSCAR_DASHBOARD_NAVIGATION.append(
    {
        'label': _('PayPal'),
        'icon': 'icon-globe',
        'children': [
            {
                'label': _('Express transactions'),
                'url_name': 'paypal-express-list',
            },
        ]
    })
```

Finally, you need to modify oscar's basket template to include the button that links to PayPal. This can be done by creating a new template `templates/basket/partials/basket_content.html` with content:

```
{% extends 'oscar/basket/partials/basket_content.html' %}
{% load i18n %}

{% block formactions %}
<div class="form-actions">
    {% if anon_checkout_allowed or request.user.is_authenticated %}
        <a href="{% url 'paypal-redirect' %}"><img src="https://www.paypal.com/en_US/
↪i/btn/btn_xpressCheckout.gif" align="left" style="margin-right:7px;"></a>
    {% endif %}
    <a href="{% url 'checkout:index' %}" class="pull-right btn btn-large btn-primary">
↪{% trans "Proceed to checkout" %}</a>
</div>
{% endblock %}
```

Note that we are extending the `basket/partials/basket_content.html` template from oscar and overriding the `formactions` block. For this trick to work, you need to ensure that you have `OSCAR_MAIN_TEMPLATE_DIR` in your `TEMPLATE_DIRS` after your local templates setting:

```
from oscar import OSCAR_MAIN_TEMPLATE_DIR
TEMPLATE_DIRS = (
    location('templates'),
    OSCAR_MAIN_TEMPLATE_DIR,
)
```

If anything is unclear or not workin as expected then review how the 'sandbox' installation is set-up. This is a working Oscar install that uses PayPal Express.

### 2.1.2 Settings

There's a smorgasboard of options that can be used, as there's many ways to customised the Express Checkout experience. Most of these are handled by simple settings.

- PAYPAL_SANDBOX_MODE - whether to use PayPal's sandbox. Defaults to True.
- PAYPAL_CALLBACK_HTTPS - whether to use HTTPS for the callback URLs passed to PayPal. Defaults to True.
- PAYPAL_CURRENCY - the currency to use for transactions. Defaults to GBP.
- PAYPAL_API_VERSION - the version of API used (defaults to 119)
- PAYPAL_ALLOW_NOTE - whether to allow the customer to enter a note (defaults to True)
- PAYPAL_CUSTOMER_SERVICES_NUMBER - customer services number to display on the PayPal review page.
- PAYPAL_HEADER_IMG - the absolute path to a header image
- PAYPAL_HEADER_BACK_COLOR - background color (6-char hex value) for header background
- PAYPAL_HEADER_BORDER_COLOR - background color (6-char hex value) for header border
- PAYPAL_CALLBACK_TIMEOUT - timeout in seconds for the instant update callback
- PAYPAL_SOLUTION_TYPE - type of checkout flow ('Sole' or 'Mark')
- PAYPAL_LANDING_PAGE - type of PayPal page to display ('Billing' or 'Login')
- PAYPAL_BRAND_NAME - a label that overrides the business name in the PayPal account on the PayPal hosted checkout pages
- PAYPAL_PAGESTYLE - name of the Custom Payment Page Style for payment pages associated with this button or link
- PAYPAL_PAYFLOW_COLOR - background color (6-char hex value) for the payment page

Some of these options, like the display ones, can be set in your PayPal merchant profile.

You can also override the raw paypal params by defining a new paypal.express.views.RedirectView and define the _get_paypal_params method:

```
from paypal.express.views import RedirectView as OscarPaypalRedirectView


class RedirectView(OscarPaypalRedirectView):
    def _get_paypal_params(self):
        return {
            'SOLUTIONTYPE': 'Mark',
            'LANDINGPAGE': 'Login',
            'BRANDNAME': 'My Brand name'
        }
```

Please note that all the dynamic paypal params (e.g. amount, return_url, cancel_url etc.) cannot be overridden by _get_paypal_params.

### 2.1.3 PayPal Dashboard

You can view the merchant dashboard in PayPal's sandbox site by logging in as the sandbox master user, selecting the test seller account in the 'Test Accounts' tab then clicking 'Enter sandbox'.

### 2.1.4 Not included

The following options are part of the PayPal Express API but are not handled within this implementation - mainly as it's not obvious how you can handle these in a 'generic' way within Oscar:

- Gift wrapping
- Buyer consent to receive promotional emails
- Survey questions
- User confirming order on PayPal (bypassing review stage)
- Recurring payments
- Fraud management

### 2.1.5 Known issues

- Vouchers may have expired during the time when the user is on the PayPal site.

## 2.2 Payflow Pro

Payflow Pro is a server-to-server payment option for businesses who have a merchant account. Unlike Express Checkout, it doesn't require redirecting the user to PayPal's site and allows a customer to use a normal bankcard instead of their PayPal account. Read more details on the PayPal site.

---

**Note:** This version of the library was built using the PP_PayflowPro_Guide.pdf guide found in the docs/guides folder. It is recommended that developers at least skim-read this guide so that they are familiar with the overall processes. It also has magic bankcard numbers that can be used for testing. Find the latest developer docs on the PayPal site.

---

### 2.2.1 Getting started

You'll need to create an Payflow account with PayPal in order to get a:

- Vendor ID
- Username (usually same as vendor ID)
- Password
- Partner ID (normally "PayPal" when you register directly with PayPal)

In practice, you only really need a vendor ID and a password. Add settings to your project with your credentials:

```
# settings.py
...
PAYPAL_PAYFLOW_VENDOR_ID = 'mypaypalaccount'
PAYPAL_PAYFLOW_PASSWORD = 'asdfasdfasdf'
```

### 2.2.2 Next steps

The next steps are to plumb the payment gateway into your checkout and order processing. There is no one-size-fits-all solution here - your implementation will depend on your business model.

A good way to start is to browse the sandbox project within the repo - this is a fully integrated Oscar site.

Note that in an Oscar site, you should only consume the API of the `paypal.payflow.facade` module.

For checkout integration, you'll typically want the `PaymentDetailsView` in the following ways:

- Override the `get_context_data` method to provide a bankcard and billing address form.
- Override the `post` method to validate the forms and render them again in the preview view (but hidden).
- Override the `handle_payment` method of your checkout's `PaymentDetailsView` to call either the `authorize` or `sale` method of the facade depending on whether you are using one- or two-stage payment processing.

For general order processing integration, you'll likely need to adjust your `EventHandler` to make calls to the PayPal facade when certain shipping events occur. For instance, you may call `delayed_capture` when items ship in order to capture the funds at that stage.

You can log into your Payflow account to manage transactions under review and view reports. https://manager.paypal.com/

### 2.2.3 Settings

Required settings:

**PAYPAL_PAYFLOW_VENDOR_ID** Your merchant login ID created when you registered the account with PayPal.

**PAYPAL_PAYFLOW_PASSWORD** Your merchant password.

Optional settings:

**PAYPAL_PAYFLOW_CURRENCY** The 3 character currency code to use for transactions. Defaults to 'USD'.

**PAYPAL_PAYFLOW_USER** The ID of the user authorised to process transations. If you only have one user on the account, then this is the same as the VENDOR_ID and there is no need to specify it.

**PAYPAL_PAYFLOW_PARTNER** The ID provided by a PayPal reseller. If you created your account directly with PayPal, then the value to use is `"PayPal` - this is the default.

**PAYPAL_PAYFLOW_PRODUCTION_MODE** Whether to use PayPal's production servers. This defaults to `False` but should be set to `True` in production.

**PAYPAL_PAYFLOW_DASHBOARD_FORMS** Whether to show forms within the transaction detail page which allow transactions to be captured, voided or credited. Defaults to `False`.

### 2.2.4 Not included

- Recurring billing
- Account verification (TRXTYPE=A)
- Voice authorisation (TRXTYPE=F)
- SWIPE transactions (eg card present)
- Non-referenced credits (eg refunding to an arbitrary bankcard). All refunds must correspond to a previously settled transaction.

### 2.2.5 Using without Oscar

To use Payflow Pro without an Oscar install, you need to use the `paypal.payflow.gateway` module directly. This module is agnostic of Oscar and can be used independently.

The `paypal.payflow.facade` module is a bridging module that provides a simpler API designed to link Oscar to the gateway module.

### 2.2.6 API

#### Facade

#### Gateway

## 2.3 Contributing

Do this:

```
mkvirtualenv oscar-paypal
git clone git://github.com/django-oscar/django-oscar-paypal.git
cd django-oscar-paypal
make install
```

then you should be able to run the tests using:

```
py.test
```

There is also a sandbox site for exploring a sample oscar site. Set it up:

```
make sandbox
```

and run it:

```
./manage.py runserver
```

Use the Github issue tracker for any problems.

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search