trakt Documentation

Release 2.7.3

Jonathan Nappi

Mar 19, 2017

Contents

Release v2.7.3.

This module is designed to be a Pythonic interface to the Trakt.tv. REST API. The official documentation for which can be found here. trakt contains interfaces to all of the Trakt.tv functionality in an, ideally, easily scriptable fashion.

More information about getting started and accessing the information you thirst for can be found throughout the documentation below.

CHAPTER 1

Installation

There are two ways through which you can install trakt

Install Via Pip

To install with pip, just run this in your terminal:

```
$ pip install trakt
```

Get the code

trakt is available on GitHub.

You can either clone the public repository:

```
$ git clone git://github.com/moogar0880/PyTrakt.git
```

Download the tarball:

```
$ curl -OL https://github.com/moogar0880/PyTrakt/tarball/master
```

Or, download the zipball:

```
$ curl -OL https://github.com/moogar0880/PyTrakt/zipball/master
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

```
$ python setup.py install
```

CHAPTER 2

User Guide

Below you will find links to the generated documentation of the trakt module, including example usages.

Get Started

The main thing worth noting is how to authenticate via the trakt module. As of Trakt2.0, you need to generate an authentication token (API Key) in order to use this application. Regardless of whether you are a single user or a media center you'll want to hold on to this key for as long as it's good for. To generate this key you can interactively run PyTrakt's *init* function like detailed below:

Example Usage

The simplest way to generate an API key is to let *trakt.init* walk you through the process of generating one from scratch. Currently PyTrakt supports both OAuth and PIN auth methods (the default is PIN).

PIN Auth

You can authenticate to trakt via PIN Auth like so,

```
>>> import trakt
>>> trakt.APPLICATION_ID = 'MY_APPLICATION_ID'
>>> # to get an id for your app, visit https://trakt.tv/oauth/applications
>>> trakt.init()
If you do not have a Trakt.tv PIN, please visit the following url and log in to␣
↪generate one.
https://trakt.tv/pin/MY_APPLICATION_ID
Please enter your PIN:
>>> # Once you've pasted your PIN, you'll be completely authenticated and
>>> # ready to use PyTrakt
```

If you already have a Trakt PIN generated, you can provide it to the *init* function

```
>>> from trakt import init
>>> init('MY_PIN')
```

And you're all set

OAuth Auth

You can also initialize using OAuth authentication like so,

```
>>> from trakt import init
>>> import trakt.core
>>> trakt.core.AUTH_METHOD = trakt.core.OAUTH_AUTH  # Set the auth method to OAuth
>>> init('myusername')
If you do not have a client ID and secret. Please visit the following url to create␣
↪them.
http://trakt.tv/oauth/applications
Please enter your client id:
Please enter your client secret:

Please go here and authorize, <authorization_url>
Paste the Code returned here:
>>> # paste your code above and your access token will be returned
```

This example assumes that you haven't already created an OAuth application on Trakt yet. As of PyTrakt v2.0.0, if you have already registered your OAuth application on Trakt, you can specify your CLIENT_ID and CLIENT_SECRET to the *init* function and skip the first couple steps, like so

```
>>> from trakt import init
>>> init('myusername', client_id=my_client_id, client_secret=my_client_secret)
Please go here and authorize, <authorization_url>
Paste the Code returned here:
>>> # paste your code above and your access token will be returned
```

As of PyTrakt v2.0.0, *trakt.init* also exposes a *store* flag. This boolean flag can be used to store your PyTrakt API authentication data at the configurable *trakt.core.CONFIG_PATH* (the default is ~/.pytrakt.json). This will allow PyTrakt to dynamically load in your authorization settings when it runs, so that you won't have to worry about including that setup in your utilities. The store flag is set to *False* by default to appease those of you out there who are more security conscious. To set the *store* flag you can simply run *init* like so

```
>>> from trakt import init
>>> init('myusername', store=True)
```

Should you choose to store your credentials in another way and not to set the *store* flag, you will need to ensure that your application applies the following settings before attempting to interact with Trakt

- *trakt.core.api_key* * Note: api_key is deprecated in favor of OAUTH_TOKEN and will go away with the next major release
- *trakt.core.OAUTH_TOKEN*
- *trakt.core.CLIENT_ID*
- *trakt.core.CLIENT_SECRET*

These can be set like so

```
>>> import trakt
>>> trakt.core.OAUTH_TOKEN = my_oauth_token
>>> trakt.core.CLIENT_ID = my_client_id
>>> trakt.core.CLIENT_SECRET = my_client_secret
```

This is all of the authentication you'll need to perform to use the latest version of Trakt's API

Trakt Errors

All Trakt related errors that are worth processing. Note that 412 response codes are ignored because the only requests that this library sends out are guaranteed to have the application/json MIME type set.

exception trakt.errors.**TraktException**
Bases: exceptions.BaseException

Base Exception type for trakt module

http_code = None

message = None

exception trakt.errors.**BadRequestException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 401 return code is recieved

http_code = 400

message = "Bad Request - request couldn't be parsed"

exception trakt.errors.**OAuthException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 401 return code is recieved

http_code = 401

message = 'Unauthorized - OAuth must be provided'

exception trakt.errors.**ForbiddenException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 403 return code is recieved

http_code = 403

message = 'Forbidden - invalid API key or unapproved app'

exception trakt.errors.**NotFoundException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 404 return code is recieved

http_code = 404

message = 'Not Found - method exists, but no record found'

exception trakt.errors.**ConflictException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 409 return code is recieved

http_code = 409

message = 'Conflict - resource already created'

exception trakt.errors.**ProcessException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 422 return code is recieved

http_code = 422

message = 'Unprocessable Entity - validation errors'

exception trakt.errors.**RateLimitException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 429 return code is recieved

http_code = 429

message = 'Rate Limit Exceeded'

exception trakt.errors.**TraktInternalException**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 500 error is raised

http_code = 500

message = 'Internal Server Error'

exception trakt.errors.**TraktUnavailable**
Bases: *trakt.errors.TraktException*

TraktException type to be raised when a 503 error is raised

http_code = 503

message = 'Trakt Unavailable - server overloaded'

Calendars

Interfaces to all of the Calendar objects offered by the Trakt.tv API

class trakt.calendar.**Calendar**(*date=None*, *days=7*)
Base *Calendar* type serves as a foundation for other Calendar types

ext
construct the fully formatted url for this Calendar

url = None

class trakt.calendar.**PremiereCalendar**(*date=None*, *days=7*)
All shows premiering during the time period specified.

url = 'calendars/all/shows/new'

class trakt.calendar.**MyPremiereCalendar**(*date=None*, *days=7*)
Personalized calendar of all shows premiering during the time period specified.

url = 'calendars/my/shows/new'

class trakt.calendar.**ShowCalendar**(*date=None*, *days=7*)
TraktTV ShowCalendar

url = 'calendars/all/shows'

class trakt.calendar.**MyShowCalendar**(*date=None*, *days=7*)
Personalized TraktTV ShowCalendar

url = 'calendars/my/shows'

class trakt.calendar.**SeasonCalendar**(*date=None*, *days=7*)
TraktTV TV Show Season Premiere

url = 'calendars/all/shows/premieres'

class trakt.calendar.**MySeasonCalendar**(*date=None*, *days=7*)
Personalized TraktTV TV Show Season Premiere

url = 'calendars/my/shows/premieres'

class trakt.calendar.**MovieCalendar**(*date=None*, *days=7*)
TraktTV Movie Calendar. Returns all movies with a release date during the time period specified.

url = 'calendars/all/movies'

class trakt.calendar.**MyMovieCalendar**(*date=None*, *days=7*)
Personalized TraktTV Movie Calendar.

url = 'calendars/my/movies'

Example Usage

All four Calendar types *PremiereCalendar*, *ShowCalendar*, and *SeasonCalendar* and *MovieCalendar* behave similarly, the only fundamental difference is in the data they represent. They all accept optional date and days parameters which specify the start date and length of the Calendar. Below are some examples of these calendars in action.

```
>>> from trakt.calendar import PremiereCalendar
>>> p_calendar = PremiereCalendar(days=1)
>>> len(p_calendar)
21
>>> p_calendar
[<TVEpisode>: Return to Amish S1E1 Pilot,
 <TVEpisode>: The Next Food Network Star S10E10 Hollywood Calling,
 <TVEpisode>: Sladkaya Zhizn S1E1  1,
 <TVEpisode>: Ladies of London S1E1 ,
 <TVEpisode>: Longmire S3E3 The White Warrior,
 <TVEpisode>: Famous in 12 S1E1 Pilot,
 <TVEpisode>: Top Gear (US) S5E5 American Muscle,
 <TVEpisode>: Jennifer Falls S1E1 Pilot,
 <TVEpisode>: Orange Is the New Black S2E2 TBA,
 ...
```

You can also iterate over the Calendar itself

```
>>> for episode in p_calendar:
...     print(episode)
<TVEpisode>: Return to Amish S1E1 Pilot
<TVEpisode>: The Next Food Network Star S10E10 Hollywood Calling
<TVEpisode>: Sladkaya Zhizn S1E1  1
<TVEpisode>: Ladies of London S1E1
<TVEpisode>: Longmire S3E3 The White Warrior
<TVEpisode>: Famous in 12 S1E1 Pilot
<TVEpisode>: Top Gear (US) S5E5 American Muscle
<TVEpisode>: Jennifer Falls S1E1 Pilot
```

```
<TVEpisode>: Orange Is the New Black S2E2 TBA
...
```

Movies

Interfaces to all of the Movie objects offered by the Trakt.tv API

`trakt.movies.`**`dismiss_recommendation`**(**args*, ***kwargs*)
Dismiss the movie matching the specified criteria from showing up in recommendations.

`trakt.movies.`**`get_recommended_movies`**(**args*, ***kwargs*)
Get a list of *Movie*'s recommended based on your watching history and your friends. Results are returned with the top recommendation first.

`trakt.movies.`**`genres`**(**args*, ***kwargs*)
A list of all possible *Movie* Genres

`trakt.movies.`**`trending_movies`**(**args*, ***kwargs*)
All *Movie*'s being watched right now

`trakt.movies.`**`updated_movies`**(**args*, ***kwargs*)
Returns all movies updated since a timestamp. The server time is in PST. To establish a baseline timestamp, you can use the server/time method. It's recommended to store the timestamp so you can be efficient in using this method.

class `trakt.movies.`**`Release`**(*country*, *certification*, *release_date*, *note*, *release_type*)

`certification`
Alias for field number 1

`country`
Alias for field number 0

`note`
Alias for field number 3

`release_date`
Alias for field number 2

`release_type`
Alias for field number 4

class `trakt.movies.`**`Movie`**(*title*, *year=None*, *slug=None*, ***kwargs*)
A Class representing a Movie object

`add_to_collection`()
Add this *Movie* to your library.

`add_to_library`()
Add this *Movie* to your library.

`add_to_watchlist`()
Add this *Movie* to your watchlist

`aliases`
A list of `Alias` objects representing all of the other titles that this *Movie* is known by, and the countries where they go by their alternate titles

cast
All of the cast members that worked on this *Movie*

comment(*comment_body*, *spoiler=False*, *review=False*)
Add a comment (shout or review) to this Move on trakt.

comments
All comments (shouts and reviews) for this *Movie*. Most recent comments returned first.

crew
All of the crew members that worked on this *Movie*

dismiss()
Dismiss this movie from showing up in Movie Recommendations

ext
Base uri to retrieve basic information about this *Movie*

ext_full
Uri to retrieve all information about this *Movie*

get_releases(**args*, ***kwargs*)
Returns all :class:'Release's for a movie including country, certification, and release date.

Parameters **country_code** – The 2 character country code to search from

Returns a list of *Release* objects

get_translations(**args*, ***kwargs*)
Returns all :class:'Translation's for a movie, including language and translated values for title, tagline and overview.

Parameters **country_code** – The 2 character country code to search from

Returns a list of *Translation* objects

ids
Accessor to the trakt, imdb, and tmdb ids, as well as the trakt.tv slug

images
All of the artwork associated with this *Movie*

images_ext
Uri to retrieve additional image information

mark_as_seen(*watched_at=None*)
Add this *Movie*, watched outside of trakt, to your library.

mark_as_unseen()
Remove this *Movie*, watched outside of trakt, from your library.

people
A list of all of the People involved in this *Movie*, including both cast and crew

rate(*rating*)
Rate this *Movie* on trakt. Depending on the current users settings, this may also send out social updates to facebook, twitter, tumblr, and path.

ratings
Ratings (between 0 and 10) and distribution for a movie.

related
The top 10 *Movie*'s related to this *Movie*

remove_from_collection()
Remove this *Movie* from your library.

remove_from_library()
Remove this *Movie* from your library.

remove_from_watchlist()

scrobble(*progress*, *app_version*, *app_date*)
Notify trakt that the current user has finished watching a movie. This commits this *Movie* to the current users profile. You should use movie/watching prior to calling this method.

Parameters

- **progress** – % progress, integer 0-100. It is recommended to call the watching API every 15 minutes, then call the scrobble API near the end of the movie to lock it in.
- **app_version** – Version number of the media center, be as specific as you can including nightly build number, etc. Used to help debug your plugin.
- **app_date** – Build date of the media center. Used to help debug your plugin.

classmethod search(*title*, *year=None*)
Perform a search for a movie with a title matching *title*

Parameters

- **title** – The title to search for
- **year** – Optional year to limit results to

to_json()

watching_now
A list of all `User`'s watching a movie.

class `trakt.movies.`**Translation**(*title*, *overview*, *tagline*, *language*)

language
Alias for field number 3

overview
Alias for field number 1

tagline
Alias for field number 2

title
Alias for field number 0

Example Usage

The trakt.movies module has a handful of functionality for interfacing with the Movies hosted on Trakt.tv. The module has a few functions which you will need to be authenticated for. The *dismiss_recommendation()* function will block the specified movie from being shown in your recommended movies.

```
>>> from trakt.movies import dismiss_recommendation
>>> dismiss_recommendation(imdb_id='tt3139072', title='Son of Batman',
...                        year=2014)
```

This code snippet would prevent Son of Batman from appearing in your recommended movies list. Following the previous example you can use the *get_recommended_movies()* function to get the list of movies recommended for the currently authenticated user.

```
>>> from trakt.movies import get_recommended_movies
>>> all_movies = get_recommended_movies()
>>> all_movies
[<Movie>: b'The Dark Knight', <Movie>: b'WALLE', <Movie>: b'Up', <Movie>: b'Toy Story
↪',...
```

There's also a function to quickly rate a list of movies as the currently authenticated user.

```
>>> from trakt.movies import Movie, rate_movies
>>> rate_movies(all_movies, 'love')
```

There are a few properties that belong to the trakt.movies module as well.

```
>>> from trakt import movies
>>> movies.genres
[Genre(name='Action', slug='action'), Genre(name='Adventure', slug='adventure'),...
>>> movies.trending_movies
[<Movie>: b'The LEGO Movie', <Movie>: b'Non-Stop', <Movie>: b'Frozen', <Movie>: b
↪'RoboCop',...
>>> movies.updated_movies()
[]
```

Now to the Movie object. It's pretty straightforward, you provide a title and an optional year, and you will be returned an interface to that Movie on trakt.tv.

```
>>> from trakt.movies import Movie
>>> batman = Movie('Son of Batman')
>>> batman.overview
'Batman learns that he has a violent, unruly pre-teen son with Talia al Ghul named␣
↪Damian Wayne who is secretly being...
>>> batman.released_iso
'2014-04-20T07:00:00'
>>> batman.genres
[Genre(name='Action', slug='action'), Genre(name='Adventure', slug='adventure'),␣
↪Genre(name='Animation', slug='animation')]
>>> batman.add_to_library()
>>> batman.mark_as_seen()
```

People

Interfaces to all of the People objects offered by the Trakt.tv API

class trakt.people.**Person**(*name*, *slug=None*, ***kwargs*)

A Class representing a trakt.tv Person such as an Actor or Director

ext

ext_full

ext_movie_credits

ext_tv_credits

images
All of the artwork associated with this *Person*

images_ext

movie_credits
Return a collection of movie credits that this *Person* was a cast or crew member on

classmethod search (*name*, *year=None*)
Perform a search for an episode with a title matching *title*

Parameters

- **name** – The name of the person to search for
- **year** – Optional year to limit results to

tv_credits
Return a collection of TV Show credits that this *Person* was a cast or crew memeber on

class trakt.people.**ActingCredit** (*character*, *media*)
An individual credit for a *Person* who played a character in a Movie or TV Show

class trakt.people.**CrewCredit** (*job*, *media*)
An individual crew credit for a *Person* who had an off-screen job on a Movie or a TV Show

class trakt.people.**Credits** (***kwargs*)
A base type representing a *Person*'s credits for Movies or TV Shows

MEDIA_KEY = None

class trakt.people.**MovieCredits** (***kwargs*)
A collection of cast and crew credits for a Movie

MEDIA_KEY = 'movie'

class trakt.people.**TVCredits** (***kwargs*)
A collection of cast and crew credits for a TV Show

MEDIA_KEY = 'show'

Example Usage

The trakt.people module is pretty straightforward, it contains all of the tooling for collecting information about the cast and crew of TV Shows and Movies.

To collect information about a specific person, you need only to create a *Person* instance with their name as a parameter. Like so,

```
>>> from trakt.people import Person
>>> heyy = Person('Matthew McConaughey')
```

If you don't know the person's exact name, or believe it could be obscured by another person's name, you can also search

```
>>> heyy = Person.search('Matthew McConaughey')[0]
```

Once you have your *Person* instance, it's easy to collect information about them

```
>>> heyy.birthday
'1969-11-04
>>> heyy.birthplace
```

```
'Uvalde, Texas, USA'
>>> heyy.biography
'\u200bFrom Wikipedia, the free encyclopedia. \xa0\n\nMatthew David McConaughey (born␣
↪November 4, 1969) is an American actor.\n...'
>>> heyy.images
{'headshot': 'http://slurm.trakt.us/images/poster-dark.jpg'}
```

As of PyTrakt version 2.7.0, you can also access a *Person*'s Movie and television credits

```
>>> heyy.movie_credits.crew
{'production': [<CrewCredit> Producer - Surfer, Dude, <CrewCredit> Executive Producer␣
↪- Sahara]}
>>> heyy.movie_credits.cast
[<ActingCredit> Man in Black - The Dark Tower,
<ActingCredit> Arthur Brennan - The Sea of Trees,
<ActingCredit> Beetle (voice) - Kubo and the Two Strings,
...
```

Television

Interfaces to all of the TV objects offered by the Trakt.tv API

`trakt.tv.`**`dismiss_recommendation`**(**args*, ***kwargs*)
Dismiss the show matching the specified criteria from showing up in recommendations.

`trakt.tv.`**`get_recommended_shows`**(**args*, ***kwargs*)
Get a list of *TVShow*'s recommended based on your watching history and your friends. Results are returned with the top recommendation first.

`trakt.tv.`**`genres`**(**args*, ***kwargs*)
A list of all possible *TVShow* Genres

`trakt.tv.`**`popular_shows`**(**args*, ***kwargs*)

`trakt.tv.`**`trending_shows`**(**args*, ***kwargs*)
All *TVShow*'s being watched right now

`trakt.tv.`**`updated_shows`**(**args*, ***kwargs*)
All *TVShow*'s updated since *timestamp* (PST). To establish a baseline timestamp, you can use the server/time method. It's recommended to store the timestamp so you can be efficient in using this method.

class `trakt.tv.`**`TVShow`**(*title=''*, *slug=None*, ***kwargs*)
A Class representing a TV Show object

`add_to_collection`()
Add this *TVShow* to your library.

`add_to_library`()
Add this *TVShow* to your library.

`add_to_watchlist`()
Add this *TVShow* to your watchlist

`aliases`
A list of `Alias` objects representing all of the other titles that this *TVShow* is known by, and the countries where they go by their alternate titles

`cast`
All of the cast members that worked on this *TVShow*

comment(*comment_body*, *spoiler=False*, *review=False*)
Add a comment (shout or review) to this `Move` on trakt.

comments
All comments (shouts and reviews) for this *TVShow*. Most recent comments returned first.

crew
All of the crew members that worked on this *TVShow*

dismiss()
Dismiss this movie from showing up in Movie Recommendations

ext

ext_full

get_translations(**args*, ***kwargs*)
Returns all *Translation*'s for a movie, including language and translated values for title, tagline and overview.

Parameters **country_code** – The 2 character country code to search from

Returns a `list` of *Translation* objects

ids
Accessor to the trakt, imdb, and tmdb ids, as well as the trakt.tv slug

images
All of the artwork associated with this *TVShow*

images_ext
Uri to retrieve additional image information

mark_as_seen(*watched_at=None*)
Add this *TVShow*, watched outside of trakt, to your library.

mark_as_unseen()
Remove this *TVShow*, watched outside of trakt, from your library.

people
A `list` of all of the `People` involved in this *TVShow*, including both cast and crew

rate(*rating*)
Rate this *TVShow* on trakt. Depending on the current users settings, this may also send out social updates to facebook, twitter, tumblr, and path.

ratings
Ratings (between 0 and 10) and distribution for a movie.

related
The top 10 *TVShow*'s related to this *TVShow*

remove_from_collection()
Remove this *TVShow* from your library.

remove_from_library()
Remove this *TVShow* from your library.

remove_from_watchlist()

classmethod search(*title*, *year=None*)
Perform a search for the specified *title*

Parameters **title** – The title to search for

seasons
A list of *TVSeason* objects representing all of this show's seasons

to_json()

watching_now
A list of all User's watching a movie.

class trakt.tv.**TVEpisode**(*show*, *season*, *number=-1*, ***kwargs*)
Container for TV Episodes

add_to_collection()
Add this *TVEpisode* to your Trakt.tv library

add_to_library()
Add this *TVEpisode* to your Trakt.tv library

add_to_watchlist()
Add this *TVEpisode* to your watchlist

comment(*comment_body*, *spoiler=False*, *review=False*)
Add a comment (shout or review) to this *TVEpisode* on trakt.

comments
All comments (shouts and reviews) for this *TVEpisode*. Most recent comments returned first.

ext

ext_full

first_aired_date
Python datetime object representation of the first_aired date of this *TVEpisode*

get_description()
backwards compatible function that returns this *TVEpisode*'s overview '

ids
Accessor to the trakt, imdb, and tmdb ids, as well as the trakt.tv slug

images
All of the artwork associated with this *TVEpisode*

images_ext
Uri to retrieve additional image information

mark_as_seen(*watched_at=None*)
Mark this episode as seen

mark_as_unseen()
Remove this *TVEpisode* from your list of watched episodes

rate(*rating*)
Rate this *TVEpisode* on trakt. Depending on the current users settings, this may also send out social updates to facebook, twitter, tumblr, and path.

ratings
Ratings (between 0 and 10) and distribution for a movie.

remove_from_collection()
Remove this *TVEpisode* from your library

remove_from_library()
Remove this *TVEpisode* from your library

remove_from_watchlist()
Remove this *TVEpisode* from your watchlist

scrobble(*progress*, *app_version*, *app_date*)
Scrobble this *TVEpisode* via the TraktTV Api

Parameters

- **progress** – % progress, integer 0-100. It is recommended to call the watching API every 15 minutes, then call the scrobble API near the end of the movie to lock it in.
- **app_version** – Version number of the media center, be as specific as you can including nightly build number, etc. Used to help debug your plugin.
- **app_date** – Build date of the media center. Used to help debug your plugin.

classmethod search(*title*, *year=None*)
Perform a search for an episode with a title matching *title*

Parameters

- **title** – The title to search for
- **year** – Optional year to limit results to

to_json()
Return this *TVEpisode* as a trakt recognizable JSON object

watching_now
A list of all User's watching a movie.

class trakt.tv.**TVSeason**(*show*, *season=1*, *slug=None*, ***kwargs*)
Container for TV Seasons

add_to_collection()
Add this *TVSeason* to your library.

add_to_library()
Add this *TVSeason* to your library.

comments
All comments (shouts and reviews) for this *TVSeason*. Most recent comments returned first.

episodes
A list of *TVEpisode* objects representing all of the Episodes in this *TVSeason*. Because there is no "Get all episodes for a season" endpoint on the trakt api

ratings
Ratings (between 0 and 10) and distribution for a movie.

remove_from_collection()
Remove this *TVSeason* from your library.

remove_from_library()
Remove this *TVSeason* from your library.

to_json()
Return this *TVSeason* as a Trakt consumable API blob

watching_now
A list of all User's watching a movie.

class trakt.tv.**Translation**(*title*, *overview*, *language*)

language
Alias for field number 2

overview
Alias for field number 1

title
Alias for field number 0

Example Usage

The trakt.tv module has interfaces to all of the TV resources mentioned above and is almost a direct port of the trakt.movies module. It has the same interfaces to dismiss shows from recommendations, getting recommendations, rating a list of shows, rating a list of episodes, getting a list of valid show genres, a list of trending shows, and getting a list of recently updated shows and dealing with specific shows, seasons, and episodes.

For our first example let's start by grabbing a specific show

```
>>> from trakt.tv import TVShow
>>> it_crowd = TVShow('The IT Crowd')
```

Well that was pretty painless. Ok, now let's pull some data out of our `it_crowd` object

```
>>> it_crowd.people
[<Person>: Chris O'Dowd, <Person>: Katherine Parkinson, <Person>: None,
<Person>: Richard Ayoade, <Person>: Chris Morris, <Person>: Matt Berry, <Person>:␣
↪Noel Fielding]
>>> it_crowd.top_episodes
[<TVEpisode>: The IT Crowd S1E1 Yesterday's Jam, <TVEpisode>: The IT Crowd S1E2␣
↪Calamity Jen,
 <TVEpisode>: The IT Crowd S2E1 The Work Outing, <TVEpisode>: The IT Crowd S1E4 The␣
↪Red Door,...
>>> it_crowd.top_watchers
[<User>: b'Vaelek', <User>: b'Governa', <User>: b'shanos404', <User>: b'b_jammin666',
<User>: b'pavvoc', <User>: b'heartbraden', <User>: b'tressal', <User>: b'hherrera',...
>>> it_crowd.genres
[Genre(name='Comedy', slug='comedy')]
```

Now that we've gotten some information on the show, let's start doing something interesting and interacting with the API via the *TVShow*'s methods

```
>>> it_crowd.add_to_library()
>>> it_crowd.comment('Wow, I REALLY love this show')
>>> it_crowd.comment('They should never have given Jen the internet.',
                     spoiler=True, review=True)
```

Now that we've gotten some information on the show, let's dive down and get some information on the show's seasons and episodes

```
>>> s1 = it_crowd.seasons[1]
>>> s1.episodes
[<TVEpisode>: The IT Crowd S1E-1 Yesterday's Jam, <TVEpisode>: The IT Crowd S1E-1␣
↪Calamity Jen,
<TVEpisode>: The IT Crowd S1E-1 Fifty-Fifty, <TVEpisode>: The IT Crowd S1E-1 The Red␣
↪Door,
<TVEpisode>: The IT Crowd S1E-1 The Haunting of Bill Crouse, <TVEpisode>: The IT␣
↪Crowd S1E-1 Aunt Irma Visits]
>>> pilot = s1.episodes[0]
```

```
>>> pilot.title
'Yesterday's Jam'
>>> pilot.overview
'Jen is hired as the manager Reynholm Industries although she doesn't know the first␣
↪thing about computers.'
```

Users

class `trakt.users.`**`User`**(*username*)

A Trakt.tv User

`follow`()

Follow this `User`

`followers`

A list of all followers including the since timestamp which is when the relationship began. Protected users won't return any data unless you are friends. Any friends of the main user that are protected won't display data either.

`following`

A list of all user's this `User` follows including the since timestamp which is when the relationship began. Protected users won't return any data unless you are friends. Any friends of the main user that are protected won't display data either.

`friends`

A list of this `User`'s friends (a 2 way relationship where each user follows the other) including the since timestamp which is when the friendship began. Protected users won't return any data unless you are friends. Any friends of the main user that are protected won't display data either.

static `get_follower_requests`()

Return a list of all pending follower requests for the authenticated user

`get_list`(*title*)

Get the specified list from this `User`. Protected `User`'s won't return any data unless you are friends. To view your own private lists, you will need to authenticate as yourself.

`get_ratings`(**args*, ***kwargs*)

Get a user's ratings filtered by type. You can optionally filter for a specific rating between 1 and 10.

Parameters

- **`media_type`** – The type of media to search for. Must be one of 'movies', 'shows', 'seasons', 'episodes'
- **`rating`** – Optional rating between 1 and 10

`get_stats`(**args*, ***kwargs*)

Returns stats about the movies, shows, and episodes a user has watched and collected

`lists`

All custom lists for this `User`. Protected `User`'s won't return any data unless you are friends. To view your own private lists, you will need to authenticate as yourself.

`movie_collection`

All `Movie`'s in this `User`'s library collection. Collection items might include blu-rays, dvds, and digital downloads. Protected users won't return any data unless you are friends.

show_collection
All `TVShow`'s in this `User`'s library collection. Collection items might include blu-rays, dvds, and digital downloads. Protected users won't return any data unless you are friends.

unfollow()
Unfollow this `User`, if you already follow them

watched_movies
Watched profess for all `Movie`'s in this `User`'s collection.

watched_shows
Watched profess for all `TVShow`'s in this `User`'s collection.

watching
The `TVEpisode` or `Movie` this `User` is currently watching. If they aren't watching anything, a blank object will be returned. Protected users won't return any data unless you are friends.

watchlist_movies
Returns all watchlist movies of `User`.

watchlist_shows
Returns all watchlist shows of `User`.

class trakt.users.**UserList**(*user_name*, *slug=''*)
A list created by a Trakt.tv `User`

add_items(**args*, ***kwargs*)
Add *items* to this `UserList`, where items is an iterable

classmethod create(**args*, ***kwargs*)
Create a new custom class:*UserList*. *name* is the only required field, but the other info is recommended.

Parameters

- **name** – Name of the list.
- **description** – Description of this list.
- **privacy** – Valid values are 'private', 'friends', or 'public'
- **display_numbers** – Bool, should each item be numbered?
- **allow_comments** – Bool, are comments allowed?

delete_list(**args*, ***kwargs*)
Delete this `UserList`

get(**args*, ***kwargs*)
Returns a single custom `UserList`

Parameters **title** – Name of the list.

get_items(**args*, ***kwargs*)
A list of the list items using class instances instance types: movie, show, season, episode, person

like(**args*, ***kwargs*)
Like this `UserList`. Likes help determine popular lists. Only one like is allowed per list per user.

remove_items(**args*, ***kwargs*)
Remove *items* to this `UserList`, where items is an iterable

unlike(**args*, ***kwargs*)
Remove a like on this `UserList`.

Examples

To access a `User` all you need do is pass the `User`'s username to the `User`'s __init__ method

```
>>> from trakt.users import User
>>> my = User('moogar0880')
>>> my
'<User>: moogar0880'
```

Good, now we have a hold of the `User` object. Now we can get all of the information available from this trakt.tv `User`.

```
>>> my.gender
'male'
>>> my.location
'Newmarket NH'
>>> my.movie_collection
[<Movie>: b'2 Fast 2 Furious', <Movie>: b'A Beautiful Mind', <Movie>: b'A Bronx Tale',
↪ <Movie>: b"A Bug's Life", <Movie>: b'A Christmas Carol',...
```

Trakt Core

Objects, properties, and methods to be shared across other modules in the trakt package

class `trakt.core.`**`Airs`**(*day*, *time*, *timezone*)

day
Alias for field number 0

time
Alias for field number 1

timezone
Alias for field number 2

class `trakt.core.`**`Alias`**(*title*, *country*)

country
Alias for field number 1

title
Alias for field number 0

class `trakt.core.`**`Comment`**(*id*, *parent_id*, *created_at*, *comment*, *spoiler*, *review*, *replies*, *user*, *updated_at*, *likes*, *user_rating*)

comment
Alias for field number 3

created_at
Alias for field number 2

id
Alias for field number 0

likes
Alias for field number 9

parent_id
Alias for field number 1

replies
Alias for field number 6

review
Alias for field number 5

spoiler
Alias for field number 4

updated_at
Alias for field number 8

user
Alias for field number 7

user_rating
Alias for field number 10

class trakt.core.**Genre**(*name*, *slug*)

name
Alias for field number 0

slug
Alias for field number 1

trakt.core.**init**(**args*, ***kwargs*)
Run the auth function specified by *AUTH_METHOD*

trakt.core.**BASE_URL = 'https://api-v2launch.trakt.tv/'**
The base url for the Trakt API. Can be modified to run against different Trakt.tv environments

trakt.core.**CLIENT_ID = None**
The Trakt.tv OAuth Client ID for your OAuth Application

trakt.core.**CLIENT_SECRET = None**
The Trakt.tv OAuth Client Secret for your OAuth Application

trakt.core.**REDIRECT_URI = 'urn:ietf:wg:oauth:2.0:oob'**
The OAuth2 Redirect URI for your OAuth Application

trakt.core.**HEADERS = {'trakt-api-version': '2', 'Content-Type': 'application/json'}**
Default request HEADERS

trakt.core.**CONFIG_PATH = '/home/docs/.pytrakt.json'**
Default path for where to store your trakt.tv API authentication information

trakt.core.**OAUTH_TOKEN = None**
Your personal Trakt.tv OAUTH Bearer Token

trakt.core.**PIN_AUTH = 'PIN'**
Flag used to enable Trakt PIN authentication

trakt.core.**OAUTH_AUTH = 'OAUTH'**
Flag used to enable Trakt OAuth authentication

trakt.core.**AUTH_METHOD = 'PIN'**
The currently enabled authentication method. Default is PIN_AUTH

trakt.core.**APPLICATION_ID** = **None**
The ID of the application to register with, when using PIN authentication

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index