



Mover.py Script

```
#!/usr/bin/env python3
# This standalone script is used to pause torrents older than last x days,
# run mover (in Unraid) and start torrents again once completed
import argparse
import logging
import os
import sys
import time
from datetime import datetime
from datetime import timedelta

# Configure logging
logging.basicConfig(
    level=logging.DEBUG,
    format="%(asctime)s - %(levelname)s - %(message)s",
    datefmt="%Y-%m-%d %H:%M:%S",
    handlers=[logging.StreamHandler(sys.stdout)],
)

parser = argparse.ArgumentParser(prog="Qbit Mover", description="Stop torrents and kick off Unraid mover process")
parser.add_argument("--host", help="qbittorrent host including port", required=True)
parser.add_argument("-u", "--user", help="qbittorrent user", default="admin")
parser.add_argument("-p", "--password", help="qbittorrent password", default="adminadmin")
parser.add_argument(
    "--cache-mount",
    "--cache_mount",
    help="Cache mount point in Unraid. This is used to additionally filter for only torrents that exists on the cache mount."
    "Use this option ONLY if you follow TRaSH Guides folder structure. (For default cache drive set this to /mnt/cache)",
    default=None,
)
parser.add_argument(
    "--days-from", "--days_from", help="Set Number of Days to stop torrents between two offsets", type=int,
    default=0
)
parser.add_argument(
    "--days-to", "--days_to", help="Set Number of Days to stop torrents between two offsets",
    type=int, default=2
)
parser.add_argument(
    "--mover-old",
    help="Use mover.old instead of mover. Useful if you're using the Mover Tuning Plugin",
    action="store_true",
    default=False,
)
parser.add_argument(
    "--status-filter",
    help="Define a status to limit which torrents to pause. Useful if you want to leave certain torrents unpaused.",
    choices=[
        "all",
        "downloading",
        "seeding",
        "completed",
        "paused",
    ],
)
```

```

        "stopped",
        "active",
        "inactive",
        "resumed",
        "running",
        "stalled",
        "stalled_uploading",
        "stalled_downloading",
        "checking",
        "moving",
        "errored",
    ],
    default="completed",
)
# --DEFINE VARIABLES--#

# --START SCRIPT--#
try:
    from qbittorrentapi import APIConnectionError
    from qbittorrentapi import Client
    from qbittorrentapi import LoginFailed
except ModuleNotFoundError:
    logging.error(
        'Requirements Error: qbittorrent-api not installed. Please install using the command "pip install qbittorrent-api"'
    )
    sys.exit(1)

def filter_torrents(torrent_list, timeoffset_from, timeoffset_to, cache_mount):
    result = []
    for torrent in torrent_list:
        if torrent.added_on >= timeoffset_to and torrent.added_on <= timeoffset_from:
            if not cache_mount or exists_in_cache(cache_mount, torrent.content_path):
                result.append(torrent)
        elif torrent.added_on < timeoffset_to:
            break
    return result

def exists_in_cache(cache_mount, content_path):
    cache_path = os.path.join(cache_mount, content_path.lstrip("/"))
    return os.path.exists(cache_path)

def stop_start_torrents(torrent_list, pause=True):
    for torrent in torrent_list:
        if pause:
            logging.info(f"Pausing: {torrent.name} [{torrent.added_on}]")
            torrent.pause()
        else:
            logging.info(f"Resuming: {torrent.name} [{torrent.added_on}]")
            torrent.resume()

if __name__ == "__main__":
    current = datetime.now()
    args = parser.parse_args()

    if args.days_from > args.days_to:
        raise ("Config Error: days_from must be set lower than days_to")

    try:
        client = Client(host=args.host, username=args.user, password=args.password)
    except LoginFailed:
        raise ("Qbittorrent Error: Failed to login. Invalid username/password.")
    except APIConnectionError:
        raise ("Qbittorrent Error: Unable to connect to the client.")
    except Exception:
        raise ("Qbittorrent Error: Unable to connect to the client.")

    timeoffset_from = current - timedelta(days=args.days_from)
    timeoffset_to = current - timedelta(days=args.days_to)
    torrent_list = client.torrents.info(status_filter=args.status_filter, sort="added_on", reverse=True)

```

```
torrents = filter_torrents(torrent_list, timeoffset_from.timestamp(), timeoffset_to.timestamp(),
args.cache_mount)

# Pause Torrents
logging.info(f"Pausing [{len(torrents)}] torrents from {args.days_from} - {args.days_to} days ago")
stop_start_torrents(torrents, True)
time.sleep(10)
# Or using mover tuning
if args.mover_old:
    # Start mover
    logging.info("Starting mover.old to move files in to array disks.")
    os.system("/usr/local/sbin/mover.old start")
else:
    # Start mover
    logging.info("Starting mover to move files in to array disks based on mover tuning preferences.")
    os.system("/usr/local/sbin/mover start")
# Start Torrents
logging.info(f"Resuming [{len(torrents)}] paused torrents from {args.days_from} - {args.days_to} days ago")
stop_start_torrents(torrents, False)
```