

---

# **TorrentStream**

***Release GPL2+***

**David Francos <[opensource@davidfrancos.net](mailto:opensource@davidfrancos.net)>**

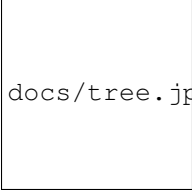
**Nov 17, 2020**



**CONTENTS:**

<b>1</b>	<b>Libtorrent made easy</b>	<b>3</b>
<b>2</b>	<b>Indices and tables</b>	<b>5</b>





docs/tree.jpeg



## LIBTORRENT MADE EASY

This is pythonic high-level libtorrent API, inspired on the for-humans trend set by Kenneth Reitz (<https://github.com/not-kennethreitz>).

TorrentStream is centered around the principle of *streaming* a torrent (sequential download, buffering and playing).

TorrentStream exposes a CLI command, intended as an example usage.



docs/torrentstream\_usage.png

Torrent objects are context managers that can clean up torrent content after you finish using them.

`add_torrent` method of a `TorrentSession` returns a `Torrent` object, thus can be used directly as a context manager.

```
async def stream_torrent(hash_torrent):
    session = TorrentSession()

    # By default this will cleanup torrent contents after playing
    with session.add_torrent(magnet_link=hash_torrent, remove_after=True) as torrent:
        # Force sequential mode
        torrent.sequential(True)

        # Wait for torrent to be started
        await torrent.wait_for('started')

        # Get first match of a media file
        try:
            media = next(a for a in torrent
                          if a.is_media and not 'sample' in a.path.lower())
        except StopIteration:
            raise Exception('Could not find a playable source')

        with timeout(5 * 60): # Abort if we can't fill 5% in 5 minutes
            await media.wait_for_completion(5)

        return await asyncio.gather(media.wait_for_completion(100),
                                     media.launch())
```





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`