torrt Documentation

Release 1.0.0

Igor 'idle sign' Starikov

Feb 03, 2023

Contents

https://github.com/idlesign/torrt

CHAPTER 1

Description

Automates torrent updates for you.

torrt automatically checks your favourite torrent tracker sites, where torrents are organized as articles (i.e forum-like tracker), to verify whether specific torrents have been updated (e.g. torrent bundling some TV-series is updated with a new episode), and instructs your torrent client to download new files.

torrt can function both as a console application and Python module.

1.1 Trackers

Automatic updates are available for:

- AniDUB - http://tr.anidub.com/
- AniLibria - https://www.anilibria.tv/
- CasStudio - https://casstudio.tv
- Kinozal - http://kinozal.tv/
- NNM-Club - http://nnm-club.me/
- RUTOR - http://rutor.org/
- RuTracker (ex torrents.ru) - http://rutracker.org/

1.2 Torrent clients

torrt is able to cooperate with the following torrent clients:

- Transmission (using built-in JSON RPC)
- Deluge (using *deluge-webapi* plugin - https://github.com/idlesign/deluge-webapi)
- uTorrent (using built-in RPC)

- qBittorrent (using built-in RPC) v4.1+

1.3 Notifications

torrt is able to send update notifications using:

- E-Mail (SMTP)
- Telegram (via Telegram Bot API) - https://core.telegram.org/bots/api

1.4 Bots

torrt can be managed using messenger's bots:

- Telegram (via Telegram Bot API) - https://core.telegram.org/bots/api

CHAPTER 2

Requirements

1. Python 3.7+
2. `deluge-webapi` plugin (to work with Deluge)
3. `python-telegram-bot` (to run Telegram bot)

CHAPTER 3

Table of Contents

3.1 Quickstart

In this examples we'll use console commands.

1. **torrt** cooperates with actual torrent clients (through RPC), so we need to tell it how to connect to them.

   Let's configure Transmission (https://www.transmissionbt.com/) connection (considering Transmission web interface is on 192.168.1.5):

   ```
   > torrt configure_rpc transmission host=192.168.1.5 user=idle password=pSW0rt
   ```

   To get known RPCs aliases use *list_rpc* command.

2. Second step is to tell **torrt** how to authorize into private torrent trackers.

   Let's configure authorization for http://rutracker.org tracker:

   ```
   > torrt configure_tracker rutracker.org username=idle password=pSW0rt
   ```

   To get known tracker aliases use *list_trackers* command.

3. Now let's subscribe to torrent updates, say from http://rutracker.org/forum/viewtopic.php?t=4430338:

   ```
   > torrt add_torrent http://rutracker.org/forum/viewtopic.php?t=4430338
   ```

   To get torrents registered for updates use *list_torrents* command.

4. Configure notifications about torrent updates.

   Let's configure notifications through email:

   ```
   > torrt configure_notifier email host=smtp.server.com port=25 use_tls=True␣
   ↪email=your@email.com user=idle password=pSW0rt
   ```

5. Updates checks for torrents registered within **torrt** can be done with *walk* command:

```
> torrt walk
```

6. Use *set_walk_interval* command to set walk interval in hours:

```
> torrt set_walk_interval 24
```

Note: More information on commands supported by **torrt** console application is available through *–help* command line switch:

```
> torrt --help
> torrt configure_rpc --help
> torrt add_torrent --help
```

3.2 Console application commands

Those are **torrt** console application commands:

3.2.1 Trackers

- **list_trackers** - Shows known trackers aliases
- **configure_tracker** - Sets torrent tracker settings (login credentials, etc.)

3.2.2 PRCs

- **list_rpc** - Shows known RPCs aliases
- **configure_rpc** - Sets RPCs settings (login credentials, etc.)
- **enable_rpc** - Enables RPC by its alias
- **disable_rpc** - Disables RPC by its alias

3.2.3 Torrents

- **add_torrent** - Adds torrent from an URL both to *torrt* and torrent clients
- **remove_torrent** - Removes torrent by its hash both from *torrt* and torrent clients
- **register_torrent** - Registers torrent within *torrt* by its hash (for torrents already existing at torrent clients)
- **unregister_torrent** - Unregisters torrent from *torrt* by its hash

3.2.4 Notifications

- **configure_notifier** - Sets notification settings (SMTP server or telegram bot ID)
- **remove_notifier** - Remove notifier from *torrt* by its hash

3.2.5 Update procedure

- **walk** - Walks through registered torrents and performs automatic updates
- **set_walk_interval** - Sets an interval *in hours* between consecutive torrent updates checks

3.2.6 Bots configuration and run

- **configure_bot** - Sets bot settings (token and users allowed to add torrents)
- **run_bots** - Run bot processes. Note that this command starts process that never ends.

Note: More information on commands supported by **torrt** console application is available through *–help* command line switch

---

3.3 API

torrt exposes API so it can be used as an ordinary Python module.

3.3.1 Toolbox

Most commonly used functions are locate in toolbox.

3.3.2 Utils

Utility function and methods are also available.

3.3.3 Base RPC class

RPC classes should be implemented using this.

3.3.4 Base Tracker classes

Torrent tracker classes should be implemented using this.

3.3.5 Base Notification class

Notifier classes should be implemented using this.

3.3.6 Base Bots class

Bot classes should be implemented using this.

3.4 Telegram Bot

You can add new torrents via Telegram bot.

3.4.1 Register bot

1. Register your bot with BotFather as described in https://core.telegram.org/bots#6-botfather

---

Note: If you have already configured notifications with Telegram you don't need create a new bot. Use an existing one.

---

2. Install `python-telegram-bot` library to your python environment with

```
$ pip install python-telegram-bot
```

   You may install **torrt** with required dependencies with:

```
$ pip install torrt[telegram]
```

3. Configure **torrt** to use the bot:

```
$ torrt configure_bot telegram token=YOUR_TOKEN
```

   Restricts users (comma-separated) talking to the bot with option `allowed_users`:

```
$ torrt configure_bot telegram token=YOUR_TOKEN allowed_users=user1,user2
```

4. Create a new Telegram group and add the bot.

3.4.2 Listen to commands

Start listening to user commands:

```
$ torrt run_bots
```

Now your bot is ready to accept messages and fully functional.

Note: It is recommended to start `run_bots` process using a process management system (see `supervisord` or `systemd` configuration example below).

3.4.3 Talking to the bot

Bot supports a number of commands.

1. To start new conversation with bot use command:

```
/start
```

and follow further instructions. You can add new, list or remove already registered torrents.

Note: If you want to cancel current operation use */cancel* command.

2. Add a torrent using `/add` command (torrent is downloaded to a default directory.):

```
/add https://rutracker.org/forum/viewtopic.php?t=1234567
```

3. All registered torrents can be viewed with:

```
/list
```

4. To remove torrent use command:

```
/remove
```

5. To show all available commands use:

```
/help
```

3.4.4 Supervisor configuration

Here described how to configure and start torrt's Telegram bot with `supervisord`.

1. Install `supervisord` on your host as described at http://supervisord.org/installing.html
2. Create configuration file `torrt.conf` at `/etc/supervisor/conf.d/`:

```
[program:torrt]
directory=/tmp
command=PATH_TO_TORRT_SCRIPT run_bots
user=USER_ON_HOST
autostart=true
autorestart=true
```

   Replace `PATH_TO_TORRT_SCRIPT` with a location of **torrt** executable file and `USER_ON_HOST` with a user starting a process.

3. Start process with following commands:

```
# supervisorctl reread
# supervisorctl reload
# supervisorctl start torrt
```

3.4.5 Systemd user service

If you are running basically any modern Linux distribution you can run Telegram bot under your user with `systemd`, without having to deal with global system configuration.

1. Mark your user as the one allowed to 'linger'

```
# loginctl enable-linger `whoami`
```

2. Create service definition in your home directory:

```
$ mkdir -p ~/.config/systemd/user/
$ echo << "EOF" > ~/.config/systemd/user/torrt.service
[Unit]
Description=torrt bot

[Install]
WantedBy=default.target

[Service]
ExecStart=PATH_TO_TORRT_SCRIPT run_bots
EOF
```

Replace `PATH_TO_TORRT_SCRIPT` with a location of **torrt** executable file

3. Start Service

```
$ systemctl --user daemon-reload
$ systemctl --user start torrt
```

4. (Optional) Enable service autostart

```
$ systemctl --user enable torrt
```

CHAPTER 4

Get involved into torrt

Submit issues. If you spotted something weird in application behavior or want to propose a feature you can do that at https://github.com/idlesign/torrt/issues

Write code. If you are eager to participate in application development, fork it at https://github.com/idlesign/torrt, write your code, whether it should be a bugfix or a feature implementation, and make a pull request right from the forked project page.

Spread the word. If you have some tips and tricks or any other words in mind that you think might be of interest for the others — publish it.