# django-latch Documentation

*Release 0.2*

**Javier Moral**

**Mar 31, 2019**

# Contents

django-latch provides integration with Latch service by ElevenPaths, adding an aditional layer of security to the authentication process.

Currently, we support

- Python 3.4, 3.5, 3.6, 3.7
- Django 2.0, 2.1, 2.2

This package doesn't give an authentication method by itself, just modify the authentication flow stoping it if the user has decided to lock in his account. You must rely on another authentication method like Django default `ModelBackend`.

Originaly developed by Javier Olascoaga and RootedCON

CHAPTER 1

# Documentation contents

## 1.1 Installation

To install django-latch run:

```
$ pip install django-latch
```

You can install directly from source if you don't want to use PyPI.

To install it this way, simply:

```
$ git clone https://github.com/javimoral/django-latch.git
$ cd django-latch
$ python setup.py install
```

### 1.1.1 Configuration and use

We use Django Message Framework so you must configure your application accordingly.

Then, you must

- Include `latch` in your `INSTALLED_APPS`
- Append `latch.auth_backend.LatchAuthBackend` to `AUTHENTICATION_BACKENDS`
- Configure your API credentials
- Configure `LATCH_BYPASS_WHEN_UNREACHABLE`

```
INSTALLED_APPS = (
    [...]
     'latch',
 )
```

(continues on next page)

(continued from previous page)

```
# Append Latch Auth Backend the first in list
AUTHENTICATION_BACKENDS = [
    'latch.auth_backend.LatchAuthBackend',
    [...]
]

LATCH_APP_ID = <APP Id>
LATCH_APP_SECRET = <APP Secret>

LATCH_BYPASS_WHEN_UNREACHABLE = True # True is the default behaviour. Configure as
↪you need.
```

Next step, configure your project URLs.

```
from django.urls import path, include

urlpatterns = [
    [...]
    path('latch/', include('latch.urls'))
    [...]
]
```

Last, apply migrations after installing the app:

```
$ python manage.py makemigrations
```

**Warning: Upgrading from 0.2**

The method for configuring Latch API has changed. When upgrading from 0.2 the applied migrations will remove the model where API parameters used to be stored. You can get them again from Latch API settings page though.

### 1.1.2 Authentication mechanism

Latch doesn't care about the authentication mechanism, just stops authentication process raising a PermissionDeniedException when the account is locked.

Your application must rely on another authentication backends, putting `LatchAuthBackend` first in the list.

**Note: Timing attacks**

When a locked out paired account trys to connect we run the password hasher once to avoid timing attack. If the account doesn't exists, we pass the responsability to the next auth backend in chain.

### 1.1.3 Bypass when service is unreachable

`LATCH_BYPASS_WHEN_UNREACHABLE` controls the behaviour when Latch service is unavailable.

- If left unconfigured or set to `True` login attempts of paired accounts will be granted permission when connections with Latch service fails.
- If set to `False` login attempts of paired accounts will be denied when connections with Latch service fails.

### 1.1.4 Configuring API credentials

Configure in your `settings.py` as you prefer.

**Note:** The method for configuring your application sensitive settings will be dependant of you deployment type. Usually, the prefered method is to use environment variables.

```
import os

[...]

LATCH_APP_ID = os.environ.get('LATCH_APP_ID')
LATCH_APP_SECRET = os.environ.get('LATCH_APP_SECRET')
```

## 1.2 Usage

### 1.2.1 Views & Decorators

django-latch exposes the following views:

`latch.views.`**`pair`**(*request*, *template_name="latch_pair.html"*)
Process the pair form. If the user is alredy paired, redirects to the status view and shows a message using Django Message Framework.

View Name: `latch_pair`

`latch.views.`**`unpair`**(*request*, *template_name="latch_unpair.html"*)
Process the unpair form. If the user is not paired, redirects to the status view and shows a message using Django Message Framework.

View Name: `latch_unpair`

`latch.views.`**`status`**(*request*, *template_name="latch_status.html"*)
Show status about installation. We load two variables in this view:

- `accountid` Latch API account ID of the current user, if the account is paired.
- `account_status` Indicating if the user can login, value `on`, or not, value `off`.

Also note that retrieving status using the API will count as an login attempt, and the users can receive notifications on their devices.

View Name: `latch_status`

Also, it expose a decorator

`latch.views.`**`latch_is_configured`**(*view*)
If Latch is installed but not configured, decorated views will redirect to `status` instead.

### 1.2.2 Templates

Like django.contrib.auth we extend Django Admin templates. You can override the following templates:

```
latch
└── templates
    ├── latch_pair.html
    ├── latch_status.html
    └── latch_unpair.html
```

### 1.2.3 Loggers

We log failed API connections using `logger.exception`.

When `LATCH_BYPASS_WHEN_UNREACHABLE` we register each bypassed login with `info` level.

# Python Module Index

## l

# Index