# Tranny Documentation

***Release 0.1.1***

**Leigh MacDonald**

**May 04, 2017**

# Contents

Contents:

CHAPTER 1

# Tranny Setup Info

This documents outlines setting up the tranny process to start on boot as well as configuring different services to fetch from.

## Requirements

One of the following torrent clients configured for remote API access

- deluge This is currently the only client that will function with webui capabilities
- rTorrent (scgi) (Partial support)
- Transmission (jsonrpc) (Not recommended yet)
- uTorrent 3.x (webui) (Not recommended yet)

Along with a python distribution and required python libraries

- Python 2.7 - Python 3 support will come with python 3 gevent support
- watchdog
- transmissionrpc
- requests
- feedparser
- jsonrpclib

## Setup

Its recommended to use a virtualenv to install if possible. Ill outline how to install using it below.

Checkout the sources:

```
$ git clone git://github.com/leighmacdonald/tranny.git
$ cd tranny
```

## Unix System Prep

If you are using a Unix like system (linux,osx,bsd,solaris) you should use these steps, for windows users please see below.

Install virtualenv with your package manager if its not already installed. Install the dependencies using pip and the dependencies which are listed under the requirements.txt file.:

```
$ virtualenv virtenv # Create virtualenv for tranny
$ source virtenv/bin/activate # Activate new interpreter on your path
$ pip install -r requirements.txt # Install required dependencies
```

## Windows System Prep

Below are links to installers for some of the libraries required. Make sure you choose the correct one for your python version. Sticking with CPython 2.7 would be considered best for now.

- distribute
- pip
- psutil

The following libraries are installed using pip as they are not available from the site above. You can install them like so: *pip install package_name*

- six
- pathtools

A precompiled installer for pyimdb 32bit python 2.7. - imdbpy

## Common configuration setup

Create your own config based on the example provided and rename it to *tranny.ini*. We will also put this in the user's config directory under ~/.config/tranny

$ mkdir ~/.config/tranny $ cp tranny_dist.ini ~/.config/tranny/tranny.ini

From this point you will want to take a moment and setup your configuration file. You should enable all the services you want to now. For more info check out the following setup doc pages:

- *External Services* Service configuration [imdb, trakt, tmdb, etc]
- *Notification Services*

Using the editor of your choice, configure any options desired in your new configuration.

$ vim tranny.ini

Once you have configured your configuration file as desired, notably making sure the database uri is specified properly, you can then initialize a blank database schema. There are 3 optional parameters that you can also set if you desire. -u/-p will set the credentials for the admin user. This is currently only used for the optional web ui. If you do not specify then default values of admin/tranny will be used for the user/password. -w If specified will wipe all existing tables and data from your database. This is not recoverable so be sure to do it only if you need to.

> $ python tranny-cli.py db_init [-u admin_username] [-p admin_password] [-w wipe existing database] 2014-09-16 01:32:32,338 - Initialized db schema successfully 2014-09-16 01:32:32,363 - Created admin user successfully

You can now start the daemon process like so.

> $ python tranny-cli.py run

If you are having issues you can also run the application with debugging log level as well as follows:

> $ python tranny-cli.py -l DEBUG run

I recommend running it like this for a while so you can monitor it for any issues that arise. Once you are confident in how its working you can proceed to enable it to start automatically at boot as a daemon.

## Auto Start On Boot

There are a plethora of ways to install the service to run at boot time. These however are implemented quite a bit different across platforms so be warned, I only test on Linux currently. If you have success on other platforms please submit a pull request for this document with the relevant steps required outlined or even just a note telling me of the status.

## Linux (supervisord)

This is the easiest method, mostly because its the method most tested with. You will of course need to install the supervisor daemon and make sure its enable to start at system boot.

Below is a sample configuration file that should be customized according to your setup. Save the file as */etc/supervisor.d/tranny.ini* or append this config to the standard */etc/supervisord.conf* file:

```
[program:tranny]
command=/home/user/tranny/virtenv/bin/python /home/user/tranny/tranny-cli.py run
directory=/home/user/tranny
stdout_logfile=/home/user/.config/tranny/tranny-supervisor.log
redirect_stderr=true
user=user
```

## Windows

Who knows... but [this](http://stackoverflow.com/questions/32404/is-it-possible-to-run-a-python-script-as-a-service-in-windows-if-pos
can probably help you get started. Please let me know if you have success.

CHAPTER 2

# Management Scripts

There are several scripts beyond the main application which provide some extra convience for performing some tasks via the command line. Most of the scripts live under the scripts path of the source tree with the exception of *manage.py*. These scripts will all attempt to load the global config *tranny.ini* before executing, so you can expect those values to be available when executing.

## manage.py

This script proves 2 main functions. Initial configuration of the application and running the application.

- *manage.py initdb* Drop the databased tables, if they exist, and reinitialize it, creating a new admin user in the process.
- *manage.py run* Load and run the application from the command line, this does not fork the process into the background.

## tranny-add.py

This script will attempt to load the files passed into it as arguments into the configured torrent client. The client it uses is determined by the *tranny.ini* client value. The script will accept 1 or more paths to torrent files, an example of loading 2 files is below.

```
./tranny-add.py The.Torent.Name.2012.S01E01.HDTV.x264-GRP.torrent The.Other.Name.2012.
↪x264-GRP.torrent
> Connected to rTorrent 0.9.3/0.13.3
-> The.Torent.Name.2012.S01E01.HDTV.x264-GRP @ 376.2MB
--> Upload successful
-> The.Other.Name.2012.x264-GRP @ 2356.3MB
--> Upload successful
```

CHAPTER 3

# Torrent Client Configuration

Examples of how to setup backend torrent clients for use with tranny.

The most feature complete client is deluge so it is the recommended choice for now.

## rTorrent

The simplest way to make this work is using direct SCGI access, however this method is less secure by default and should have firewall rules in place to lock it down. The main advantage is using it locally and not needing to setup a web server to proxy the SCGI request to rTorrent. If you are accessing a remote instance of rTorrent its probably a better idea to use an actual web server if you are not confident in setting up secure firewall restrictions.

## Deluge (JSON-RPC)

More info available at Deluge JSON-RPC

To allow connections to deluge you must enable the WebUI plugin. Additionally the default settings disallow remote connections. Change the "allow_remote" setting in $HOME/.config/deluge/core.conf:

```
"allow_remote": true
```

deluge_api

## UTorrent

Support coming soon

## Transmission

For tranny to be able to communicate with the transmission client you must enable remove access. You can find this in the `preferences -> remote` menu of the client.

If you are accessing the client from a remote connection, as in your client is not on the same computer as tranny, you must enable the rpc-whitelist for your ip:

```
"rpc-whitelist": "127.0.0.1,192.168.10.*,10.0.0.*",
"rpc-whitelist-enabled": true,
```

You can set this in transmissions settings.json

CHAPTER 4

# Metadata Services

## IMDB (HTTP)

**Note:** As of version 5 of the IMDBpy library this step is no longer necessary and it will work as expected from the official sources which should automatically be installed. If you are stuck with 4.x for some reason, then you will need to install a current version similar to how is outlined below.

As of 17/04/2013, the IMDBpy library does not work with the current IMDB site. To enable usage of IMDB you must install a development version of the library. You will need mercurial installed to download the source tree. Below is a step by step example of installing the library.:

```
$ cd tranny
$ source virtenv/bin/activate
$ hg clone ssh://hg@bitbucket.org/alberanid/imdbpy
$ cd imdbpy
$ python setup.py install
```

## IMDB (SQL)

Setup for this has been automated so it will download and install the database for you. This will auto create the tables and populate them for you automatically in the sqlalchemy URI that you have defined in your config.:

```
$ python tranny-cli.py imdb
```

**Warning:** This process can take a *VERY* long time to complete depending on your hardware and database used. If you are using anything but SQLite this can take more than 24 hours to complete on fairly good hardware, even with an SSD. Below i will show some times its taken me when testing to do this. If you have more data points please send them to me or add them and create a pull request.

- SQLite ~35mins (i5-3470/Samsung Pro SSD/32gb)
- Postgresql 9.3.5 30hrs (conf: fsync=off) (i5-3470/Samsung Pro SSD/32gb)

Once you have completed your import, you must change your config to show sql = true.:

```
[service_imdb]
enabled = true
sql = true
```

You will probably want to run this every so often to keep things current with the live imdb site by using a cron job of some sort. I wouldn't recommend doing this every day, but once a week for sqlite is probably ok, once every couple weeks to a month for postgres.

## themoviedb.org

To use themoviedb info, you must register and create an API key. Once you have the API key you can add it to your tranny.ini as follows::

```
[themoviedb]
api_key = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

**Note:** This module is not really used at the moment and may be removed if it doesnt offer enough over the others.

CHAPTER 5

# Providers

Providers are considered to be modules that allow fetching release lists from remote sources as well as downloading from the sources. This includes common stuff like RSS and API based services, but can be extended to support anything that can provide that type of information

## BTN API Provider

BTN provides its own API over JSON-RPC. It requires a API key which can be obtained from going to your profile and clicking the API tab. If you do not already have a key, you can create a new API key, otherwise use your existing one.

Below is a full set of configuration values which are used for the service. You should only need to update the api_token to have a working service.:

```
[provider_broadcastthenet]
enabled = true
url = http://api.btnapps.net/
interval = 150
; 32 character API Key.
api_token = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Be aware that BTN has an API rate limit of 150req/s so do not set your interval too low. If you want data faster than this, you should be using an IRC bot instead as it is event based and will get notified immediately of a new release.

## RSS Feed Provider

Tranny supports downloading over RSS feeds. You can add as many sections as you would like:

```
[rss_name]
url = http://rss.torrensite.org/
```

```
enabled = true
interval = 300
```

You should be careful not to set your interval too low. Most sites really do not like less than 300 seconds (5min) and you may get banned or run into rate limit issues if you do not respect that.

## PTP Provider

This is currently not 100% implemented and should not be used yet.

## HDBits Provider

This is currently not 100% implemented and should not be used yet. Also keep in mind that HDBits is not all that keen on scripts being used to download from them.

CHAPTER 6

# External Services

External/3rd party API's used to fetch metadata for media

## IMDB

### SQL Mode

You can optionally import the entire imdb into a local databse for faster lookup times.:

```
$ tranny-cli.py imdb
```

Using SQLite on a very fast machine with SSD's this took about 30 minutes to complete. Other databases are likely to be considerably slower than this. For reference i have seen postgres take 2 days with a default configuration.

Once complete the config flag for sql must be set to true:

```
[service_imdb]
enabled = true
sql = true
```

If using postgresql there are some parameters you can tune in the database to speed up this load time.:

```
fsync = off
full_page_writes = off
```

For more information on speeding up postgresql's insertion speed see this post.

CHAPTER 7

# Notification Services

yep.

## IRC

## Email

## SMS

## Growl

CHAPTER 8

# Developer Guide

A number of different tools are used to develop with. The most complicated setup is in regards to the web tool set used, built around grunt and sass. The python stuff is very simple and easy to get running with PIP.

## Python Setup

Its highly recommended you develop under a virtualenv, so this is the only method i will describe.

We require python 2.7 currently, so if you do not have this please install it. You do not need to install it system wide, nor should you. The setup for this is outside the scope of this doc so please refer to external docs for this. I do however recommend just installing it to a custom directory using `./configure --prefix=/home/user/python/2.7` or something similar. Be sure to reference that specific python executable when running any commands.

If you do not have virtualenv installed please do so. If you just built python 2.7 then you will want to install it. Here are the projects instructions on how to do this.

With that installed we can go ahead and create our virtualenv:

```
$ cd $project_root
$ /path/to/python2.7 virtualenv virtenv
```

This will install it under virtenv, but you can name this whatever you want. We can now activate the virtualenv using the activate script. This will make the virtualenv python the default python used while under that shell.:

```
$ source virtenv/bin/activate
```

We can now install all of the standard python dependencies and well as the developer dependencies since this is the dev guide:

```
$ pip install -r requirements.txt
$ pip install -r requirements-devel.txt
```

You should now be good to go if there was no problems.

## Web Development Setup

In short you will need the following tools installed before continuing, the installation process for these is out of scope for this, so you should read their respective docs on how to get them installed correctly:

- ruby
- nodejs

Basic steps on linux (# denoting root or sudo command):

```
# npm install -g bower grunt-cli
$ gem install foundation
```

With all of the base tools installed we can pull in our JS dependencies using bower:

```
$ bower install --config.directory=tranny/static/vendor
```

Now we will install the grunt tasks that we take advantage of:

```
$ npm install grunt-contrib-concat --save-dev
$ npm install grunt-contrib-coffee --save-dev
$ npm install grunt-contrib-uglify --save-dev
$ npm install grunt-contrib-cssmin --save-dev
```

If that all completed successfully you should be able to run the following with a similar result, notably the last line.:

```
$ grunt build
Running "sass:dist" (sass) task
File "tranny/static/css/app.css" created.

Running "coffee:compileWithMaps" (coffee) task
>> 1 files created.
>> 1 source map files created.

Running "concat:dist" (concat) task
File tranny/static/js/vendor.js created.

Done, without errors.
```

If that was successful you should be good to go and can now use the default grunt task which will watch for changes in your files and recompile them on each save.:

```
$ grunt
Running "sass:dist" (sass) task
File "tranny/static/css/app.css" created.

Running "coffee:compileWithMaps" (coffee) task
>> 1 files created.
>> 1 source map files created.

Running "concat:dist" (concat) task
File tranny/static/js/vendor.js created.

Running "watch" task
Waiting...
```

For a basic video guide on doing this, please see the zurb foundation demonstration video.

### Problems

There is an issue with libsass as it does not support sass 3.4 yet. If you receive this error when trying to build the project you must make a small change:

```
Warning: tranny/static/vendor/foundation/scss/foundation/functions:13: error: error␣
↪reading values after )
Use --force to continue.
```

To fix this go to line 13 of tranny/static/vendor/foundation/scss/foundation/functions and remove the !global declaration so it looks like this:

```
$modules: append($modules, $name);
```

## Building Release Versions

todo

## Testing

When running unit tests there are 2 environment variables used to configure what is tested and loaded. By setting `TEST=` you will load the test configuration in the fixtures folder. There are additional values used to flag the testing of specific clients since we do not currently test them all and do not mock everything yet.:

```
TEST=1/0
TEST_TRANSMISSION=1/0
TEST_DELUGE=1/0
TEST_RTORRENT=1/0
TEST_QBITTORRENT=1/0
TEST_UTORRENT=1/0
```

---

**Note:** When running the test suites you don't generally need to set TEST=1 manually as its done automatically when loading the tests/testcase.py module.

---

CHAPTER 9

# Deluge API Methods

A list of the methods exported for use by Deluges WebUI module.

## Pre Authentication

Before auth.login -> web.connect sequence the following methods are available

auth.change_password
auth.check_session
auth.delete_session
auth.login
web.add_host
web.add_torrents
web.connect
web.connected
web.deregister_event_listener
web.disconnect
web.download_torrent_from_url
web.get_config
web.get_events
web.get_host_status
web.get_hosts
web.get_magnet_info
web.get_plugin_info
web.get_plugin_resources
web.get_plugins
web.get_torrent_files

web.get_torrent_info
web.get_torrent_status
web.register_event_listener
web.remove_host
web.set_config
web.start_daemon
web.stop_daemon
web.update_ui
web.upload_plugin

## Post Authentication

auth.change_password
auth.check_session
auth.delete_session
auth.login
core.add_torrent_file
core.add_torrent_magnet
core.add_torrent_url
core.connect_peer
core.create_torrent
core.disable_plugin
core.enable_plugin
core.force_reannounce
core.force_recheck
core.get_available_plugins
core.get_cache_status
core.get_config
core.get_config_value
core.get_config_values
core.get_enabled_plugins
core.get_filter_tree
core.get_free_space
core.get_libtorrent_version
core.get_listen_port
core.get_num_connections
core.get_path_size
core.get_session_state
core.get_session_status
core.get_torrent_status
core.get_torrents_status
core.glob
core.move_storage
core.pause_all_torrents
core.pause_torrent
core.queue_bottom

core.queue_down
core.queue_top
core.queue_up
core.remove_torrent
core.rename_files
core.rename_folder
core.rescan_plugins
core.resume_all_torrents
core.resume_torrent
core.set_config
core.set_torrent_auto_managed
core.set_torrent_file_priorities
core.set_torrent_max_connections
core.set_torrent_max_download_speed
core.set_torrent_max_upload_slots
core.set_torrent_max_upload_speed
core.set_torrent_move_completed
core.set_torrent_move_completed_path
core.set_torrent_options
core.set_torrent_prioritize_first_last
core.set_torrent_remove_at_ratio
core.set_torrent_stop_at_ratio
core.set_torrent_stop_ratio
core.set_torrent_trackers
core.test_listen_port
core.upload_plugin
daemon.get_method_list
daemon.info
daemon.shutdown
web.add_host
web.add_torrents
web.connect
web.connected
web.deregister_event_listener
web.disconnect
web.download_torrent_from_url
web.get_config
web.get_events
web.get_host_status
web.get_hosts
web.get_magnet_info
web.get_plugin_info
web.get_plugin_resources
web.get_plugins
web.get_torrent_files
web.get_torrent_info
web.get_torrent_status

web.register_event_listener
web.remove_host
web.set_config
web.start_daemon
web.stop_daemon
web.update_ui
web.upload_plugin
webui.get_config
webui.got_deluge_web
webui.set_config
webui.start
webui.stop

CHAPTER 10

# Tranny

## tranny package

### Subpackages

#### tranny.client package

#### Submodules

#### tranny.client.deluge module

#### tranny.client.rtorrent module

#### tranny.client.transmission module

#### tranny.client.utorrent module

#### Module contents

#### tranny.handlers package

#### Submodules

#### tranny.handlers.filters module

#### tranny.handlers.home module

#### tranny.handlers.rss module

#### Module contents

### Submodules

### tranny.app module

### tranny.configuration module

### tranny.constants module

### tranny.datastore module

### tranny.events module

### tranny.exceptions module

### tranny.extensions module

### tranny.forms module

### tranny.manager module

### tranny.models module

### tranny.net module

### tranny.notification module

### tranny.parser module

### tranny.plugin module

### tranny.release module

### tranny.stats module

### tranny.torrent module

### tranny.ui module

### tranny.util module

### tranny.watch module

## Module contents

## Client API Info

This product includes GeoLite2 data created by MaxMind, available from www.maxmind.com

CHAPTER 11

# Indices and tables

- genindex
- modindex
- search