# newauth Documentation

***Release 0.0.1***

**adrien-f**

April 11, 2015

Contents

NewAuth is a full featured authentication system for your Eve Online alliance or corporation. It brings out of the box support for multiple api keys per user, user groups with applications and invitations, and pings.

It is easily extensible and we're very much welcoming of new features!

CHAPTER 1

# Installation

## 1.1 Dependencies

You will need, of course, Python (tested on 2.7, might need some fixes for Python 3.x), a running SQL server (NewAuth has been tested with SQLite and PostgreSQL but it should work with MySQL) and a Redis server. Create a new SQL database and a user for NewAuth and keep its login details close. You might also need some extra packages to compile python dependencies, most notably libxml2 and database driver specific libraries.

## 1.2 Downloading

Here's the basic commands to download and install NewAuth:

```
$ git clone https://github.com/J4LP/newauth.git
$ cd newauth
$ virtualenv .
$ source bin/activate
$ pip install -r requirements.txt
```

## 1.3 Configuration

NewAuth configuration is read from the `newauth/settings.py` file. Refer to *Settings* for documentation and come back here!

## 1.4 Corporation API Key

In order to pick up your corporation/alliance, you will need to generate a corporation or alliance key with the Contact and Standings permissions. Please click this link to open CCP's API page to create one.

## 1.5 Initial launch

Once NewAuth is configured, you will need to migrate the database:

```
$ python manage.py db migrate
$ python manage.py db upgrade
```

You should then be able to start NewAuth with its development server:

```
$ python run.py
 * Running on http://0.0.0.0:5002/
```

Once NewAuth is confirmed to be working, it's time to import your corporation's contacts:

```
$ python manage.py update_contacts
```

And there you have it! NewAuth is now running on your computer. You are now able to create an account and login with it! Once your account created, make yourself admin with:

```
$ python manage.py make_admin $YOUR_USER_ID
```

Thanks you for using NewAuth!

## 1.6 Background Tasks

NewAuth is using Celery to delegate heavy tasks to the background and cronjobs. Please refer to Celery's documentation for more configuration options. You will need to launch it alongside newauth with:

```
celery -A manage.celery worker -Q newauth,celery -B
```

This command should be ran in the root directory of NewAuth.

CHAPTER 2

# Settings

Settings in NewAuth works by subclassing `BaseConfig` with "DevConfig" or "ProdConfig" or any other prefix. The prefix will be read from *NEWAUTH_ENV* and default to "Dev".

**class** `newauth.settings_dist.`**`BaseConfig`**

File based configuration object.

**`SECRET_KEY`**
Secret key for securing cookies. Generate one with *openssl rand –base64 64*

**`APP_DIR`**
Application absolute path

**`PROJECT_ROOT`**
Project root

**`DEBUG`**
Turn on debug mode by environment

**`SQLALCHEMY_DATABASE_URI`**
Default SQLAlchemy database

**`SQLALCHEMY_ECHO`**
Turn on debug mode for SQLAlchemy (prints out queries)

**`EVE`**
Eve related settings, see *Eve Settings*.

**`ADMIN_GROUP`**
The admin group

**`PING_GROUP`**
The ping group

**`PINGERS`**
Array of pings to load and use

**`PINGERS_SETTINGS`**
Runtime configuration for pingers

**`PLUGINS`**
Plugins list

## 2.1 Mail Settings

```
#: Mail settings for https://pythonhosted.org/flask-mail/
MAIL_SERVER = 'localhost'
MAIL_PORT = 25
MAIL_USE_TLS = False
MAIL_USE_SSL = False
MAIL_DEBUG = os.getenv('DEBUG', False)
MAIL_USERNAME = None
MAIL_PASSWORD = None
MAIL_DEFAULT_SENDER = None
```

## 2.2 Celery Settings

```
#: Celery Settings, more available http://celery.readthedocs.org/en/latest/configuration.html
CELERY_BROKER_URL = 'redis://localhost:6379'
CELERY_RESULT_BACKEND = 'db+sqlite:///celery.sqlite'
CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
```

## 2.3 Eve Settings

The Eve settings configure your NewAuth instance with your corporation/alliance identity and needs. For NewAuth to work, it needs one or several corporation api keys that will allow it to query its contacts to compute a list of allowable characters. See *Update Contacts* for more informations. Here's an example for the *I Whip My Slaves Back and Forth* alliance:

```
EVE = {
    'auth_name': 'J4LP',
    'requirements': {
        # Members need a mask of 65544538 and a key that does not expires
        # Allies need a mask of 50331656 and a key that can expires
        'internal': {'mask': 65544538, 'expires': False},
        'ally': {'mask': 50331656, 'expires': True}
    },
    'alliances': [99002172], # This is J4LP's alliance id
    'corporations': [],
    'keys': [(0, 'sekret')]
}
```

The *alliances* and *corporations* lists are a list of entity ID to add by default when updating the contacts. Here's an example

CHAPTER 3

# Manage commands

NewAuth ships with some CLI commands to help make your life easier.

## 3.1 Update Contacts

Usage:

```
$ python manage.py update_contacts
```

This method will fetch all contact lists of all the api keys given in *Eve Settings* in order to build a list of corporations and alliances to allow or deny access and registration to NewAuth.

## 3.2 Make Admin

**Description** Make a user join the group `newauth.settings_dist.BaseConfig.ADMIN_GROUP` and create the group if necessary.

**Usage** $ python manage.py make_admin $USER_ID

## 3.3 Make Ping

**Description** Make a user join the group `newauth.settings_dist.BaseConfig.PING_GROUP` and create the group if necessary.

**Usage** $ python manage.py make_ping $USER_ID

## 3.4 Update Users

**Description** Update one user or all users with Celery

**Usage** $ python manage.py update_users [user_id]

CHAPTER 4

# Models

**class** newauth.models.**User**(***kwargs*)

User model

**update_password**(*new_password*)

Hash a new password to bcrypt.

**Parameters** **new_password** (*str*) – The new password

**check_password**(*password*)

Check if given password checks out. If the password is in a LDAP format, it will convert it to bcrypt.

**Parameters** **password** (*str*) – The password to compare

**Returns** bool – If the password checks out or not

**update_status**()

Loop around characters and keys to determinate best status.

Ineligible < Ally < Internal

:returns CharacterStatus – The status that was computed

**update_keys**()

Update all API Keys and Characters associated.

CHAPTER 5

# Plugins

NewAuth allows the loading of foreign plugins to extend its features. These plugins are loaded at the end of the creation of the application after models, routes and other settings have been loaded.

Plugins follow the architecture of a Flask extension.

## 5.1 Example plugin

This is an example of a NewAuth plugin that hooks into different signals and add a route to the admin.

```
class ExamplePlugin(object):
    """Example plugin for NewAuth"""

    def __init__(self, app=None):
        if app:
            self.init_app(app)

    def init_app(self, app):
        self.app = app

        self.app.logger.debug("ExamplePlugin enabled.")

        # Registering events
        User.login_success.connect_via(app)(self.login_success)
        User.login_fail.connect_via(app)(self.login_fail)

        # Registering template hooks
        if not hasattr(app, 'admin_user_hooks'):
            app.admin_user_hooks = [self.admin_user_hook]
        else:
            app.admin_user_hooks.append(self.admin_user_hook)
        if not hasattr(app, 'dashboard_hooks'):
            app.dashboard_hooks = [self.dashboard_hook]
        else:
            app.dashboard_hooks.append(self.dashboard_hook)

        # Registering routes
        app.add_url_rule('/admin/example', 'example_route', self.admin_example_route)

        # Registering to navbar
        app.navbar['admin'].append(('fa-info', 'Example Route', '/admin/example'))
```

```
    def login_success(self, app, user):
        current_app.logging.debug('Login success')

    def login_fail(self, app, user):
        current_app.logging.debug('Login fail')

    def admin_user_hook(self, user):
        return 'This will be displayed on the user\'s admin profile'

    def dashboard_hook(self, user):
        return 'This is a dashboard widget'

    def admin_example_route(self):
        return 'This is an extra route'
```

## 5.2 LDAP Plugin

NewAuth ships with a LDAP plugin that allow the application to save its user data to a LDAP directory.

### 5.2.1 Configuration

```
SYNC_LDAP_HOST = '127.0.0.1'
SYNC_LDAP_ADMIN_USER = 'cn=admin,dc=example,dc=org'
SYNC_LDAP_ADMIN_PASS = 'admin'
SYNC_LDAP_BASEDN = 'dc=example,dc=org'
SYNC_LDAP_MEMBERDN = 'ou=People,dc=example,dc=org'
```

### 5.2.2 Management commands

```
python manage.py ldap import_users [--user_id $USER_ID]
```

This command will import all users in *SYNC_LDAP_MEMBERDN*. We recommend making a backup of your LDAP server before though. If you include add *–user_id* it will only import this user.

## 5.3 Tasks Dashboard

NewAuth is using Celery to leverage some heavy tasks to the background, by default, no dashboard is available because it requires an external dependency. If you so wish, you can get a basic dashboard available to your admin and a better more functional one for your IT team. More information about Flower can be found here.

### 5.3.1 Requirements

This plugin requires you to install Flower, a Celery dashboard providing a REST API.

```
pip install flower
```

We also need to run Flower alongside NewAuth and Celery with:

```
celery -A newauth.tasks flower --address=127.0.0.1 --port=5555
```

### 5.3.2 Configuration

To enable *Tasks Dashboard*, add `newauth.plugins.tasks_dashboard.TasksDashboard` to the *PLUGINS* setting.

This plugin only requires one setting:

```
CELERY_FLOWER_URL = 'http://127.0.0.1:5555/  # Note the trailing slash
```

CHAPTER 6

# Pings

A Ping is a message sent through NewAuth to a group of people registered. It can be sent over XMPP or Pushbullet for example.

## 6.1 XMPP Pinger

NewAuth can send its pings to a Jabber server. To enable this pinger, add `newauth.plugins.ping.xmpp.XMPPPinger` to the *PINGERS* setting.

## 6.2 Pushbullet Pinger

NewAuth can send its pings to Pushbullet, a notification web service that has applications for Chrome, Android, iOS and more.

To enable this pinger, add `newauth.plugins.ping.pushbullet.PushbulletPinger` to the *PINGERS* setting.

## 6.3 Create your own Pinger

You would need to subclass and implement the `newauth.plugins.ping.Pinger`.

**class** `newauth.plugins.ping.`**`Pinger`**(*app=None*)
Pinger base class to create a new pinger. Works like a plugin.

**`description`** = ''
Description to be displayed, supports Markdown.

**`disable`**(*user*, *configuration*)
Called when disabling the pinger.

You can do additional steps before NewAuth disable this pinger for the user.

Feel free to flash messages through Flask's flash.

*configuration* will be serialized, saved and tied to the user.

**Returns** Return the configuration if successful or False

**`display_name`** = ''
Pinger name to be displayed.

**enable**(*user*, *configuration*)
Called when enabling the pinger after the form has been validated.

Feel free to flash messages through Flask's flash.

*configuration* will be serialized, saved and tied to the user.

**Returns** Return True, False, or a dict for more action. See *pingers* for more info.

**enabled**(*user*)
Test if the user has the pinger enabled and configured.

**Return type** boolean

**get_form**(*user_config*)
Return a WTForm to display on the pinger settings page or None if no configuration needed. :param user_config: :return:

**immutable = False**
Can this pinger be disabled ?

**init_app**(*app*)
Called when the application starts, use to check for settings or register signals :param app: :return:

**name = ''**
Pinger name used for configuration storage and in other various places.

**save_configuration**(*user*, *configuration*, *form*)
Called when saving the plugin settings and the form has been validated.

Feel free to flash messages through Flask's flash.

*configuration* will be serialized, saved and tied to the user.

**Returns** Return the configuration if successful or False

**send_ping**(*ping*)
Send the ping to the users.

CHAPTER 7

# Indices and tables

- *genindex*
- *modindex*
- *search*