Python PlexAPI Documentation

M.Shepanski

Mar 22, 2026

OVERVIEW

CHAPTER

ONE

PYTHON-PLEXAPI

1.1 Overview

Unofficial Python bindings for the Plex API. Our goal is to match all capabilities of the official Plex Web Client. A few of the many features we currently support are:

- Navigate local or remote shared libraries.
- Perform library actions such as scan, analyze, empty trash.
- Remote control and play media on connected clients, including *Controlling Sonos speakers*
- Listen in on all Plex Server notifications.

1.2 Installation & Documentation

```
pip install plexapi
```

Install extra features:

```
pip install plexapi[alert]  # Install with dependencies required for plexapi.alert
pip install plexapi[jwt]    # Install with dependencies required for Plex JWT␣
↪authentication
```

Documentation can be found at Read the Docs.

Join our Discord for support and discussion.

1.3 Getting a PlexServer Instance

There are two types of authentication. If you are running on a separate network or using Plex Users you can log into MyPlex to get a PlexServer instance. An example of this is below. NOTE: Servername below is the name of the server

(not the hostname and port). If logged into Plex Web you can see the server name in the top left above your available libraries.

```
from plexapi.myplex import MyPlexAccount
account = MyPlexAccount('<USERNAME>', '<PASSWORD>')
plex = account.resource('<SERVERNAME>').connect()  # returns a PlexServer instance
```

If you want to avoid logging into MyPlex and you already know your auth token string, you can use the PlexServer object directly as above, by passing in the baseurl and auth token directly.

```
from plexapi.server import PlexServer
baseurl = 'http://plexserver:32400'
token = '2ffLuB84dqLswk9skLos'
plex = PlexServer(baseurl, token)
```

1.4 Usage Examples

```
# Example 1: List all unwatched movies.
movies = plex.library.section('Movies')
for video in movies.search(unwatched=True):
    print(video.title)
```

```
# Example 2: Mark all Game of Thrones episodes as played.
plex.library.section('TV Shows').get('Game of Thrones').markPlayed()
```

```
# Example 3: List all clients connected to the Server.
for client in plex.clients():
    print(client.title)
```

```
# Example 4: Play the movie Cars on another client.
# Note: Client must be on same network as server.
cars = plex.library.section('Movies').get('Cars')
client = plex.client("Michael's iPhone")
client.playMedia(cars)
```

```
# Example 5: List all content with the word 'Game' in the title.
for video in plex.search('Game'):
    print(f'{video.title} ({video.TYPE})')
```

```
# Example 6: List all movies directed by the same person as Elephants Dream.
movies = plex.library.section('Movies')
elephants_dream = movies.get('Elephants Dream')
director = elephants_dream.directors[0]
for movie in movies.search(None, director=director):
    print(movie.title)
```

```
# Example 7: List files for the latest episode of The 100.
last_episode = plex.library.section('TV Shows').get('The 100').episodes()[-1]
for part in last_episode.iterParts():
    print(part.file)
```

```
# Example 8: Get audio/video/all playlists
for playlist in plex.playlists():
    print(playlist.title)
```

```
# Example 9: Rate the 100 four stars.
plex.library.section('TV Shows').get('The 100').rate(8.0)
```

1.5 Controlling Sonos speakers

To control Sonos speakers directly using Plex APIs, the following requirements must be met:

1. Active Plex Pass subscription
2. Sonos account linked to Plex account
3. Plex remote access enabled

Due to the design of Sonos music services, the API calls to control Sonos speakers route through https://sonos.plex.tv and back via the Plex server's remote access. Actual media playback is local unless networking restrictions prevent the Sonos speakers from connecting to the Plex server directly.

```
from plexapi.myplex import MyPlexAccount
from plexapi.server import PlexServer

baseurl = 'http://plexserver:32400'
token = '2ffLuB84dqLswk9skLos'

account = MyPlexAccount(token)
server = PlexServer(baseurl, token)

# List available speakers/groups
for speaker in account.sonos_speakers():
    print(speaker.title)

# Obtain PlexSonosPlayer instance
speaker = account.sonos_speaker("Kitchen")

album = server.library.section('Music').get('Stevie Wonder').album('Innervisions')

# Speaker control examples
speaker.playMedia(album)
speaker.pause()
speaker.setVolume(10)
speaker.skipNext()
```

1.6 Running tests over PlexAPI

Use:

```
tools/plex-boostraptest.py
```

with appropriate arguments and add this new server to a shared user which username is defined in environment variable *SHARED_USERNAME*. It uses official docker image to create a proper instance.

For skipping the docker and reuse a existing server use

```
python plex-bootstraptest.py --no-docker --username USERNAME --password PASSWORD --
↪server-name NAME-OF-YOUR-SEVER
```

Also in order to run most of the tests you have to provide some environment variables:

- *PLEXAPI_AUTH_SERVER_BASEURL* containing an URL to your Plex instance, e.g. *http://127.0.0.1:32400* (without trailing slash)
- *PLEXAPI_AUTH_MYPLEX_USERNAME* and *PLEXAPI_AUTH_MYPLEX_PASSWORD* with your MyPlex username and password accordingly

After this step you can run tests with following command:

```
py.test tests -rxXs --ignore=tests/test_sync.py
```

Some of the tests in main test-suite require a shared user in your account (e.g. *test_myplex_users*, *test_myplex_updateFriend*, etc.), you need to provide a valid shared user's username to get them running you need to provide the username of the shared user as an environment variable *SHARED_USERNAME*. You can enable a Guest account and simply pass *Guest* as *SHARED_USERNAME* (or just create a user like *plexapitest* and play with it).

To be able to run tests over Mobile Sync api you have to some some more environment variables, to following values exactly:

- PLEXAPI_HEADER_PROVIDES='controller,sync-target'
- PLEXAPI_HEADER_PLATFORM=iOS
- PLEXAPI_HEADER_PLATFORM_VERSION=11.4.1
- PLEXAPI_HEADER_DEVICE=iPhone

And finally run the sync-related tests:

```
py.test tests/test_sync.py -rxXs
```

1.7 Common Questions

Why are you using camelCase and not following PEP8 guidelines?

This API reads XML documents provided by MyPlex and the Plex Server. We decided to conform to their style so that the API variable names directly match with the provided XML documents.

Why don't you offer feature XYZ?

This library is meant to be a wrapper around the XML pages the Plex server provides. If we are not providing an API that is offered in the XML pages, please let us know! – Adding additional features beyond that should be done outside the scope of this library.

What are some helpful links if trying to understand the raw Plex API?

- https://github.com/plexinc/plex-media-player/wiki/Remote-control-API
- https://forums.plex.tv/discussion/104353/pms-web-api-documentation
- https://github.com/Arcanemagus/plex-api/wiki

CHAPTER

TWO

CONFIGURATION

Python-PlexAPI will work fine without any configuration. However, sometimes there are things you may wish to alter for more control of the default behavior. The default configuration file path is `~/.config/plexapi/config.ini` which can be overridden by setting the environment variable `PLEXAPI_CONFIG_PATH` with the file path you desire. All configuration variables in this file are optional. An example config.ini file may look like the following with all possible value specified.

```
# ~/.config/plexapi/config.ini
[plexapi]
container_size = 50
timeout = 30
timezone = false

[auth]
myplex_username = johndoe
myplex_password = kodi-stinks
server_baseurl = http://127.0.0.1:32400
server_token = XBHSMSJSDJ763JSm
client_baseurl = http://127.0.0.1:32433
client_token = BDFSLCNSNL789FH7

[header]
identifier = 0x485b314307f3L
platorm = Linux
platform_version = 4.4.0-62-generic
product = PlexAPI
version = 3.0.0

[log]
backup_count = 3
format = %(asctime)s %(module)12s:%(lineno)-4s %(levelname)-9s %(message)s
level = INFO
path = ~/.config/plexapi/plexapi.log
rotate_bytes = 512000
show_secrets = false
```

2.1 Environment Variables

All configuration values can be set or overridden via environment variables. The environment variable names are in all upper case and follow the format `PLEXAPI_<SECTION>_<NAME>`. For example, if you wish to set the log path via an environment variable, you may specify: *PLEXAPI_LOG_PATH="/tmp/plexapi.log"*

2.2 Section [plexapi] Options

container_size
Default max results to return in on single search page. Looping through result pages is done internally by the API. Therefore, tuning this setting will not affect usage of plexapi. However, it help improve performance for large media collections (default: 50).

timeout
Timeout in seconds to use when making requests to the Plex Media Server or Plex Client resources (default: 30).

autoreload
By default PlexAPI will automatically `reload()` any `PlexPartialObject` when accessing a missing attribute. When this option is set to *false*, automatic reloading will be disabled and `reload()` must be called manually (default: true).

enable_fast_connect
By default Plex will be trying to connect with all available connection methods simultaneously, combining local and remote addresses, http and https, and be waiting for all connection to establish (or fail due to timeout / any other error), this can take long time when you're trying to connect to your Plex Server outside of your home network.

When the options is set to *true* the connection procedure will be aborted with first successfully established connection (default: false).

timezone
Controls whether `toDatetime()` returns timezone-aware datetime objects.

- *false* (default): keep naive datetime objects (backward compatible).
- *true* or *local*: use the local machine timezone.
- IANA timezone string (for example *UTC* or *America/New_York*): use that timezone.

This feature relies on Python's `zoneinfo.ZoneInfo` and the availability of IANA tzdata on the system. On platforms without system tzdata (notably Windows), you may need to install the `tzdata` Python package for IANA timezone strings (such as `America/New_York`) to work as expected. Toggling this option may break comparisons between aware and naive datetimes.

2.3 Section [auth] Options

myplex_username
Default MyPlex (plex.tv) username to use when creating a new `MyPlexAccount` object. Specifying this along with `auth.myplex_password` allow you to more easily connect to your account and remove the need to hard code the username and password in any supplemental scripts you may write. To create an account object using these values you may simply specify `account = MyPlexAccount()` without any arguments (default: None).

myplex_password
Default MyPlex (plex.tv) password to use when creating a new `MyPlexAccount` object. See *auth.myplex_password* for more information and example usage (default: None).

WARNING: When specifying a password or token in the configuration file, be sure lock it down (permission 600) to ensure no other users on the system can read them. Or better yet, only specify sensitive values as a local environment variables.

server_baseurl
Default baseurl to use when creating a new `PlexServer` object. Specifying this along with `auth.server_token` allow you to more easily connect to a server and remove the need to hard code the baseurl and token in any supplemental scripts you may write. To create a server object using these values you may simply specify `plex = PlexServer()` without any arguments (default: None).

server_token
Default token to use when creating a new *PlexServer* object. See *auth.server_baseurl* for more information and example usage (default: None).

WARNING: When specifying a password or token in the configuration file, be sure lock it down (permission 600) to ensure no other users on the system can read them. Or better yet, only specify sensitive values as a local environment variables.

client_baseurl
Default baseurl to use when creating a new *PlexClient* object. Specifying this along with `auth.client_token` allow you to more easily connect to a client and remove the need to hard code the baseurl and token in any supplemental scripts you may write. To create a client object using these values you may simply specify `client = PlexClient()` without any arguments (default: None).

client_token
Default token to use when creating a new *PlexClient* object. See *auth.client_baseurl* for more information and example usage (default: None).

WARNING: When specifying a password or token in the configuration file, be sure lock it down (permission 600) to ensure no other users on the system can read them. Or better yet, only specify sensitive values as a local environment variables.

2.4 Section [header] Options

device
Header value used for X_PLEX_DEVICE to all Plex server and Plex client requests. Example devices include: iPhone, FireTV, Linux (default: *result of platform.uname()[0]*).

device_name
Header value used for X_PLEX_DEVICE_NAME to all Plex server and Plex client requests. Example device names include: hostname or phone name (default: *result of platform.uname()[1]*).

identifier
Header value used for X_PLEX_IDENTIFIER to all Plex server and Plex client requests. This is generally a UUID, serial number, or other number unique id for the device (default: *result of hex(uuid.getnode())*).

language
Header value used for X_PLEX_LANGUAGE to all Plex server and Plex client requests. This is an ISO 639-1 language code (default: en).

platform
Header value used for X_PLEX_PLATFORM to all Plex server and Plex client requests. Example platforms include: iOS, MacOSX, Android, LG (default: *result of platform.uname()[0]*).

platform_version
Header value used for X_PLEX_PLATFORM_VERSION to all Plex server and Plex client requests. This is generally the server or client operating system version: 4.3.1, 10.6.7, 3.2 (default: *result of platform.uname()[2]*).

product
Header value used for X_PLEX_PRODUCT to all Plex server and Plex client requests. This is the Plex application name: Laika, Plex Media Server, Media Link (default: PlexAPI).

provides
Header value used for X_PLEX_PROVIDES to all Plex server and Plex client requests This is generally one or more of: controller, player, server (default: PlexAPI).

version
Header value used for X_PLEX_VERSION to all Plex server and Plex client requests. This is the Plex application version (default: plexapi.VERSION).

2.5 Section [log] Options

backup_count
Number backup log files to keep before rotating out old logs (default 3).

format
Log file format to use for plexapi logging. (default: '%(asctime)s %(module)12s:%(lineno)-4s %(levelname)-9s %(message)s'). Ref: https://docs.python.org/2/library/logging.html#logrecord-attributes

level
Log level to use when for plexapi logging (default: INFO).

path
File path to save plexapi logs to. If not specified, plexapi will not save logs to an output file (default: None).

rotate_bytes
Max size of the log file before rotating logs to a backup file (default: 512000 equals 0.5MB).

show_secrets
By default Plex will hide all passwords and token values when logging. Set this to 'true' to enable logging these secrets. This should only be done on a private server and only enabled when needed (default: false).

CHAPTER

THREE

ALERT PLEXAPI.ALERT

class plexapi.alert.**AlertListener**(*server*, *callback: ~typing.Callable = None*, *callbackError: ~typing.Callable = None*, *ws_socket: <module 'socket' from '/home/docs/.asdf/installs/python/3.10.17/lib/python3.10/socket.py'> = None*)

Bases: `Thread`

Creates a websocket connection to the PlexServer to optionally receive alert notifications. These often include messages from Plex about media scans as well as updates to currently running Transcode Sessions. This class implements threading.Thread, therefore to start monitoring alerts you must call .start() on the object once it's created. When calling *PlexServer.startAlertListener()*, the thread will be started for you.

Known *state*-values for timeline entries, with identifier=`com.plexapp.plugins.library`:

0
: The item was created

1
: Reporting progress on item processing

2
: Matching the item

3
: Downloading the metadata

4
: Processing downloaded metadata

5
: The item processed

9
: The item deleted

When metadata agent is not set for the library processing ends with state=1.

Parameters

- **server** (*PlexServer*) – PlexServer this listener is connected to.
- **callback** (*func*) – Callback function to call on received messages. The callback function will be sent a single argument 'data' which will contain a dictionary of data received from the server. `def my_callback(data): ...`
- **callbackError** (*func*) – Callback function to call on errors. The callback function will be sent a single argument 'error' which will contain the Error object. `def my_callback(error): ...`

- **ws_socket** (*socket*) – Socket to use for the connection. If not specified, a new socket will be created.

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

stop()

Stop the AlertListener thread. Once the notifier is stopped, it cannot be directly started again. You must call *startAlertListener()* from a PlexServer instance.

CHAPTER

FOUR

AUDIO PLEXAPI.AUDIO

class plexapi.audio.**Audio**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexPartialObject*, *PlayedUnplayedMixin*

Base class for all audio objects including *Artist*, *Album*, and *Track*.

Variables

- **addedAt** (*datetime*) – Datetime the item was added to the library.
- **art** (*str*) – URL to artwork image (/library/metadata/<ratingKey>/art/<artid>).
- **artBlurHash** (*str*) – BlurHash string for artwork image.
- **distance** (*float*) – Sonic Distance of the item from the seed item.
- **fields** (List<*Field*>) – List of field objects.
- **guid** (*str*) – Plex GUID for the artist, album, or track (plex://artist/5d07bcb0403c64029053ac4c).
- **images** (List<*Image*>) – List of image objects.
- **index** (*int*) – Plex index number (often the track number).
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **lastRatedAt** (*datetime*) – Datetime the item was last rated.
- **lastViewedAt** (*datetime*) – Datetime the item was last played.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **librarySectionKey** (*str*) – *LibrarySection* key.
- **librarySectionTitle** (*str*) – *LibrarySection* title.
- **listType** (*str*) – Hardcoded as 'audio' (useful for search filters).
- **moods** (List<*Mood*>) – List of mood objects.
- **musicAnalysisVersion** (*int*) – The Plex music analysis version for the item.
- **ratingKey** (*int*) – Unique key identifying the item.
- **summary** (*str*) – Summary of the artist, album, or track.
- **thumb** (*str*) – URL to thumbnail image (/library/metadata/<ratingKey>/thumb/<thumbid>).
- **thumbBlurHash** (*str*) – BlurHash string for thumbnail image.
- **title** (*str*) – Name of the artist, album, or track (Jason Mraz, We Sing, Lucky, etc.).
- **titleSort** (*str*) – Title to use when sorting (defaults to title).

- **type** (*str*) – 'artist', 'album', or 'track'.
- **updatedAt** (*datetime*) – Datetime the item was updated.
- **userRating** (*float*) – Rating of the item (0.0 - 10.0) equaling (0 stars - 5 stars).
- **viewCount** (*int*) – Count of times the item was played.

url(*part*)

Returns the full URL for the audio item. Typically used for getting a specific track.

property hasSonicAnalysis

Returns True if the audio has been sonically analyzed.

sync(*bitrate*, *client=None*, *clientId=None*, *limit=None*, *title=None*)

Add current audio (artist, album or track) as sync item for specified device. See *sync()* for possible exceptions.

Parameters

- **bitrate** (*int*) – maximum bitrate for synchronized music, better use one of MUSIC_BITRATE_* values from the module *sync*.
- **client** (*MyPlexDevice*) – sync destination, see *sync()*.
- **clientId** (*str*) – sync destination, see *sync()*.
- **limit** (*int*) – maximum count of items to sync, unlimited if *None*.
- **title** (*str*) – descriptive title for the new *SyncItem*, if empty the value would be generated from metadata of current media.

Returns
an instance of created syncItem.

Return type
SyncItem

sonicallySimilar(*limit: int | None = None*, *maxDistance: float | None = None*, ***kwargs*) → List[TAudio]

Returns a list of sonically similar audio items.

Parameters

- **limit** (*int*) – Maximum count of items to return. Default 50 (server default)
- **maxDistance** (*float*) – Maximum distance between tracks, 0.0 - 1.0. Default 0.25 (server default).
- ****kwargs** – Additional options passed into *fetchItems()*.

Returns
list of sonically similar audio items.

Return type
List[*Audio*]

class plexapi.audio.**Artist**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Audio*, *ArtistMixins*

Represents a single Artist.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'artist'

- **albumSort** (*int*) – Setting that indicates how albums are sorted for the artist (-1 = Library default, 0 = Newest first, 1 = Oldest first, 2 = By name).
- **audienceRating** (*float*) – Audience rating.
- **collections** (List<*Collection*>) – List of collection objects.
- **countries** (List<*Country*>) – List country objects.
- **genres** (List<*Genre*>) – List of genre objects.
- **guids** (List<*Guid*>) – List of guid objects.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **labels** (List<*Label*>) – List of label objects.
- **locations** (*List<str>*) – List of folder paths where the artist is found on disk.
- **rating** (*float*) – Artist rating (7.9; 9.8; 8.1).
- **similar** (List<*Similar*>) – List of similar objects.
- **styles** (List<*Style*>) – List of style objects.
- **theme** (*str*) – URL to theme resource (/library/metadata/<ratingkey>/theme/<themeid>).
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.

album(*title*)

Returns the *Album* that matches the specified title.

Parameters

title (*str*) – Title of the album to return.

albums(***kwargs*)

Returns a list of *Album* objects by the artist.

track(*title=None*, *album=None*, *track=None*)

Returns the *Track* that matches the specified title.

Parameters

- **title** (*str*) – Title of the track to return.
- **album** (*str*) – Album name (default: None; required if title not specified).
- **track** (*int*) – Track number (default: None; required if title not specified).

Raises

BadRequest – If title or album and track parameters are missing.

tracks(***kwargs*)

Returns a list of *Track* objects by the artist.

get(*title=None*, *album=None*, *track=None*)

Alias of *track()*.

download(*savepath=None*, *keep_original_name=False*, *subfolders=False*, ***kwargs*)

Download all tracks from the artist. See *download()* for details.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **keep_original_name** (*bool*) – True to keep the original filename otherwise a friendlier filename is generated.

- **subfolders** (*bool*) – True to separate tracks in to album folders.
- ****kwargs** – Additional options passed into `getStreamURL()`.

popularTracks()

Returns a list of *Track* popular tracks by the artist.

station()

Returns a *Playlist* artist radio station or *None*.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.audio.**Album**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Audio*, *AlbumMixins*

Represents a single Album.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'album'
- **audienceRating** (*float*) – Audience rating.
- **collections** (List<*Collection*>) – List of collection objects.
- **formats** (List<*Format*>) – List of format objects.
- **genres** (List<*Genre*>) – List of genre objects.
- **guids** (List<*Guid*>) – List of guid objects.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **labels** (List<*Label*>) – List of label objects.
- **leafCount** (*int*) – Number of items in the album view.
- **loudnessAnalysisVersion** (*int*) – The Plex loudness analysis version level.
- **originallyAvailableAt** (*datetime*) – Datetime the album was released.
- **parentGuid** (*str*) – Plex GUID for the album artist (plex://artist/5d07bcb0403c64029053ac4c).
- **parentKey** (*str*) – API URL of the album artist (/library/metadata/<parentRatingKey>).
- **parentRatingKey** (*int*) – Unique key identifying the album artist.
- **parentTheme** (*str*) – URL to artist theme resource (/library/metadata/<parentRatingkey>/theme/<themeid>).
- **parentThumb** (*str*) – URL to album artist thumbnail image (/library/metadata/<parentRatingKey>/thumb/<thumbid>).
- **parentTitle** (*str*) – Name of the album artist.
- **rating** (*float*) – Album rating (7.9; 9.8; 8.1).
- **studio** (*str*) – Studio that released the album.
- **styles** (List<*Style*>) – List of style objects.
- **subformats** (List<*Subformat*>) – List of subformat objects.
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.

- **viewedLeafCount** (*int*) – Number of items marked as played in the album view.
- **year** (*int*) – Year the album was released.

track(*title=None*, *track=None*)

Returns the *Track* that matches the specified title.

Parameters

- **title** (*str*) – Title of the track to return.
- **track** (*int*) – Track number (default: None; required if title not specified).

Raises

BadRequest – If title or track parameter is missing.

tracks(***kwargs*)

Returns a list of *Track* objects in the album.

get(*title=None*, *track=None*)

Alias of *track()*.

artist()

Return the album's *Artist*.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)

Download all tracks from the album. See *download()* for details.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **keep_original_name** (*bool*) – True to keep the original filename otherwise a friendlier filename is generated.
- ****kwargs** – Additional options passed into `getStreamURL()`.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.audio.**Track**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Audio*, *Playable*, *TrackMixins*

Represents a single Track.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'track'
- **audienceRating** (*float*) – Audience rating.
- **chapters** (List<*Chapter*>) – List of Chapter objects.
- **chapterSource** (*str*) – Unknown
- **collections** (List<*Collection*>) – List of collection objects.
- **duration** (*int*) – Length of the track in milliseconds.
- **genres** (List<*Genre*>) – List of genre objects.
- **grandparentArt** (*str*) – URL to album artist artwork (/library/metadata/<grandparentRatingKey>/art/<artid>).

- **grandparentGuid** (*str*) – Plex GUID for the album artist (plex://artist/5d07bcb0403c64029053ac4c).
- **grandparentKey** (*str*) – API URL of the album artist (/library/metadata/<grandparentRatingKey>).
- **grandparentRatingKey** (*int*) – Unique key identifying the album artist.
- **grandparentTheme** (*str*) – URL to artist theme resource (/library/metadata/<grandparentRatingkey>/theme/<themeid>). (/library/metadata/<grandparentRatingkey>/theme/<themeid>).
- **grandparentThumb** (*str*) – URL to album artist thumbnail image (/library/metadata/<grandparentRatingKey>/thumb/<thumbid>).
- **grandparentTitle** (*str*) – Name of the album artist for the track.
- **guids** (List<*Guid*>) – List of guid objects.
- **labels** (List<*Label*>) – List of label objects.
- **media** (List<*Media*>) – List of media objects.
- **originalTitle** (*str*) – The artist for the track.
- **parentGuid** (*str*) – Plex GUID for the album (plex://album/5d07cd8e403c640290f180f9).
- **parentIndex** (*int*) – Disc number of the track.
- **parentKey** (*str*) – API URL of the album (/library/metadata/<parentRatingKey>).
- **parentRatingKey** (*int*) – Unique key identifying the album.
- **parentThumb** (*str*) – URL to album thumbnail image (/library/metadata/<parentRatingKey>/thumb/<thumbid>).
- **parentTitle** (*str*) – Name of the album for the track.
- **primaryExtraKey** (*str*)
- **rating** (*float*) – Track rating (7.9; 9.8; 8.1).
- **ratingCount** (*int*) – Number of listeners who have scrobbled this track, as reported by Last.fm.
- **skipCount** (*int*) – Number of times the track has been skipped.
- **sourceURI** (*str*) – Remote server URI (server://<machineIdentifier>/com.plexapp.plugins.library) (remote playlist item only).
- **viewOffset** (*int*) – View offset in milliseconds.
- **year** (*int*) – Year the track was released.

property locations

This does not exist in plex xml response but is added to have a common interface to get the locations of the track.

Returns
List<str> of file paths where the track is found on disk.

property trackNumber

Returns the track number.

album()

Return the track's *Album*.

artist()

Return the track's *Artist*.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

sonicAdventure(*to: TTrack*, ***kwargs: Any*) → list[TTrack]

Returns a sonic adventure from the current track to the specified track.

Parameters

- **to** (*Track*) – The target track for the sonic adventure.
- ****kwargs** – Additional options passed into *sonicAdventure()*.

Returns
list of tracks in the sonic adventure.

Return type
List[*Track*]

class plexapi.audio.**TrackSession**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexSession*, *Track*

Represents a single Track session loaded from *sessions()*.

class plexapi.audio.**TrackHistory**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexHistory*, *Track*

Represents a single Track history entry loaded from *history()*.

CHAPTER

FIVE

BASE PLEXAPI.BASE

class plexapi.base.**cached_data_property**(*func*)

Bases: cached_property

Caching for PlexObject data properties.

This decorator creates properties that cache their values with automatic invalidation on data changes.

class plexapi.base.**PlexObjectMeta**(*name*, *bases*, *attrs*)

Bases: type

Metaclass for PlexObject to handle cached_data_properties.

class plexapi.base.**PlexObject**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: object

Base class for all Plex objects.

Parameters

- **server** (*PlexServer*) – PlexServer this client is connected to (optional)
- **data** (*ElementTree*) – Response from PlexServer used to build this object (optional).
- **initpath** (*str*) – Relative path requested when retrieving specified *data* (optional).
- **parent** (*PlexObject*) – The parent object that this object is built from (optional).

fetchItems(*ekey*, *cls=None*, *container_start=None*, *container_size=None*, *maxresults=None*, *params=None*, ***kwargs*)

Load the specified key to find and build all items with the specified tag and attrs.

Parameters

- **ekey** (*str or List<int>*) – API URL path in Plex to fetch items from. If a list of ints is passed in, the key will be translated to /library/metadata/<key1,key2,key3>. This allows fetching multiple items only knowing their key-ids.
- **cls** (*PlexObject*) – If you know the class of the items to be fetched, passing this in will help the parser ensure it only returns those items. By default we convert the xml elements with the best guess PlexObjects based on tag and type attrs.
- **etag** (*str*) – Only fetch items with the specified tag.
- **container_start** (*None, int*) – offset to get a subset of the data
- **container_size** (*None, int*) – How many items in data
- **maxresults** (*int, optional*) – Only return the specified number of results.
- **params** (*dict, optional*) – Any additional params to add to the request.

- ****kwargs** (*dict*) – Optionally add XML attribute to filter the items. See the details below for more info.

Filtering XML Attributes

Any XML attribute can be filtered when fetching results. Filtering is done before the Python objects are built to help keep things speedy. For example, passing in `viewCount=0` will only return matching items where the view count is `0`. Note that case matters when specifying attributes. Attributes further down in the XML tree can be filtered by *prepending* the attribute with each element tag `Tag__`.

Examples

```
fetchItem(ekey, viewCount=0)
fetchItem(ekey, contentRating="PG")
fetchItem(ekey, Genre__tag="Animation")
fetchItem(ekey, Media__videoCodec="h265")
fetchItem(ekey, Media__Part__container="mp4)
```

Note that because some attribute names are already used as arguments to this function, such as `tag`, you may still reference the attr tag by prepending an underscore. For example, passing in `_tag='foobar'` will return all items where `tag='foobar'`.

Using PlexAPI Operators

Optionally, PlexAPI operators can be specified by *appending* it to the end of the attribute for more complex lookups. For example, passing in `viewCount__gte=0` will return all items where `viewCount >= 0`.

List of Available Operators:

- `__contains`: Value contains specified arg.
- `__endswith`: Value ends with specified arg.
- `__exact`: Value matches specified arg.
- `__exists` (*bool*): Value is or is not present in the attrs.
- `__gt`: Value is greater than specified arg.
- `__gte`: Value is greater than or equal to specified arg.
- `__icontains`: Case insensitive value contains specified arg.
- `__iendswith`: Case insensitive value ends with specified arg.
- `__iexact`: Case insensitive value matches specified arg.
- `__in`: Value is in a specified list or tuple.
- `__iregex`: Case insensitive value matches the specified regular expression.
- `__istartswith`: Case insensitive value starts with specified arg.
- `__lt`: Value is less than specified arg.
- `__lte`: Value is less than or equal to specified arg.
- `__regex`: Value matches the specified regular expression.
- `__startswith`: Value starts with specified arg.

Examples

```
fetchItem(ekey, viewCount__gte=0)
fetchItem(ekey, Media__container__in=["mp4", "mkv"])
fetchItem(ekey, guid__regex=r"com\.plexapp\.agents\.(imdb|themoviedb)://|tt\d+")
fetchItem(ekey, guid__id__regex=r"(imdb|tmdb|tvdb)://")
fetchItem(ekey, Media__Part__file__startswith="D:\Movies")
```

fetchItem(*ekey*, *cls=None*, ***kwargs*)

Load the specified key to find and build the first item with the specified tag and attrs. If no tag or attrs are specified then the first item in the result set is returned.

Parameters

- **ekey** (*str or int*) – Path in Plex to fetch items from. If an int is passed in, the key will be translated to /library/metadata/<key>. This allows fetching an item only knowing its key-id.
- **cls** (*PlexObject*) – If you know the class of the items to be fetched, passing this in will help the parser ensure it only returns those items. By default we convert the xml elements with the best guess PlexObjects based on tag and type attrs.
- **etag** (*str*) – Only fetch items with the specified tag.
- ****kwargs** (*dict*) – Optionally add XML attribute to filter the items. See *fetchItems()* for more details on how this is used.

findItems(*data*, *cls=None*, *initpath=None*, *rtag=None*, ***kwargs*)

Load the specified data to find and build all items with the specified tag and attrs. See *fetchItem()* for more details on how this is used.

findItem(*data*, *cls=None*, *initpath=None*, *rtag=None*, ***kwargs*)

Load the specified data to find and build the first items with the specified tag and attrs. See *fetchItem()* for more details on how this is used.

firstAttr(**attrs*)

Return the first attribute in attrs that is not None.

listAttrs(*data*, *attr*, *rtag=None*, ***kwargs*)

Return a list of values from matching attribute.

reload(*key=None*, ***kwargs*)

Reload the data for this object.

Parameters

- **key** (*string, optional*) – Override the key to reload.
- ****kwargs** (*dict*) – A dictionary of XML include parameters to include/exclude or override. See *PlexPartialObject* for all the available include parameters. Set parameter to True to include and False to exclude.

Example

```
from plexapi.server import PlexServer
plex = PlexServer('http://localhost:32400', token='xxxxxxxxxxxxxxxxxxxx')

# Search results are partial objects.
```

(continues on next page)

(continued from previous page)

```
movie = plex.library.section('Movies').get('Cars')
movie.isPartialObject()  # Returns True

# Partial reload of the movie without a default include parameter.
# The movie object will remain as a partial object.
movie.reload(includeMarkers=False)
movie.isPartialObject()  # Returns True

# Full reload of the movie with all default include parameters.
# The movie object will be a full object.
movie.reload()
movie.isFullObject()  # Returns True

# Full reload of the movie with all default and extra include parameter.
# Including `checkFiles` will tell the Plex server to check if the file
# still exists and is accessible.
# The movie object will be a full object.
movie.reload(checkFiles=True)
movie.isFullObject()  # Returns True
```

class plexapi.base.**PlexPartialObject**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Not all objects in the Plex listings return the complete list of elements for the object. This object will allow you to assume each object is complete, and if the specified value you request is None it will fetch the full object automatically and update itself.

analyze()

Tell Plex Media Server to performs analysis on it this item to gather information. Analysis includes:

- **Gather Media Properties: All of the media you add to a Library has**
  properties that are useful to know–whether it's a video file, a music track, or one of your photos (container, codec, resolution, etc).
- **Generate Default Artwork: Artwork will automatically be grabbed from a**
  video file. A background image will be pulled out as well as a smaller image to be used for poster/thumbnail type purposes.
- **Generate Video Preview Thumbnails: Video preview thumbnails are created,**
  if you have that feature enabled. Video preview thumbnails allow graphical seeking in some Apps. It's also used in the Plex Web App Now Playing screen to show a graphical representation of where playback is. Video preview thumbnails creation is a CPU-intensive process akin to transcoding the file.
- **Generate intro video markers: Detects show intros, exposing the**
  'Skip Intro' button in clients.

isFullObject()

Returns True if this is already a full object. A full object means all attributes were populated from the api path representing only this item. For example, the search result for a movie often only contain a portion of the attributes a full object (main url) for that movie would contain.

isPartialObject()

Returns True if this is not a full object.

isLocked(*field: str*)

Returns True if the specified field is locked, otherwise False.

Parameters
field (*str*) – The name of the field.

edit(***kwargs*)

Edit an object. Note: This is a low level method and you need to know all the field/tag keys. See *EditFieldMixin* and *EditTagsMixin* for individual field and tag editing methods.

Parameters
kwargs (*dict*) – Dict of settings to edit.

Example

```
edits = {
    'type': 1,
    'id': movie.ratingKey,
    'title.value': 'A new title',
    'title.locked': 1,
    'summary.value': 'This is a summary.',
    'summary.locked': 1,
    'collection[0].tag.tag': 'A tag',
    'collection.locked': 1}
}
movie.edit(**edits)
```

batchEdits()

Enable batch editing mode to save API calls. Must call *saveEdits()* at the end to save all the edits. See *EditFieldMixin* and *EditTagsMixin* for individual field and tag editing methods.

Example

```
# Batch editing multiple fields and tags in a single API call
Movie.batchEdits()
Movie.editTitle('A New Title').editSummary('A new summary').editTagline('A new␣
↪tagline') \
    .addCollection('New Collection').removeGenre('Action').addLabel('Favorite')
Movie.saveEdits()
```

saveEdits()

Save all the batch edits. The object needs to be reloaded manually, if required. See *batchEdits()* for details.

refresh()

Refreshing a Library or individual item causes the metadata for the item to be refreshed, even if it already has metadata. You can think of refreshing as "update metadata for the requested item even if it already has some". You should refresh a Library or individual item if:

- You've changed the Library Metadata Agent.
- **You've added "Local Media Assets" (such as artwork, theme music, external** subtitle files, etc.)
- You want to freshen the item posters, summary, etc.
- There's a problem with the poster image that's been downloaded.

- **Items are missing posters or other downloaded information. This is possible if** the refresh process is interrupted (the Server is turned off, internet connection dies, etc).

`section`()

Returns the *LibrarySection* this item belongs to.

`delete`()

Delete a media element. This has to be enabled under settings > server > library in plex webui.

`history`(*maxresults=None*, *mindate=None*)

Get Play History for a media item.

Parameters

- **maxresults** (*int*) – Only return the specified number of results (optional).
- **mindate** (*datetime*) – Min datetime to return results from.

`getWebURL`(*base=None*)

Returns the Plex Web URL for a media item.

Parameters

base (*str*) – The base URL before the fragment (`#!`). Default is https://app.plex.tv/desktop.

`playQueue`(**args*, ***kwargs*)

Returns a new *PlayQueue* from this media item. See *create()* for available parameters.

`class plexapi.base.Playable`

Bases: `object`

This is a mixin to store functions specific to media that is Playable. Things were getting mixed up a bit when dealing with Shows, Season, Artists, Albums which are all not playable.

Variables

- **playlistItemID** (*int*) – Playlist item ID (only populated for *Playlist* items).
- **playQueueItemID** (*int*) – PlayQueue item ID (only populated for `PlayQueue` items).

`getStreamURL`(***kwargs*)

Returns a stream url that may be used by external applications such as VLC.

Parameters

****kwargs** (*dict*) – optional parameters to manipulate the playback when accessing the stream. A few known parameters include: maxVideoBitrate, videoResolution offset, copyts, protocol, mediaIndex, partIndex, platform.

Raises

Unsupported – When the item doesn't support fetching a stream URL.

`iterParts`()

Iterates over the parts of this media item.

`videoStreams`()

Returns a list of `videoStream` objects for all MediaParts.

`audioStreams`()

Returns a list of *AudioStream* objects for all MediaParts.

`subtitleStreams`()

Returns a list of *SubtitleStream* objects for all MediaParts.

lyricStreams()

Returns a list of *LyricStream* objects for all MediaParts.

play(*client*)

Start playback on the specified client.

Parameters

client (*PlexClient*) – Client to start playing on.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)

Downloads the media item to the specified location. Returns a list of filepaths that have been saved to disk.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **keep_original_name** (*bool*) – True to keep the original filename otherwise a friendlier filename is generated. See filenames below.
- ****kwargs** (*dict*) – Additional options passed into `getStreamURL()` to download a transcoded stream, otherwise the media item will be downloaded as-is and saved to disk.

Filenames

- Movie: `<title> (<year>)`
- Episode: `<show title> - s00e00 - <episode title>`
- Track: `<artist title> - <album title> - 00 - <track title>`
- Photo: `<photoalbum title> - <photo/clip title>` or `<photo/clip title>`

updateProgress(*time*, *state='stopped'*)

Set the watched progress for this video.

Note that setting the time to 0 will not work. Use *markPlayed()* or *markUnplayed()* to achieve that goal.

Parameters

- **time** (*int*) – milliseconds watched
- **state** (*string*) – state of the video, default 'stopped'

updateTimeline(*time*, *state='stopped'*, *duration=None*)

Set the timeline progress for this video.

Parameters

- **time** (*int*) – milliseconds watched
- **state** (*string*) – state of the video, default 'stopped'
- **duration** (*int*) – duration of the item

class plexapi.base.**PlexSession**

Bases: `object`

This is a mixin to store functions specific to media that is a Plex Session.

Variables

- **live** (*bool*) – True if this is a live tv session.
- **player** (*PlexClient*) – PlexClient object for the session.
- **session** (*Session*) – Session object for the session if the session is using bandwidth (None otherwise).

- **sessionKey** (*int*) – The session key for the session.
- **transcodeSession** (*TranscodeSession*) – TranscodeSession object if item is being transcoded (None otherwise).

property user

Returns the *MyPlexAccount* object (for admin) or *MyPlexUser* object (for users) for this session.

reload()

Reload the data for the session. Note: This will return the object as-is if the session is no longer active.

source()

Return the source media object for the session.

stop(*reason=''*)

Stop playback for the session.

Parameters
reason (*str*) – Message displayed to the user for stopping playback.

class plexapi.base.**PlexHistory**

Bases: object

This is a mixin to store functions specific to media that is a Plex history item.

Variables

- **accountID** (*int*) – The associated *SystemAccount* ID.
- **deviceID** (*int*) – The associated *SystemDevice* ID.
- **historyKey** (*str*) – API URL (/status/sessions/history/<historyID>).
- **viewedAt** (*datetime*) – Datetime item was last watched.

source()

Return the source media object for the history entry or None if the media no longer exists on the server.

delete()

Delete the history entry.

class plexapi.base.**MediaContainer**(*server: PlexServer*, *data: Element*, **args: PlexObjectT*, *initpath: str | None = None*, *parent:* PlexObject *| None = None*)

Bases: Generic[PlexObjectT], List[PlexObjectT], *PlexObject*

Represents a single MediaContainer.

Variables

- **TAG** (*str*) – 'MediaContainer'
- **allowSync** (*int*) – Sync/Download is allowed/disallowed for feature.
- **augmentationKey** (*str*) – API URL (/library/metadata/augmentations/<augmentationKey>).
- **identifier** (*str*) – "com.plexapp.plugins.library"
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **librarySectionTitle** (*str*) – *LibrarySection* title.
- **librarySectionUUID** (*str*) – *LibrarySection* UUID.
- **mediaTagPrefix** (*str*) – "/system/bundle/media/flags/"

- **mediaTagVersion** (*int*) – Unknown
- **offset** (*int*) – The offset of current results.
- **size** (*int*) – The number of items in the hub.
- **totalSize** (*int*) – The total number of items for the query.

extend(*_MediaContainer__iterable: Iterable[PlexObjectT] | MediaContainerT*) → None

Extend list by appending elements from the iterable.

CHAPTER

SIX

CLIENT PLEXAPI.CLIENT

class plexapi.client.**PlexClient**(*server=None*, *data=None*, *initpath=None*, *baseurl=None*, *identifier=None*, *token=None*, *connect=True*, *session=None*, *timeout=None*, *parent=None*)

Bases: *PlexObject*

Main class for interacting with a Plex client. This class can connect directly to the client and control it or proxy commands through your Plex Server. To better understand the Plex client API's read this page: https://github.com/plexinc/plex-media-player/wiki/Remote-control-API

Parameters

- **server** (*PlexServer*) – PlexServer this client is connected to (optional).
- **data** (*ElementTree*) – Response from PlexServer used to build this object (optional).
- **initpath** (*str*) – Path used to generate data.
- **baseurl** (*str*) – HTTP URL to connect directly to this client.
- **identifier** (*str*) – The resource/machine identifier for the desired client. May be necessary when connecting to a specific proxied client (optional).
- **token** (*str*) – X-Plex-Token used for authentication (optional).
- **session** (Session) – requests.Session object if you want more control (optional).
- **timeout** (*int*) – timeout in seconds on initial connect to client (default config.TIMEOUT).

Variables

- **TAG** (*str*) – 'Player'
- **key** (*str*) – '/resources'
- **device** (*str*) – Best guess on the type of device this is (PS, iPhone, Linux, etc).
- **deviceClass** (*str*) – Device class (pc, phone, etc).
- **machineIdentifier** (*str*) – Unique ID for this device.
- **model** (*str*) – Unknown
- **platform** (*str*) – Unknown
- **platformVersion** (*str*) – Description
- **product** (*str*) – Client Product (Plex for iOS, etc).
- **protocol** (*str*) – Always seems ot be 'plex'.
- **protocolCapabilities** (*list<str>*) – List of client capabilities (navigation, playback, timeline, mirror, playqueues).

- **protocolVersion** (*str*) – Protocol version (1, future proofing?)
- **server** (*PlexServer*) – Server this client is connected to.
- **session** (Session) – Session object used for connection.
- **state** (*str*) – Unknown
- **title** (*str*) – Name of this client (Johns iPhone, etc).
- **token** (*str*) – X-Plex-Token used for authentication
- **vendor** (*str*) – Unknown
- **version** (*str*) – Device version (4.6.1, etc).
- **_baseurl** (*str*) – HTTP address of the client.
- **_token** (*str*) – Token used to access this client.
- **_session** (*obj*) – Requests session object used to access this client.
- **_proxyThroughServer** (*bool*) – Set to True after calling *proxyThroughServer()* (default False).

connect(*timeout=None*)

Alias of reload as any subsequent requests to this client will be made directly to the device even if the object attributes were initially populated from a PlexServer.

reload()

Alias to self.connect().

proxyThroughServer(*value=True*, *server=None*)

Tells this PlexClient instance to proxy all future commands through the PlexServer. Useful if you do not wish to connect directly to the Client device itself.

Parameters

value (*bool*) – Enable or disable proxying (optional, default True).

Raises

Unsupported – Cannot use client proxy with unknown server.

query(*path*, *method=None*, *headers=None*, *timeout=None*, ***kwargs*)

Main method used to handle HTTPS requests to the Plex client. This method helps by encoding the response to utf-8 and parsing the returned XML into and ElementTree object. Returns None if no data exists in the response.

sendCommand(*command*, *proxy=None*, ***params*)

Convenience wrapper around *query()* to more easily send simple commands to the client. Returns an ElementTree object containing the response.

Parameters

- **command** (*str*) – Command to be sent in for format '<controller>/<command>'.
- **proxy** (*bool*) – Set True to proxy this command through the PlexServer.
- ****params** (*dict*) – Additional GET parameters to include with the command.

Raises

Unsupported – When we detect the client doesn't support this capability.

url(*key*, *includeToken=False*)

Build a URL string with proper token argument. Token will be appended to the URL if either includeToken is True or CONFIG.log.show_secrets is 'true'.

contextMenu()

Open the context menu on the client.

goBack()

Navigate back one position.

goToHome()

Go directly to the home screen.

goToMusic()

Go directly to the playing music panel.

moveDown()

Move selection down a position.

moveLeft()

Move selection left a position.

moveRight()

Move selection right a position.

moveUp()

Move selection up a position.

nextLetter()

Jump to next letter in the alphabet.

pageDown()

Move selection down a full page.

pageUp()

Move selection up a full page.

previousLetter()

Jump to previous letter in the alphabet.

select()

Select element at the current position.

toggleOSD()

Toggle the on screen display during playback.

goToMedia(*media*, ***params*)

Navigate directly to the specified media page.

Parameters

- **media** (*Media*) – Media object to navigate to.
- ****params** (*dict*) – Additional GET parameters to include with the command.

pause(*mtype='video'*)

Pause the currently playing media type.

Parameters

mtype (*str*) – Media type to take action against (music, photo, video).

play(*mtype='video'*)

Start playback for the specified media type.

Parameters
mtype (*str*) – Media type to take action against (music, photo, video).

refreshPlayQueue(*playQueueID*, *mtype='video'*)

Refresh the specified Playqueue.

Parameters

- **playQueueID** (*str*) – Playqueue ID.
- **mtype** (*str*) – Media type to take action against (music, photo, video).

seekTo(*offset*, *mtype='video'*)

Seek to the specified offset (ms) during playback.

Parameters

- **offset** (*int*) – Position to seek to (milliseconds).
- **mtype** (*str*) – Media type to take action against (music, photo, video).

skipNext(*mtype='video'*)

Skip to the next playback item.

Parameters
mtype (*str*) – Media type to take action against (music, photo, video).

skipPrevious(*mtype='video'*)

Skip to previous playback item.

Parameters
mtype (*str*) – Media type to take action against (music, photo, video).

skipTo(*key*, *mtype='video'*)

Skip to the playback item with the specified key.

Parameters

- **key** (*str*) – Key of the media item to skip to.
- **mtype** (*str*) – Media type to take action against (music, photo, video).

stepBack(*mtype='video'*)

Step backward a chunk of time in the current playback item.

Parameters
mtype (*str*) – Media type to take action against (music, photo, video).

stepForward(*mtype='video'*)

Step forward a chunk of time in the current playback item.

Parameters
mtype (*str*) – Media type to take action against (music, photo, video).

stop(*mtype='video'*)

Stop the currently playing item.

Parameters
mtype (*str*) – Media type to take action against (music, photo, video).

setRepeat(*repeat*, *mtype='video'*)

Enable repeat for the specified playback items.

Parameters

- **repeat** (*int*) – Repeat mode (0=off, 1=repeatone, 2=repeatall).
- **mtype** (*str*) – Media type to take action against (music, photo, video).

setShuffle(*shuffle*, *mtype='video'*)

Enable shuffle for the specified playback items.

Parameters

- **shuffle** (*int*) – Shuffle mode (0=off, 1=on)
- **mtype** (*str*) – Media type to take action against (music, photo, video).

setVolume(*volume*, *mtype='video'*)

Enable volume for the current playback item.

Parameters

- **volume** (*int*) – Volume level (0-100).
- **mtype** (*str*) – Media type to take action against (music, photo, video).

setAudioStream(*audioStreamID*, *mtype='video'*)

Select the audio stream for the current playback item (only video).

Parameters

- **audioStreamID** (*str*) – ID of the audio stream from the media object.
- **mtype** (*str*) – Media type to take action against (music, photo, video).

setSubtitleStream(*subtitleStreamID*, *mtype='video'*)

Select the subtitle stream for the current playback item (only video).

Parameters

- **subtitleStreamID** (*str*) – ID of the subtitle stream from the media object.
- **mtype** (*str*) – Media type to take action against (music, photo, video).

setVideoStream(*videoStreamID*, *mtype='video'*)

Select the video stream for the current playback item (only video).

Parameters

- **videoStreamID** (*str*) – ID of the video stream from the media object.
- **mtype** (*str*) – Media type to take action against (music, photo, video).

playMedia(*media*, *offset=0*, ***params*)

Start playback of the specified media item. See also:

Parameters

- **media** (*Media*) – Media item to be played back (movie, music, photo, playlist, playqueue).
- **offset** (*int*) – Number of milliseconds at which to start playing with zero representing the beginning (default 0).

- ****params** (*dict*) – Optional additional parameters to include in the playback request. See also: https://github.com/plexinc/plex-media-player/wiki/Remote-control-API#modified-commands

setParameters(*volume=None*, *shuffle=None*, *repeat=None*, *mtype='video'*)

Set multiple playback parameters at once.

Parameters

- **volume** (*int*) – Volume level (0-100; optional).
- **shuffle** (*int*) – Shuffle mode (0=off, 1=on; optional).
- **repeat** (*int*) – Repeat mode (0=off, 1=repeatone, 2=repeatall; optional).
- **mtype** (*str*) – Media type to take action against (optional music, photo, video).

setStreams(*audioStreamID=None*, *subtitleStreamID=None*, *videoStreamID=None*, *mtype='video'*)

Select multiple playback streams at once.

Parameters

- **audioStreamID** (*str*) – ID of the audio stream from the media object.
- **subtitleStreamID** (*str*) – ID of the subtitle stream from the media object.
- **videoStreamID** (*str*) – ID of the video stream from the media object.
- **mtype** (*str*) – Media type to take action against (optional music, photo, video).

timelines(*wait=0*)

Poll the client's timelines, create, and return timeline objects. Some clients may not always respond to timeline requests, believe this to be a Plex bug.

property timeline

Returns the active timeline object.

isPlayingMedia(*includePaused=True*)

Returns True if any media is currently playing.

Parameters

includePaused (*bool*) – Set True to treat currently paused items as playing (optional; default True).

class plexapi.client.**ClientTimeline**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Get the timeline's attributes.

CHAPTER

SEVEN

COLLECTION PLEXAPI.COLLECTION

class plexapi.collection.**Collection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexPartialObject*, *CollectionMixins*

Represents a single Collection.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'collection'
- **addedAt** (*datetime*) – Datetime the collection was added to the library.
- **art** (*str*) – URL to artwork image (/library/metadata/<ratingKey>/art/<artid>).
- **artBlurHash** (*str*) – BlurHash string for artwork image.
- **audienceRating** (*float*) – Audience rating.
- **childCount** (*int*) – Number of items in the collection.
- **collectionFilterBasedOnUser** (*int*) – Which user's activity is used for the collection filtering.
- **collectionMode** (*int*) – How the items in the collection are displayed.
- **collectionPublished** (*bool*) – True if the collection is published to the Plex homepage.
- **collectionSort** (*int*) – How to sort the items in the collection.
- **content** (*str*) – The filter URI string for smart collections.
- **contentRating** (*str) Content rating (PG-13; NR; TV-G*)
- **fields** (List<*Field*>) – List of field objects.
- **guid** (*str*) – Plex GUID for the collection (collection://XXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXX).
- **images** (List<*Image*>) – List of image objects.
- **index** (*int*) – Plex index number for the collection.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **labels** (List<*Label*>) – List of label objects.
- **lastRatedAt** (*datetime*) – Datetime the collection was last rated.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **librarySectionKey** (*str*) – *LibrarySection* key.

- **librarySectionTitle** (*str*) – *LibrarySection* title.
- **maxYear** (*int*) – Maximum year for the items in the collection.
- **minYear** (*int*) – Minimum year for the items in the collection.
- **rating** (*float*) – Collection rating (7.9; 9.8; 8.1).
- **ratingCount** (*int*) – The number of ratings.
- **ratingKey** (*int*) – Unique key identifying the collection.
- **smart** (*bool*) – True if the collection is a smart collection.
- **subtype** (*str*) – Media type of the items in the collection (movie, show, artist, or album).
- **summary** (*str*) – Summary of the collection.
- **theme** (*str*) – URL to theme resource (/library/metadata/<ratingkey>/theme/<themeid>).
- **thumb** (*str*) – URL to thumbnail image (/library/metadata/<ratingKey>/thumb/<thumbid>).
- **thumbBlurHash** (*str*) – BlurHash string for thumbnail image.
- **title** (*str*) – Name of the collection.
- **titleSort** (*str*) – Title to use when sorting (defaults to title).
- **type** (*str*) – 'collection'
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.
- **updatedAt** (*datetime*) – Datetime the collection was updated.
- **userRating** (*float*) – Rating of the collection (0.0 - 10.0) equaling (0 stars - 5 stars).

property listType

Returns the listType for the collection.

property metadataType

Returns the type of metadata in the collection.

property isVideo

Returns True if this is a video collection.

property isAudio

Returns True if this is an audio collection.

property isPhoto

Returns True if this is a photo collection.

filters()

Returns the search filter dict for smart collection. The filter dict be passed back into *search()* to get the list of items.

section()

Returns the *LibrarySection* this collection belongs to.

item(*title*)

Returns the item in the collection that matches the specified title.

Parameters
title (*str*) – Title of the item to return.

Raises
plexapi.exceptions.NotFound – When the item is not found in the collection.

items()

Returns a list of all items in the collection.

visibility()

Returns the *ManagedHub* for this collection.

get(*title*)

Alias to `item()`.

filterUserUpdate(*user=None*)

Update the collection filtering user advanced setting.

Parameters
user (*str*) – One of the following values: "admin" (Always the server admin user), "user" (User currently viewing the content)

Example

```
collection.updateMode(user="user")
```

modeUpdate(*mode=None*)

Update the collection mode advanced setting.

Parameters
mode (*str*) – One of the following values: "default" (Library default), "hide" (Hide Collection), "hideItems" (Hide Items in this Collection), "showItems" (Show this Collection and its Items)

Example

```
collection.updateMode(mode="hide")
```

sortUpdate(*sort=None*)

Update the collection order advanced setting.

Parameters
sort (*str*) – One of the following values: "release" (Order Collection by release dates), "alpha" (Order Collection alphabetically), "custom" (Custom collection order)

Example

```
collection.sortUpdate(sort="alpha")
```

addItems(*items*)

Add items to the collection.

Parameters
items (*List*) – List of *Audio*, *Video*, or *Photo* objects to be added to the collection.

Raises
plexapi.exceptions.BadRequest – When trying to add items to a smart collection.

removeItems(*items*)

Remove items from the collection.

Parameters
items (*List*) – List of *Audio*, *Video*, or *Photo* objects to be removed from the collection.

Raises
plexapi.exceptions.BadRequest – When trying to remove items from a smart collection.

moveItem(*item*, *after=None*)

Move an item to a new position in the collection.

Parameters

- **item** (*obj*) – *Audio*, *Video*, or *Photo* object to be moved in the collection.
- **after** (*obj*) – *Audio*, *Video*, or *Photo* object to move the item after in the collection.

Raises
plexapi.exceptions.BadRequest – When trying to move items in a smart collection.

updateFilters(*libtype=None*, *limit=None*, *sort=None*, *filters=None*, ***kwargs*)

Update the filters for a smart collection.

Parameters

- **libtype** (*str*) – The specific type of content to filter (movie, show, season, episode, artist, album, track, photoalbum, photo, collection).
- **limit** (*int*) – Limit the number of items in the collection.
- **sort** (*str or list, optional*) – A string of comma separated sort fields or a list of sort fields in the format `column:dir`. See *search()* for more info.
- **filters** (*dict*) – A dictionary of advanced filters. See *search()* for more info.
- ****kwargs** (*dict*) – Additional custom filters to apply to the search results. See *search()* for more info.

Raises
plexapi.exceptions.BadRequest – When trying update filters for a regular collection.

delete()

Delete the collection.

classmethod create(*server*, *title*, *section*, *items=None*, *smart=False*, *limit=None*, *libtype=None*, *sort=None*, *filters=None*, ***kwargs*)

Create a collection.

Parameters

- **server** (*PlexServer*) – Server to create the collection on.
- **title** (*str*) – Title of the collection.
- **section** (*LibrarySection*, str) – The library section to create the collection in.
- **items** (*List*) – Regular collections only, list of *Audio*, *Video*, or *Photo* objects to be added to the collection.
- **smart** (*bool*) – True to create a smart collection. Default False.
- **limit** (*int*) – Smart collections only, limit the number of items in the collection.
- **libtype** (*str*) – Smart collections only, the specific type of content to filter (movie, show, season, episode, artist, album, track, photoalbum, photo).
- **sort** (*str or list, optional*) – Smart collections only, a string of comma separated sort fields or a list of sort fields in the format `column:dir`. See *search()* for more info.

- **filters** (*dict*) – Smart collections only, a dictionary of advanced filters. See *search()* for more info.
- ****kwargs** (*dict*) – Smart collections only, additional custom filters to apply to the search results. See *search()* for more info.

Raises

- ***plexapi.exceptions.BadRequest*** – When no items are included to create the collection.
- ***plexapi.exceptions.BadRequest*** – When mixing media types in the collection.

Returns
A new instance of the created Collection.

Return type
Collection

sync(*videoQuality=None*, *photoResolution=None*, *audioBitrate=None*, *client=None*, *clientId=None*, *limit=None*, *unwatched=False*, *title=None*)

Add the collection as sync item for the specified device. See *sync()* for possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *sync* module. Used only when collection contains video.
- **photoResolution** (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module *sync*. Used only when collection contains photos.
- **audioBitrate** (*int*) – maximum bitrate for synchronized music, better use one of MUSIC_BITRATE_* values from the module *sync*. Used only when collection contains audio.
- **client** (*MyPlexDevice*) – sync destination, see *sync()*.
- **clientId** (*str*) – sync destination, see *sync()*.
- **limit** (*int*) – maximum count of items to sync, unlimited if *None*.
- **unwatched** (*bool*) – if *True* watched videos wouldn't be synced.
- **title** (*str*) – descriptive title for the new *SyncItem*, if empty the value would be generated from metadata of current photo.

Raises

- ***BadRequest*** – When collection is not allowed to sync.
- ***Unsupported*** – When collection content is unsupported.

Returns
A new instance of the created sync item.

Return type
SyncItem

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

CHAPTER

EIGHT

CONFIG PLEXAPI.CONFIG

class plexapi.config.**PlexConfig**(*path*)

Bases: ConfigParser

PlexAPI configuration object. Settings are stored in an INI file within the user's home directory and can be overridden after importing plexapi by simply setting the value. See the documentation section 'Configuration' for more details on available options.

Parameters

path (*str*) – Path of the configuration file to load.

get(*key*, *default=None*, *cast=None*)

Returns the specified configuration value or <default> if not found.

Parameters

- **key** (*str*) – Configuration variable to load in the format '<section>.<variable>'.
- **default** – Default value to use if key not found.
- **cast** (*func*) – Cast the value to the specified type before returning.

plexapi.config.**reset_base_headers**()

Convenience function returns a dict of all base X-Plex-* headers for session requests.

CHAPTER

NINE

EXCEPTIONS PLEXAPI.EXCEPTIONS

exception plexapi.exceptions.**PlexApiException**

Bases: Exception

Base class for all PlexAPI exceptions.

exception plexapi.exceptions.**BadRequest**

Bases: *PlexApiException*

An invalid request, generally a user error.

exception plexapi.exceptions.**NotFound**

Bases: *PlexApiException*

Request media item or device is not found.

exception plexapi.exceptions.**UnknownType**

Bases: *PlexApiException*

Unknown library type.

exception plexapi.exceptions.**Unsupported**

Bases: *PlexApiException*

Unsupported client request.

exception plexapi.exceptions.**Unauthorized**

Bases: *BadRequest*

Invalid username/password or token.

exception plexapi.exceptions.**TwoFactorRequired**

Bases: *Unauthorized*

Two factor authentication required.

CHAPTER

TEN

GDM PLEXAPI.GDM

Support for discovery using GDM (Good Day Mate), multicast protocol by Plex.

Licensed Apache 2.0 # From https://github.com/home-assistant/netdisco/netdisco/gdm.py

Inspired by:
hippojay's plexGDM: https://github.com/hippojay/script.plexbmc.helper/resources/lib/plexgdm.py iBaa's PlexConnect: https://github.com/iBaa/PlexConnect/PlexAPI.py

class plexapi.gdm.**GDM**

Bases: object

Base class to discover GDM services.

Variables
entries (*List<dict>*) – List of server and/or client data discovered.

scan(*scan_for_clients=False*)

Scan the network.

all(*scan_for_clients=False*)

Return all found entries.

Will scan for entries if not scanned recently.

find_by_content_type(*value*)

Return a list of entries that match the content_type.

find_by_data(*values*)

Return a list of entries that match the search parameters.

update(*scan_for_clients*)

Scan for new GDM services.

Examples of the dict list assigned to self.entries by this function:

Server:

[{'data': {
'Content-Type': 'plex/media-server', 'Host': '53f4b5b6023d41182fe88a99b0e714ba.plex.direct', 'Name': 'myfirstplexserver', 'Port': '32400', 'Resource-Identifier': '646ab0aa8a01c543e94ba975f6fd6efadc36b7', 'Updated-At': '1585769946', 'Version': '1.18.8.2527-740d4c206',

},
'from': ('10.10.10.100', 32414)}]

Clients:

[{'data': {'Content-Type': 'plex/media-player',
'Device-Class': 'stb', 'Name': 'plexamp', 'Port': '36000', 'Product': 'Plexamp', 'Protocol': 'plex', 'Protocol-Capabilities': 'timeline,playback,playqueues,playqueues-creation', 'Protocol-Version': '1', 'Resource-Identifier': 'b6e57a3f-e0f8-494f-8884-f4b58501467e', 'Version': '1.1.0',

},
'from': ('10.10.10.101', 32412)}]

plexapi.gdm.**main**()

Test GDM discovery.

CHAPTER

ELEVEN

LIBRARY PLEXAPI.LIBRARY

class plexapi.library.**Library**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a PlexServer library. This contains all sections of media defined in your Plex server including video, shows and audio.

Variables

- **key** (*str*) – '/library'
- **identifier** (*str*) – Unknown ('com.plexapp.plugins.library').
- **mediaTagVersion** (*str*) – Unknown (/system/bundle/media/flags/)
- **server** (*PlexServer*) – PlexServer this client is connected to.
- **title1** (*str*) – 'Plex Library' (not sure how useful this is).
- **title2** (*str*) – Second title (this is blank on my setup).

sections()

Returns a list of all media sections in this library. Library sections may be any of *MovieSection*, *ShowSection*, *MusicSection*, *PhotoSection*.

section(*title*)

Returns the *LibrarySection* that matches the specified title. Note: Multiple library sections with the same title is ambiguous. Use *sectionByID()* instead for an exact match.

Parameters
title (*str*) – Title of the section to return.

Raises
NotFound – The library section title is not found on the server.

sectionByID(*sectionID*)

Returns the *LibrarySection* that matches the specified sectionID.

Parameters
sectionID (*int*) – ID of the section to return.

Raises
NotFound – The library section ID is not found on the server.

hubs(*sectionID=None*, *identifier=None*, ***kwargs*)

Returns a list of *Hub* across all library sections.

Parameters

- **sectionID** (*int or str or list, optional*) – IDs of the sections to limit results or "playlists".
- **identifier** (*str or list, optional*) – Names of identifiers to limit results. Available on *Hub* instances as the *hubIdentifier* attribute. Examples: 'home.continue' or 'home.ondeck'

all(***kwargs*)

Returns a list of all media from all library sections. This may be a very large dataset to retrieve.

onDeck()

Returns a list of all media items on deck.

recentlyAdded()

Returns a list of all media items recently added.

search(*title=None*, *libtype=None*, ***kwargs*)

Searching within a library section is much more powerful. It seems certain attributes on the media objects can be targeted to filter this search down a bit, but I haven't found the documentation for it.

Example: "studio=Comedy%20Central" or "year=1999" "title=Kung Fu" all work. Other items such as actor=<id> seem to work, but require you already know the id of the actor. TLDR: This is untested but seems to work. Use library section search when you can.

cleanBundles()

Poster images and other metadata for items in your library are kept in "bundle" packages. When you remove items from your library, these bundles aren't immediately removed. Removing these old bundles can reduce the size of your install. By default, your server will automatically clean up old bundles once a week as part of Scheduled Tasks.

emptyTrash()

If a library has items in the Library Trash, use this option to empty the Trash.

optimize()

The Optimize option cleans up the server database from unused or fragmented data. For example, if you have deleted or added an entire library or many items in a library, you may like to optimize the database.

update()

Scan this library for new items.

cancelUpdate()

Cancel a library update.

refresh()

Forces a download of fresh media information from the internet. This can take a long time. Any locked fields are not modified.

deleteMediaPreviews()

Delete the preview thumbnails for the all sections. This cannot be undone. Recreating media preview files can take hours or even days.

add(*name=''*, *type=''*, *agent=''*, *scanner=''*, *location=''*, *language='en-US'*, **args*, ***kwargs*)

Simplified add for the most common options.

Parameters

- **name** (*str*) – Name of the library
- **agent** (*str*) – Example com.plexapp.agents.imdb

- **type** (*str*) – movie, show, # check me
- **location** (*str or list*) – /path/to/files, ["/path/to/files", "/path/to/morefiles"]
- **language** (*str*) – Four letter language code (e.g. en-US)
- **kwargs** (*dict*) – Advanced options should be passed as a dict. where the id is the key.

Photo Preferences

- **agent** (str): com.plexapp.agents.none
- **enableAutoPhotoTags** (bool): Tag photos. Default value false.
- **enableBIFGeneration** (bool): Enable video preview thumbnails. Default value true.
- **includeInGlobal** (bool): Include in dashboard. Default value true.
- **scanner** (str): Plex Photo Scanner

Movie Preferences

- **agent** (str): com.plexapp.agents.none, com.plexapp.agents.imdb, tv.plex.agents.movie, com.plexapp.agents.themoviedb
- **enableBIFGeneration** (bool): Enable video preview thumbnails. Default value true.
- **enableCinemaTrailers** (bool): Enable Cinema Trailers. Default value true.
- **includeInGlobal** (bool): Include in dashboard. Default value true.
- **scanner** (str): Plex Movie, Plex Movie Scanner, Plex Video Files Scanner, Plex Video Files

IMDB Movie Options (com.plexapp.agents.imdb)

- **title** (bool): Localized titles. Default value false.
- **extras** (bool): Find trailers and extras automatically (Plex Pass required). Default value true.
- **only_trailers** (bool): Skip extras which aren't trailers. Default value false.
- **redband** (bool): Use red band (restricted audiences) trailers when available. Default value false.
- **native_subs** (bool): Include extras with subtitles in Library language. Default value false.
- **cast_list** (int): Cast List Source: Default value 1 Possible options: 0:IMDb,1:The Movie Database.
- **ratings** (int): Ratings Source, Default value 0 Possible options: 0:Rotten Tomatoes, 1:IMDb, 2:The Movie Database.
- **summary** (int): Plot Summary Source: Default value 1 Possible options: 0:IMDb,1:The Movie Database.
- **country** (int): Default value 46 Possible options 0:Argentina, 1:Australia, 2:Austria, 3:Belgium, 4:Belize, 5:Bolivia, 6:Brazil, 7:Canada, 8:Chile, 9:Colombia, 10:Costa Rica, 11:Czech Republic, 12:Denmark, 13:Dominican Republic, 14:Ecuador, 15:El Salvador, 16:France, 17:Germany, 18:Guatemala, 19:Honduras, 20:Hong Kong SAR, 21:Ireland, 22:Italy, 23:Jamaica, 24:Korea, 25:Liechtenstein, 26:Luxembourg, 27:Mexico, 28:Netherlands, 29:New Zealand, 30:Nicaragua, 31:Panama, 32:Paraguay, 33:Peru, 34:Portugal, 35:Peoples Republic of China, 36:Puerto Rico, 37:Russia, 38:Singapore, 39:South Africa, 40:Spain, 41:Sweden, 42:Switzerland, 43:Taiwan, 44:Trinidad, 45:United Kingdom, 46:United States, 47:Uruguay, 48:Venezuela.
- **collections** (bool): Use collection info from The Movie Database. Default value false.
- **localart** (bool): Prefer artwork based on library language. Default value true.
- **adult** (bool): Include adult content. Default value false.

- **usage** (bool): Send anonymous usage data to Plex. Default value true.

TheMovieDB Movie Options (com.plexapp.agents.themoviedb)

- **collections** (bool): Use collection info from The Movie Database. Default value false.
- **localart** (bool): Prefer artwork based on library language. Default value true.
- **adult** (bool): Include adult content. Default value false.
- **country** (int): Country (used for release date and content rating). Default value 47 Possible options 0:, 1:Argentina, 2:Australia, 3:Austria, 4:Belgium, 5:Belize, 6:Bolivia, 7:Brazil, 8:Canada, 9:Chile, 10:Colombia, 11:Costa Rica, 12:Czech Republic, 13:Denmark, 14:Dominican Republic, 15:Ecuador, 16:El Salvador, 17:France, 18:Germany, 19:Guatemala, 20:Honduras, 21:Hong Kong SAR, 22:Ireland, 23:Italy, 24:Jamaica, 25:Korea, 26:Liechtenstein, 27:Luxembourg, 28:Mexico, 29:Netherlands, 30:New Zealand, 31:Nicaragua, 32:Panama, 33:Paraguay, 34:Peru, 35:Portugal, 36:Peoples Republic of China, 37:Puerto Rico, 38:Russia, 39:Singapore, 40:South Africa, 41:Spain, 42:Sweden, 43:Switzerland, 44:Taiwan, 45:Trinidad, 46:United Kingdom, 47:United States, 48:Uruguay, 49:Venezuela.

Show Preferences

- **agent** (str): com.plexapp.agents.none, com.plexapp.agents.thetvdb, com.plexapp.agents.themoviedb, tv.plex.agents.series
- **enableBIFGeneration** (bool): Enable video preview thumbnails. Default value true.
- **episodeSort** (int): Episode order. Default -1 Possible options: 0:Oldest first, 1:Newest first.
- **flattenSeasons** (int): Seasons. Default value 0 Possible options: 0:Show,1:Hide.
- **includeInGlobal** (bool): Include in dashboard. Default value true.
- **scanner** (str): Plex TV Series, Plex Series Scanner

TheTVDB Show Options (com.plexapp.agents.thetvdb)

- **extras** (bool): Find trailers and extras automatically (Plex Pass required). Default value true.
- **native_subs** (bool): Include extras with subtitles in Library language. Default value false.

TheMovieDB Show Options (com.plexapp.agents.themoviedb)

- **collections** (bool): Use collection info from The Movie Database. Default value false.
- **localart** (bool): Prefer artwork based on library language. Default value true.
- **adult** (bool): Include adult content. Default value false.
- **country** (int): Country (used for release date and content rating). Default value 47 options 0:, 1:Argentina, 2:Australia, 3:Austria, 4:Belgium, 5:Belize, 6:Bolivia, 7:Brazil, 8:Canada, 9:Chile, 10:Colombia, 11:Costa Rica, 12:Czech Republic, 13:Denmark, 14:Dominican Republic, 15:Ecuador, 16:El Salvador, 17:France, 18:Germany, 19:Guatemala, 20:Honduras, 21:Hong Kong SAR, 22:Ireland, 23:Italy, 24:Jamaica, 25:Korea, 26:Liechtenstein, 27:Luxembourg, 28:Mexico, 29:Netherlands, 30:New Zealand, 31:Nicaragua, 32:Panama, 33:Paraguay, 34:Peru, 35:Portugal, 36:Peoples Republic of China, 37:Puerto Rico, 38:Russia, 39:Singapore, 40:South Africa, 41:Spain, 42:Sweden, 43:Switzerland, 44:Taiwan, 45:Trinidad, 46:United Kingdom, 47:United States, 48:Uruguay, 49:Venezuela.

Other Video Preferences

- **agent** (str): com.plexapp.agents.none, com.plexapp.agents.imdb, com.plexapp.agents.themoviedb
- **enableBIFGeneration** (bool): Enable video preview thumbnails. Default value true.
- **enableCinemaTrailers** (bool): Enable Cinema Trailers. Default value true.

- **includeInGlobal** (bool): Include in dashboard. Default value true.
- **scanner** (str): Plex Movie Scanner, Plex Video Files Scanner

IMDB Other Video Options (com.plexapp.agents.imdb)

- **title** (bool): Localized titles. Default value false.
- **extras** (bool): Find trailers and extras automatically (Plex Pass required). Default value true.
- **only_trailers** (bool): Skip extras which aren't trailers. Default value false.
- **redband** (bool): Use red band (restricted audiences) trailers when available. Default value false.
- **native_subs** (bool): Include extras with subtitles in Library language. Default value false.
- **cast_list** (int): Cast List Source: Default value 1 Possible options: 0:IMDb,1:The Movie Database.
- **ratings** (int): Ratings Source Default value 0 Possible options: 0:Rotten Tomatoes,1:IMDb,2:The Movie Database.
- **summary** (int): Plot Summary Source: Default value 1 Possible options: 0:IMDb,1:The Movie Database.
- **country** (int): Country: Default value 46 Possible options: 0:Argentina, 1:Australia, 2:Austria, 3:Belgium, 4:Belize, 5:Bolivia, 6:Brazil, 7:Canada, 8:Chile, 9:Colombia, 10:Costa Rica, 11:Czech Republic, 12:Denmark, 13:Dominican Republic, 14:Ecuador, 15:El Salvador, 16:France, 17:Germany, 18:Guatemala, 19:Honduras, 20:Hong Kong SAR, 21:Ireland, 22:Italy, 23:Jamaica, 24:Korea, 25:Liechtenstein, 26:Luxembourg, 27:Mexico, 28:Netherlands, 29:New Zealand, 30:Nicaragua, 31:Panama, 32:Paraguay, 33:Peru, 34:Portugal, 35:Peoples Republic of China, 36:Puerto Rico, 37:Russia, 38:Singapore, 39:South Africa, 40:Spain, 41:Sweden, 42:Switzerland, 43:Taiwan, 44:Trinidad, 45:United Kingdom, 46:United States, 47:Uruguay, 48:Venezuela.
- **collections** (bool): Use collection info from The Movie Database. Default value false.
- **localart** (bool): Prefer artwork based on library language. Default value true.
- **adult** (bool): Include adult content. Default value false.
- **usage** (bool): Send anonymous usage data to Plex. Default value true.

TheMovieDB Other Video Options (com.plexapp.agents.themoviedb)

- **collections** (bool): Use collection info from The Movie Database. Default value false.
- **localart** (bool): Prefer artwork based on library language. Default value true.
- **adult** (bool): Include adult content. Default value false.
- **country** (int): Country (used for release date and content rating). Default value 47 Possible options 0:, 1:Argentina, 2:Australia, 3:Austria, 4:Belgium, 5:Belize, 6:Bolivia, 7:Brazil, 8:Canada, 9:Chile, 10:Colombia, 11:Costa Rica, 12:Czech Republic, 13:Denmark, 14:Dominican Republic, 15:Ecuador, 16:El Salvador, 17:France, 18:Germany, 19:Guatemala, 20:Honduras, 21:Hong Kong SAR, 22:Ireland, 23:Italy, 24:Jamaica, 25:Korea, 26:Liechtenstein, 27:Luxembourg, 28:Mexico, 29:Netherlands, 30:New Zealand, 31:Nicaragua, 32:Panama, 33:Paraguay, 34:Peru, 35:Portugal, 36:Peoples Republic of China, 37:Puerto Rico, 38:Russia, 39:Singapore, 40:South Africa, 41:Spain, 42:Sweden, 43:Switzerland, 44:Taiwan, 45:Trinidad, 46:United Kingdom, 47:United States, 48:Uruguay, 49:Venezuela.

`history`(*maxresults=None*, *mindate=None*)

Get Play History for all library Sections for the owner. :param maxresults: Only return the specified number of results (optional). :type maxresults: int :param mindate: Min datetime to return results from. :type mindate: datetime

tags(*tag*)

Returns a list of *LibraryMediaTag* objects for the specified tag.

Parameters
tag (*str*) – Tag name (see TAGTYPES).

class plexapi.library.**LibrarySection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Base class for a single library section.

Variables

- **agent** (*str*) – The metadata agent used for the library section (com.plexapp.agents.imdb, etc).
- **allowSync** (*bool*) – True if you allow syncing content from the library section.
- **art** (*str*) – Background artwork used to respresent the library section.
- **composite** (*str*) – Composite image used to represent the library section.
- **createdAt** (*datetime*) – Datetime the library section was created.
- **filters** (*bool*) – True if filters are available for the library section.
- **key** (*int*) – Key (or ID) of this library section.
- **language** (*str*) – Language represented in this section (en, xn, etc).
- **locations** (*List<str>*) – List of folder paths added to the library section.
- **refreshing** (*bool*) – True if this section is currently being refreshed.
- **scanner** (*str*) – Internal scanner used to find media (Plex Movie Scanner, Plex Premium Music Scanner, etc.)
- **thumb** (*str*) – Thumbnail image used to represent the library section.
- **title** (*str*) – Name of the library section.
- **type** (*str*) – Type of content section represents (movie, show, artist, photo).
- **updatedAt** (*datetime*) – Datetime the library section was last updated.
- **uuid** (*str*) – Unique id for the section (32258d7c-3e6c-4ac5-98ad-bad7a3b78c63)

property totalSize

Returns the total number of items in the library for the default library type.

property totalDuration

Returns the total duration (in milliseconds) of items in the library.

property totalStorage

Returns the total storage (in bytes) of items in the library.

totalViewSize(*libtype=None*, *includeCollections=True*)

Returns the total number of items in the library for a specified libtype. The number of items for the default library type will be returned if no libtype is specified. (e.g. Specify `libtype='episode'` for the total number of episodes or `libtype='albums'` for the total number of albums.)

Parameters

- **libtype** (*str, optional*) – The type of items to return the total number for (movie, show, season, episode, artist, album, track, photoalbum). Default is the main library type.

- **includeCollections** (*bool, optional*) – True or False to include collections in the total number. Default is True.

delete()

Delete a library section.

edit(*agent=None*, ***kwargs*)

Edit a library. See *Library* for example usage.

Parameters

- **agent** (*str, optional*) – The library agent.
- **kwargs** (*dict*) – Dict of settings to edit.

addLocations(*location*)

Add a location to a library.

Parameters

location (*str or list*) – A single folder path, list of paths.

Example

```
LibrarySection.addLocations('/path/1')
LibrarySection.addLocations(['/path/1', 'path/2', '/path/3'])
```

removeLocations(*location*)

Remove a location from a library.

Parameters

location (*str or list*) – A single folder path, list of paths.

Example

```
LibrarySection.removeLocations('/path/1')
LibrarySection.removeLocations(['/path/1', 'path/2', '/path/3'])
```

get(*title*, ***kwargs*)

Returns the media item with the specified title and kwargs.

Parameters

- **title** (*str*) – Title of the item to return.
- **kwargs** (*dict*) – Additional search parameters. See *search()* for more info.

Raises

NotFound – The title is not found in the library.

getGuid(*guid*)

Returns the media item with the specified external Plex, IMDB, TMDB, or TVDB ID. Note: Only available for the Plex Movie and Plex TV Series agents.

Parameters

guid (*str*) – The external guid of the item to return. Examples: Plex `plex://show/5d9c086c46115600200aa2fe` IMDB `imdb://tt0944947`, TMDB `tmdb://1399`, TVDB `tvdb://121361`.

Raises

NotFound – The guid is not found in the library.

Example

```
result1 = library.getGuid('plex://show/5d9c086c46115600200aa2fe')
result2 = library.getGuid('imdb://tt0944947')
result3 = library.getGuid('tmdb://1399')
result4 = library.getGuid('tvdb://121361')

# Alternatively, create your own guid lookup dictionary for faster performance
guidLookup = {}
for item in library.all():
    guidLookup[item.guid] = item
    guidLookup.update({guid.id: item for guid in item.guids})

result1 = guidLookup['plex://show/5d9c086c46115600200aa2fe']
result2 = guidLookup['imdb://tt0944947']
result3 = guidLookup['tmdb://1399']
result4 = guidLookup['tvdb://121361']
```

all(*libtype=None*, ***kwargs*)

Returns a list of all items from this library section. See description of *search()* for details about filtering / sorting.

folders()

Returns a list of available *Folder* for this library section.

managedHubs()

Returns a list of available *ManagedHub* for this library section.

resetManagedHubs()

Reset the managed hub customizations for this library section.

hubs()

Returns a list of available *Hub* for this library section.

agents()

Returns a list of available *Agent* for this library section.

settings()

Returns a list of all library settings.

editAdvanced(***kwargs*)

Edit a library's advanced settings.

defaultAdvanced()

Edit all of library's advanced settings to default.

lockAllField(*field*, *libtype=None*)

Lock a field for all items in the library.

Parameters

- **field** (*str*) – The field to lock (e.g. thumb, rating, collection).
- **libtype** (*str, optional*) – The library type to lock (movie, show, season, episode, artist, album, track, photoalbum, photo). Default is the main library type.

unlockAllField(*field*, *libtype=None*)

Unlock a field for all items in the library.

Parameters

- **field** (*str*) – The field to unlock (e.g. thumb, rating, collection).
- **libtype** (*str, optional*) – The library type to lock (movie, show, season, episode, artist, album, track, photoalbum, photo). Default is the main library type.

timeline()

Returns a timeline query for this library section.

onDeck()

Returns a list of media items on deck from this library section.

continueWatching()

Return a list of media items in the library's Continue Watching hub.

recentlyAdded(*maxresults=50*, *libtype=None*)

Returns a list of media items recently added from this library section.

Parameters

- **maxresults** (*int*) – Max number of items to return (default 50).
- **libtype** (*str, optional*) – The library type to filter (movie, show, season, episode, artist, album, track, photoalbum, photo). Default is the main library type.

analyze()

Run an analysis on all of the items in this library section. See See *analyze()* for more details.

emptyTrash()

If a section has items in the Trash, use this option to empty the Trash.

update(*path=None*)

Scan this section for new media.

Parameters

path (*str, optional*) – Full path to folder to scan.

cancelUpdate()

Cancel update of this Library Section.

refresh()

Forces a download of fresh media information from the internet. This can take a long time. Any locked fields are not modified.

deleteMediaPreviews()

Delete the preview thumbnails for items in this library. This cannot be undone. Recreating media preview files can take hours or even days.

filterTypes()

Returns a list of available *FilteringType* for this library section.

getFilterType(*libtype=None*)

Returns a *FilteringType* for a specified libtype.

Parameters

libtype (*str, optional*) – The library type to filter (movie, show, season, episode, artist, album, track, photoalbum, photo, collection).

Raises
NotFound – Unknown libtype for this library.

fieldTypes()

Returns a list of available *FilteringFieldType* for this library section.

getFieldType(*fieldType*)

Returns a *FilteringFieldType* for a specified fieldType.

Parameters
fieldType (*str*) – The data type for the field (tag, integer, string, boolean, date, subtitleLanguage, audioLanguage, resolution).

Raises
NotFound – Unknown fieldType for this library.

listFilters(*libtype=None*)

Returns a list of available *FilteringFilter* for a specified libtype. This is the list of options in the filter dropdown menu (screenshot).

Parameters
libtype (*str, optional*) – The library type to filter (movie, show, season, episode, artist, album, track, photoalbum, photo, collection).

Example

```
availableFilters = [f.filter for f in library.listFilters()]
print("Available filter fields:", availableFilters)
```

listSorts(*libtype=None*)

Returns a list of available *FilteringSort* for a specified libtype. This is the list of options in the sorting dropdown menu (screenshot).

Parameters
libtype (*str, optional*) – The library type to filter (movie, show, season, episode, artist, album, track, photoalbum, photo, collection).

Example

```
availableSorts = [f.key for f in library.listSorts()]
print("Available sort fields:", availableSorts)
```

listFields(*libtype=None*)

Returns a list of available `FilteringFields` for a specified libtype. This is the list of options in the custom filter dropdown menu (screenshot).

Parameters
libtype (*str, optional*) – The library type to filter (movie, show, season, episode, artist, album, track, photoalbum, photo, collection).

Example

```
availableFields = [f.key.split('.')[-1] for f in library.listFields()]
print("Available fields:", availableFields)
```

listOperators(*fieldType*)

Returns a list of available *FilteringOperator* for a specified fieldType. This is the list of options in the custom filter operator dropdown menu (screenshot).

Parameters

fieldType (*str*) – The data type for the field (tag, integer, string, boolean, date, subtitleLanguage, audioLanguage, resolution).

Example

```
field = 'genre'  # Available filter field from listFields()
filterField = next(f for f in library.listFields() if f.key.endswith(field))
availableOperators = [o.key for o in library.listOperators(filterField.type)]
print(f"Available operators for {field}:", availableOperators)
```

listFilterChoices(*field*, *libtype=None*)

Returns a list of available *FilterChoice* for a specified *FilteringFilter* or filter field. This is the list of available values for a custom filter (screenshot).

Parameters

- **field** (*str*) – *FilteringFilter* object, or the name of the field (genre, year, contentRating, etc.).
- **libtype** (*str, optional*) – The library type to filter (movie, show, season, episode, artist, album, track, photoalbum, photo, collection).

Raises

- ***BadRequest*** – Invalid filter field.
- ***NotFound*** – Unknown filter field.

Example

```
field = 'genre'  # Available filter field from listFilters()
availableChoices = [f.title for f in library.listFilterChoices(field)]
print(f"Available choices for {field}:", availableChoices)
```

hubSearch(*query*, *mediatype=None*, *limit=None*)

Returns the hub search results for this library. See *plexapi.server.PlexServer.search()* for details and parameters.

search(*title=None*, *sort=None*, *maxresults=None*, *libtype=None*, *container_start=None*, *container_size=None*, *limit=None*, *filters=None*, ***kwargs*)

Search the library. The http requests will be batched in container_size. If you are only looking for the first <num> results, it would be wise to set the maxresults option to that amount so the search doesn't iterate over all results on the server.

Parameters

- **title** (*str, optional*) – General string query to search for. Partial string matches are allowed.
- **sort** (*FilteringSort* or str or list, optional) – A field to sort the results. See the details below for more info.
- **maxresults** (*int, optional*) – Only return the specified number of results.

- **libtype** (*str, optional*) – Return results of a specific type (movie, show, season, episode, artist, album, track, photoalbum, photo, collection) (e.g. `libtype='episode'` will only return *Episode* objects)
- **container_start** (*int, optional*) – Default 0.
- **container_size** (*int, optional*) – Default X_PLEX_CONTAINER_SIZE in your config file.
- **limit** (*int, optional*) – Limit the number of results from the filter.
- **filters** (*dict, optional*) – A dictionary of advanced filters. See the details below for more info.
- ****kwargs** (*dict*) – Additional custom filters to apply to the search results. See the details below for more info.

Raises

- *BadRequest* – When the sort or filter is invalid.
- *NotFound* – When applying an unknown sort or filter.

Sorting Results

The search results can be sorted by including the `sort` parameter.

- See *listSorts()* to get a list of available sort fields.

The `sort` parameter can be a *FilteringSort* object or a sort string in the format `field:dir`. The sort direction `dir` can be `asc`, `desc`, or `nullsLast`. Omitting the sort direction or using a *FilteringSort* object will sort the results in the default direction of the field. Multi-sorting on multiple fields can be achieved by using a comma separated list of sort strings, or a list of *FilteringSort* object or strings.

Examples

```
library.search(sort="titleSort:desc")  # Sort title in descending order
library.search(sort="titleSort")  # Sort title in the default order
# Multi-sort by year in descending order, then by audience rating in descending␣
↪order
library.search(sort="year:desc,audienceRating:desc")
library.search(sort=["year:desc", "audienceRating:desc"])
```

Using Plex Filters

Any of the available custom filters can be applied to the search results (screenshot).

- See *listFields()* to get a list of all available fields.
- See *listOperators()* to get a list of all available operators.
- See *listFilterChoices()* to get a list of all available filter values.

The following filter fields are just some examples of the possible filters. The list is not exhaustive, and not all filters apply to all library types.

- **actor** (*MediaTag*): Search for the name of an actor.
- **addedAt** (*datetime*): Search for items added before or after a date. See operators below.
- **audioLanguage** (*str*): Search for a specific audio language (3 character code, e.g. jpn).
- **collection** (*MediaTag*): Search for the name of a collection.
- **contentRating** (*MediaTag*): Search for a specific content rating.

- **country** (*MediaTag*): Search for the name of a country.
- **decade** (*int*): Search for a specific decade (e.g. 2000).
- **director** (*MediaTag*): Search for the name of a director.
- **duplicate** (*bool*) Search for duplicate items.
- **genre** (*MediaTag*): Search for a specific genre.
- **hdr** (*bool*): Search for HDR items.
- **inProgress** (*bool*): Search for in progress items.
- **label** (*MediaTag*): Search for a specific label.
- **lastViewedAt** (*datetime*): Search for items watched before or after a date. See operators below.
- **mood** (*MediaTag*): Search for a specific mood.
- **producer** (*MediaTag*): Search for the name of a producer.
- **resolution** (*str*): Search for a specific resolution (e.g. 1080).
- **studio** (*str*): Search for the name of a studio.
- **style** (*MediaTag*): Search for a specific style.
- **subtitleLanguage** (*str*): Search for a specific subtitle language (3 character code, e.g. eng)
- **unmatched** (*bool*): Search for unmatched items.
- **unwatched** (*bool*): Search for unwatched items.
- **userRating** (*int*): Search for items with a specific user rating.
- **writer** (*MediaTag*): Search for the name of a writer.
- **year** (*int*): Search for a specific year.

Tag type filter values can be a *FilterChoice* object, *MediaTag* object, the exact name `MediaTag.tag` (*str*), or the exact id `MediaTag.id` (*int*).

Date type filter values can be a `datetime` object, a relative date using a one of the available date suffixes (e.g. `30d`) (*str*), or a date in `YYYY-MM-DD` (*str*) format.

Relative date suffixes:

- `s`: `seconds`
- `m`: `minutes`
- `h`: `hours`
- `d`: `days`
- `w`: `weeks`
- `mon`: `months`
- `y`: `years`

Multiple values can be `OR` together by providing a list of values.

Examples

```
library.search(unwatched=True, year=2020, resolution="4k")
library.search(actor="Arnold Schwarzenegger", decade=1990)
library.search(contentRating="TV-G", genre="animation")
library.search(genre=["animation", "comedy"])  # Genre is animation OR comedy
library.search(studio=["Disney", "Pixar"])  # Studio contains Disney OR Pixar
```

Using a `libtype` Prefix

Some filters may be prefixed by the `libtype` separated by a `.` (e.g. `show.collection`, `episode.title`, `artist.style`, `album.genre`, `track.userRating`, etc.). This should not be confused with the `libtype` parameter. If no `libtype` prefix is provided, then the default library type is assumed. For example, in a TV show library `viewCount` is assumed to be `show.viewCount`. If you want to filter using episode view count then you must specify `episode.viewCount` explicitly. In addition, if the filter does not exist for the default library type it will fallback to the most specific `libtype` available. For example, `show.unwatched` does not exists so it will fallback to `episode.unwatched`. The `libtype` prefix cannot be included directly in the function parameters so the filters must be provided as a filters dictionary.

Examples

```
library.search(filters={"show.collection": "Documentary", "episode.inProgress":␣
↪True})
library.search(filters={"artist.genre": "pop", "album.decade": 2000})

# The following three options are identical and will return Episode objects
showLibrary.search(title="Winter is Coming", libtype='episode')
showLibrary.search(libtype='episode', filters={"episode.title": "Winter is␣
↪Coming"})
showLibrary.searchEpisodes(title="Winter is Coming")

# The following will search for the episode title but return Show objects
showLibrary.search(filters={"episode.title": "Winter is Coming"})

# The following will fallback to episode.unwatched
showLibrary.search(unwatched=True)
```

Using Plex Operators

Operators can be appended to the filter field to narrow down results with more granularity. The following is a list of possible operators depending on the data type of the filter being applied. A special & operator can also be used to `AND` together a list of values.

Type: *`MediaTag`* or *subtitleLanguage* or *audioLanguage*

- no operator: `is`
- `!`: `is not`

Type: *int*

- no operator: `is`
- `!`: `is not`
- `>>`: `is greater than`
- `<<`: `is less than`

Type: *str*

- no operator: `contains`
- `!`: `does not contain`
- `=`: `is`
- `!=`: `is not`
- `<`: `begins with`
- `>`: `ends with`

Type: *bool*

- no operator: `is true`
- `!`: `is false`

Type: *datetime*

- `<<`: `is before`
- `>>`: `is after`

Type: *resolution* or *guid*

- no operator: `is`

Operators cannot be included directly in the function parameters so the filters must be provided as a filters dictionary.

Examples

```
# Genre is horror AND thriller
library.search(filters={"genre&": ["horror", "thriller"]})

# Director is not Steven Spielberg
library.search(filters={"director!": "Steven Spielberg"})

# Title starts with Marvel and added before 2021-01-01
library.search(filters={"title<": "Marvel", "addedAt<<": "2021-01-01"})

# Added in the last 30 days using relative dates
library.search(filters={"addedAt>>": "30d"})

# Collection is James Bond and user rating is greater than 8
library.search(filters={"collection": "James Bond", "userRating>>": 8})
```

Using Advanced Filters

Any of the Plex filters described above can be combined into a single `filters` dictionary that mimics the advanced filters used in Plex Web with a tree of `and`/`or` branches. Each level of the tree must start with `and` (Match all of the following) or `or` (Match any of the following) as the dictionary key, and a list of dictionaries with the desired filters as the dictionary value.

The following example matches this advanced filter in Plex Web.

Examples

```
advancedFilters = {
    'and': [                                # Match all of the following in this␣
↪list
        {
            'or': [                         # Match any of the following in this␣
↪list
                {'title': 'elephant'},
                {'title': 'bunny'}
            ]
        },
        {'year>>': 1990},
        {'unwatched': True}
    ]
}
library.search(filters=advancedFilters)
```

Using PlexAPI Operators

For even more advanced filtering which cannot be achieved in Plex, the PlexAPI operators can be applied to any XML attribute. See *plexapi.base.PlexObject.fetchItems()* for a list of operators and how they are used. Note that using the Plex filters above will be faster since the filters are applied by the Plex server before the results are returned to PlexAPI. Using the PlexAPI operators requires the Plex server to return *all* results to allow PlexAPI to do the filtering. The Plex filters and the PlexAPI operators can be used in conjunction with each other.

Examples

```
library.search(summary__icontains="Christmas")
library.search(duration__gt=7200000)
library.search(audienceRating__lte=6.0, audienceRatingImage__startswith=
↪"rottentomatoes://")
library.search(media__videoCodec__exact="h265")
library.search(genre="holiday", viewCount__gte=3)
```

sync(*policy*, *mediaSettings*, *client=None*, *clientId=None*, *title=None*, *sort=None*, *libtype=None*, ***kwargs*)

Add current library section as sync item for specified device. See description of *search()* for details about filtering / sorting and *sync()* for possible exceptions.

Parameters

- **policy** (*Policy*) – policy of syncing the media (how many items to sync and process watched media or not), generated automatically when method called on specific LibrarySection object.
- **mediaSettings** (*MediaSettings*) – Transcoding settings used for the media, generated automatically when method called on specific LibrarySection object.
- **client** (*MyPlexDevice*) – sync destination, see *sync()*.
- **clientId** (*str*) – sync destination, see *sync()*.
- **title** (*str*) – descriptive title for the new *SyncItem*, if empty the value would be generated from metadata of current media.
- **sort** (*str*) – formatted as *column:dir*; column can be any of {*addedAt*, *originallyAvailableAt*, *lastViewedAt*, *titleSort*, *rating*, *mediaHeight*, *duration*}. dir can be *asc* or *desc*.

- **libtype** (*str*) – Filter results to a specific libtype (*movie*, *show*, *episode*, *artist*, *album*, *track*).

Returns
an instance of created syncItem.

Return type
SyncItem

Raises

- ***BadRequest*** – When the library is not allowed to sync.
- ***BadRequest*** – When the sort or filter is invalid.
- ***NotFound*** – When applying an unknown sort or filter.

Example

```
from plexapi import myplex
from plexapi.sync import Policy, MediaSettings, VIDEO_QUALITY_3_MBPS_720p

c = myplex.MyPlexAccount()
target = c.device('Plex Client')
sync_items_wd = c.syncItems(target.clientIdentifier)
srv = c.resource('Server Name').connect()
section = srv.library.section('Movies')
policy = Policy('count', unwatched=True, value=1)
media_settings = MediaSettings.create(VIDEO_QUALITY_3_MBPS_720p)
section.sync(target, policy, media_settings, title='Next best movie', sort=
↪'rating:desc')
```

history(*maxresults=None*, *mindate=None*)

Get Play History for this library Section for the owner. :param maxresults: Only return the specified number of results (optional). :type maxresults: int :param mindate: Min datetime to return results from. :type mindate: datetime

createCollection(*title*, *items=None*, *smart=False*, *limit=None*, *libtype=None*, *sort=None*, *filters=None*, ***kwargs*)

Alias for *createCollection()* using this *LibrarySection*.

collection(*title*)

Returns the collection with the specified title.

Parameters
title (*str*) – Title of the item to return.

Raises
NotFound – Unable to find collection.

collections(***kwargs*)

Returns a list of collections from this library section. See description of *search()* for details about filtering / sorting.

createPlaylist(*title*, *items=None*, *smart=False*, *limit=None*, *sort=None*, *filters=None*, *m3ufilepath=None*, ***kwargs*)

Alias for *createPlaylist()* using this *LibrarySection*.

playlist(*title*)

Returns the playlist with the specified title.

Parameters
title (*str*) – Title of the item to return.

Raises
NotFound – Unable to find playlist.

playlists(*sort=None*, ***kwargs*)

Returns a list of playlists from this library section.

getWebURL(*base=None*, *tab=None*, *key=None*)

Returns the Plex Web URL for the library.

Parameters

- **base** (*str*) – The base URL before the fragment (`#!`). Default is https://app.plex.tv/desktop.
- **tab** (*str*) – The library tab (recommended, library, collections, playlists, timeline).
- **key** (*str*) – A hub key.

common(*items*)

Returns a *Common* object for the specified items.

multiEdit(*items*, ***kwargs*)

Edit multiple objects at once. Note: This is a low level method and you need to know all the field/tag keys. See `batchMultiEdits` instead.

Parameters

- **items** (*List*) – List of *Audio*, *Video*, *Photo*, or *Collection* objects to be edited.
- **kwargs** (*dict*) – Dict of settings to edit.

batchMultiEdits(*items*)

Enable batch multi-editing mode to save API calls. Must call *saveMultiEdits()* at the end to save all the edits. See *EditFieldMixin* and *EditTagsMixin* for individual field and tag editing methods.

Parameters
items (*List*) – List of *Audio*, *Video*, *Photo*, or *Collection* objects to be edited.

Example

```
movies = MovieSection.all()
items = [movies[0], movies[3], movies[5]]

# Batch multi-editing multiple fields and tags in a single API call
MovieSection.batchMultiEdits(items)
MovieSection.editTitle('A New Title').editSummary('A new summary').editTagline(
↪'A new tagline') \
    .addCollection('New Collection').removeGenre('Action').addLabel('Favorite')
MovieSection.saveMultiEdits()
```

saveMultiEdits()

Save all the batch multi-edits. See *batchMultiEdits()* for details.

class plexapi.library.**MovieSection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibrarySection*, *MovieEditMixins*

Represents a *LibrarySection* section containing movies.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'movie'

searchMovies(***kwargs*)

Search for a movie. See *search()* for usage.

recentlyAddedMovies(*maxresults=50*)

Returns a list of recently added movies from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

sync(*videoQuality*, *limit=None*, *unwatched=False*, ***kwargs*)

Add current Movie library section as sync item for specified device. See description of *search()* for details about filtering / sorting and *sync()* for details on syncing libraries and possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *sync* module.
- **limit** (*int*) – maximum count of movies to sync, unlimited if *None*.
- **unwatched** (*bool*) – if *True* watched videos wouldn't be synced.

Returns
an instance of created syncItem.

Return type
SyncItem

Example

```
from plexapi import myplex
from plexapi.sync import VIDEO_QUALITY_3_MBPS_720p

c = myplex.MyPlexAccount()
target = c.device('Plex Client')
sync_items_wd = c.syncItems(target.clientIdentifier)
srv = c.resource('Server Name').connect()
section = srv.library.section('Movies')
section.sync(VIDEO_QUALITY_3_MBPS_720p, client=target, limit=1, unwatched=True,
             title='Next best movie', sort='rating:desc')
```

class plexapi.library.**ShowSection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibrarySection*, *ShowEditMixins*, *SeasonEditMixins*, *EpisodeEditMixins*

Represents a *LibrarySection* section containing tv shows.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'show'

searchShows(***kwargs*)

Search for a show. See *search()* for usage.

searchSeasons(***kwargs*)

Search for a season. See *search()* for usage.

searchEpisodes(***kwargs*)

Search for an episode. See *search()* for usage.

recentlyAddedShows(*maxresults=50*)

Returns a list of recently added shows from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

recentlyAddedSeasons(*maxresults=50*)

Returns a list of recently added seasons from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

recentlyAddedEpisodes(*maxresults=50*)

Returns a list of recently added episodes from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

sync(*videoQuality*, *limit=None*, *unwatched=False*, ***kwargs*)

Add current Show library section as sync item for specified device. See description of *search()* for details about filtering / sorting and *sync()* for details on syncing libraries and possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *sync* module.
- **limit** (*int*) – maximum count of episodes to sync, unlimited if *None*.
- **unwatched** (*bool*) – if *True* watched videos wouldn't be synced.

Returns
an instance of created syncItem.

Return type
SyncItem

Example

```
from plexapi import myplex
from plexapi.sync import VIDEO_QUALITY_3_MBPS_720p

c = myplex.MyPlexAccount()
target = c.device('Plex Client')
sync_items_wd = c.syncItems(target.clientIdentifier)
srv = c.resource('Server Name').connect()
section = srv.library.section('TV-Shows')
section.sync(VIDEO_QUALITY_3_MBPS_720p, client=target, limit=1, unwatched=True,
             title='Next unwatched episode')
```

class plexapi.library.**MusicSection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibrarySection*, *ArtistEditMixins*, *AlbumEditMixins*, *TrackEditMixins*

Represents a *LibrarySection* section containing music artists.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'artist'

albums()

Returns a list of *Album* objects in this section.

stations()

Returns a list of *Playlist* stations in this section.

searchArtists(***kwargs*)

Search for an artist. See *search()* for usage.

searchAlbums(***kwargs*)

Search for an album. See *search()* for usage.

searchTracks(***kwargs*)

Search for a track. See *search()* for usage.

recentlyAddedArtists(*maxresults=50*)

Returns a list of recently added artists from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

recentlyAddedAlbums(*maxresults=50*)

Returns a list of recently added albums from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

recentlyAddedTracks(*maxresults=50*)

Returns a list of recently added tracks from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

sync(*bitrate*, *limit=None*, ***kwargs*)

Add current Music library section as sync item for specified device. See description of *search()* for details about filtering / sorting and *sync()* for details on syncing libraries and possible exceptions.

Parameters

- **bitrate** (*int*) – maximum bitrate for synchronized music, better use one of MUSIC_BITRATE_* values from the module *sync*.
- **limit** (*int*) – maximum count of tracks to sync, unlimited if *None*.

Returns
an instance of created syncItem.

Return type
SyncItem

Example

```
from plexapi import myplex
from plexapi.sync import AUDIO_BITRATE_320_KBPS

c = myplex.MyPlexAccount()
target = c.device('Plex Client')
sync_items_wd = c.syncItems(target.clientIdentifier)
srv = c.resource('Server Name').connect()
section = srv.library.section('Music')
section.sync(AUDIO_BITRATE_320_KBPS, client=target, limit=100, sort=
↪'addedAt:desc',
             title='New music')
```

sonicAdventure(*start: Track | int*, *end: Track | int*, ***kwargs: Any*) → list[Track]

Returns a list of tracks from this library section that are part of a sonic adventure. ID's should be of a track, other ID's will return an empty list or items itself or an error.

Parameters

- **start** (*Track | int*) – The *Track* or ID of the first track in the sonic adventure.
- **end** (*Track | int*) – The *Track* or ID of the last track in the sonic adventure.
- **kwargs** – Additional parameters to pass to *fetchItems()*.

Returns
a list of tracks from this library section that are part of a sonic adventure.

Return type
List[*Track*]

class plexapi.library.**PhotoSection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibrarySection*, *PhotoalbumEditMixins*, *PhotoEditMixins*

Represents a *LibrarySection* section containing photos.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'photo'

all(*libtype=None*, ***kwargs*)

Returns a list of all items from this library section. See description of *plexapi.library.LibrarySection.search()* for details about filtering / sorting.

collections(***kwargs*)

Returns a list of collections from this library section. See description of *search()* for details about filtering / sorting.

searchAlbums(***kwargs*)

Search for a photo album. See *search()* for usage.

searchPhotos(***kwargs*)

Search for a photo. See *search()* for usage.

recentlyAddedAlbums(*maxresults=50*)

Returns a list of recently added photo albums from this library section.

Parameters
maxresults (*int*) – Max number of items to return (default 50).

sync(*resolution*, *limit=None*, ***kwargs*)

Add current Music library section as sync item for specified device. See description of *search()* for details about filtering / sorting and *sync()* for details on syncing libraries and possible exceptions.

Parameters

- **resolution** (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module *sync*.
- **limit** (*int*) – maximum count of tracks to sync, unlimited if *None*.

Returns
an instance of created syncItem.

Return type
SyncItem

Example

```
from plexapi import myplex
from plexapi.sync import PHOTO_QUALITY_HIGH

c = myplex.MyPlexAccount()
target = c.device('Plex Client')
sync_items_wd = c.syncItems(target.clientIdentifier)
srv = c.resource('Server Name').connect()
section = srv.library.section('Photos')
section.sync(PHOTO_QUALITY_HIGH, client=target, limit=100, sort='addedAt:desc',
             title='Fresh photos')
```

class plexapi.library.**LibraryTimeline**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a LibrarySection timeline.

Variables

- **TAG** (*str*) – 'LibraryTimeline'
- **size** (*int*) – Unknown
- **allowSync** (*bool*) – Unknown
- **art** (*str*) – Relative path to art image.
- **content** (*str*) – "secondary"
- **identifier** (*str*) – "com.plexapp.plugins.library"
- **latestEntryTime** (*int*) – Epoch timestamp
- **mediaTagPrefix** (*str*) – "/system/bundle/media/flags/"
- **mediaTagVersion** (*int*) – Unknown
- **thumb** (*str*) – Relative path to library thumb image.
- **title1** (*str*) – Name of library section.
- **updateQueueSize** (*int*) – Number of items queued to update.

- **viewGroup** (*str*) – "secondary"
- **viewMode** (*int*) – Unknown

class plexapi.library.**Location**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single library Location.

Variables

- **TAG** (*str*) – 'Location'
- **id** (*int*) – Location path ID.
- **path** (*str*) – Path used for library..

class plexapi.library.**Hub**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Hub (or category) in the PlexServer search.

Variables

- **TAG** (*str*) – 'Hub'
- **context** (*str*) – The context of the hub.
- **hubKey** (*str*) – API URL for these specific hub items.
- **hubIdentifier** (*str*) – The identifier of the hub.
- **items** (*list*) – List of items in the hub (automatically loads all items if more is True).
- **key** (*str*) – API URL for the hub.
- **random** (*bool*) – True if the items in the hub are randomized.
- **more** (*bool*) – True if there are more items to load (call items to fetch all items).
- **size** (*int*) – The number of items in the hub.
- **style** (*str*) – The style of the hub.
- **title** (*str*) – The title of the hub.
- **type** (*str*) – The type of items in the hub.

items()

Returns a list of all items in the hub.

section()

Returns the *LibrarySection* this hub belongs to.

class plexapi.library.**LibraryMediaTag**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Base class of library media tags.

Variables

- **TAG** (*str*) – 'Directory'
- **count** (*int*) – The number of items where this tag is found.
- **filter** (*str*) – The URL filter for the tag.
- **id** (*int*) – The id of the tag.

- **key** (*str*) – API URL (/library/section/<librarySectionID>/all?<filter>).
- **librarySectionID** (*int*) – The library section ID where the tag is found.
- **librarySectionKey** (*str*) – API URL for the library section (/library/section/<librarySectionID>)
- **librarySectionTitle** (*str*) – The library title where the tag is found.
- **librarySectionType** (*int*) – The library type where the tag is found.
- **reason** (*str*) – The reason for the search result.
- **reasonID** (*int*) – The reason ID for the search result.
- **reasonTitle** (*str*) – The reason title for the search result.
- **score** (*float*) – The score for the search result.
- **type** (*str*) – The type of search result (tag).
- **tag** (*str*) – The title of the tag.
- **tagKey** (*str*) – The Plex Discover ratingKey (guid) for people.
- **tagType** (*int*) – The type ID of the tag.
- **tagValue** (*int*) – The value of the tag.
- **thumb** (*str*) – The URL for the thumbnail of the tag (if available).

items(**args*, ***kwargs*)

Return the list of items within this tag.

class plexapi.library.**Aperture**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Aperture library media tag.

Variables
TAGTYPE (*int*) – 202

class plexapi.library.**Art**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Art library media tag.

Variables
TAGTYPE (*int*) – 313

class plexapi.library.**Autotag**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Autotag library media tag.

Variables
TAGTYPE (*int*) – 207

class plexapi.library.**Chapter**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Chapter library media tag.

Variables
TAGTYPE (*int*) – 9

class plexapi.library.**Collection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Collection library media tag.

Variables
TAGTYPE (*int*) – 2

class plexapi.library.**Concert**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Concert library media tag.

Variables
TAGTYPE (*int*) – 306

class plexapi.library.**Country**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Country library media tag.

Variables
TAGTYPE (*int*) – 8

class plexapi.library.**Device**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Device library media tag.

Variables
TAGTYPE (*int*) – 206

class plexapi.library.**Director**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Director library media tag.

Variables
TAGTYPE (*int*) – 4

class plexapi.library.**Exposure**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Exposure library media tag.

Variables
TAGTYPE (*int*) – 203

class plexapi.library.**Format**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Format library media tag.

Variables
TAGTYPE (*int*) – 302

class plexapi.library.**Genre**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Genre library media tag.

Variables
TAGTYPE (*int*) – 1

class plexapi.library.**Guid**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Guid library media tag.

Variables
TAGTYPE (*int*) – 314

class plexapi.library.**ISO**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single ISO library media tag.

Variables
TAGTYPE (*int*) – 204

class plexapi.library.**Label**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Label library media tag.

Variables
TAGTYPE (*int*) – 11

class plexapi.library.**Lens**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Lens library media tag.

Variables
TAGTYPE (*int*) – 205

class plexapi.library.**Make**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Make library media tag.

Variables
TAGTYPE (*int*) – 200

class plexapi.library.**Marker**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Marker library media tag.

Variables
TAGTYPE (*int*) – 12

class plexapi.library.**MediaProcessingTarget**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single MediaProcessingTarget library media tag.

Variables

- **TAG** (*str*) – 'Tag'
- **TAGTYPE** (*int*) – 42

class plexapi.library.**Model**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Model library media tag.

Variables
TAGTYPE (*int*) – 201

class plexapi.library.**Mood**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Mood library media tag.

Variables
TAGTYPE (*int*) – 300

class plexapi.library.**Network**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Network library media tag.

Variables
TAGTYPE (*int*) – 319

class plexapi.library.**Place**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Place library media tag.

Variables
TAGTYPE (*int*) – 400

class plexapi.library.**Poster**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Poster library media tag.

Variables
TAGTYPE (*int*) – 312

class plexapi.library.**Producer**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Producer library media tag.

Variables
TAGTYPE (*int*) – 7

class plexapi.library.**RatingImage**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single RatingImage library media tag.

Variables
TAGTYPE (*int*) – 316

class plexapi.library.**Review**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Review library media tag.

Variables
TAGTYPE (*int*) – 10

class plexapi.library.**Role**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Role library media tag.

Variables
TAGTYPE (*int*) – 6

class plexapi.library.**Similar**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Similar library media tag.

Variables
TAGTYPE (*int*) – 305

class plexapi.library.**Studio**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Studio library media tag.

Variables
TAGTYPE (*int*) – 318

class plexapi.library.**Style**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Style library media tag.

Variables
TAGTYPE (*int*) – 301

class plexapi.library.**Tag**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Tag library media tag.

Variables
TAGTYPE (*int*) – 0

class plexapi.library.**Theme**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Theme library media tag.

Variables
TAGTYPE (*int*) – 317

class plexapi.library.**Writer**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *LibraryMediaTag*

Represents a single Writer library media tag.

Variables
TAGTYPE (*int*) – 5

class plexapi.library.**FilteringType**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single filtering Type object for a library.

Variables

- **TAG** (*str*) – 'Type'
- **active** (*bool*) – True if this filter type is currently active.
- **fields** (List<*FilteringField*>) – List of field objects.
- **filters** (List<*FilteringFilter*>) – List of filter objects.

- **key** (*str*) – The API URL path for the libtype filter.
- **sorts** (List<*FilteringSort*>) – List of sort objects.
- **title** (*str*) – The title for the libtype filter.
- **type** (*str*) – The libtype for the filter.

class plexapi.library.**FilteringFilter**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Filter object for a *FilteringType*.

Variables

- **TAG** (*str*) – 'Filter'
- **filter** (*str*) – The key for the filter.
- **filterType** (*str*) – The *FilteringFieldType* type (string, boolean, integer, date, etc).
- **key** (*str*) – The API URL path for the filter.
- **title** (*str*) – The title of the filter.
- **type** (*str*) – 'filter'

class plexapi.library.**FilteringSort**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Sort object for a *FilteringType*.

Variables

- **TAG** (*str*) – 'Sort'
- **active** (*bool*) – True if the sort is currently active.
- **activeDirection** (*str*) – The currently active sorting direction.
- **default** (*str*) – The currently active default sorting direction.
- **defaultDirection** (*str*) – The default sorting direction.
- **descKey** (*str*) – The URL key for sorting with desc.
- **firstCharacterKey** (*str*) – API URL path for first character endpoint.
- **key** (*str*) – The URL key for the sorting.
- **title** (*str*) – The title of the sorting.

class plexapi.library.**FilteringField**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Field object for a *FilteringType*.

Variables

- **TAG** (*str*) – 'Field'
- **key** (*str*) – The URL key for the filter field.
- **title** (*str*) – The title of the filter field.
- **type** (*str*) – The *FilteringFieldType* type (string, boolean, integer, date, etc).
- **subType** (*str*) – The subtype of the filter (decade, rating, etc).

class plexapi.library.**FilteringFieldType**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single FieldType for library filtering.

Variables

- **TAG** (*str*) – 'FieldType'
- **type** (*str*) – The filtering data type (string, boolean, integer, date, etc).
- **operators** (List<*FilteringOperator*>) – List of operator objects.

class plexapi.library.**FilteringOperator**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents an single Operator for a *FilteringFieldType*.

Variables

- **TAG** (*str*) – 'Operator'
- **key** (*str*) – The URL key for the operator.
- **title** (*str*) – The title of the operator.

class plexapi.library.**FilterChoice**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single FilterChoice object. These objects are gathered when using filters while searching for library items and is the object returned in the result set of *listFilterChoices()*.

Variables

- **TAG** (*str*) – 'Directory'
- **fastKey** (*str*) – API URL path to quickly list all items with this filter choice. (/library/sections/<section>/all?genre=<key>)
- **key** (*str*) – The id value of this filter choice.
- **thumb** (*str*) – Thumbnail URL for the filter choice.
- **title** (*str*) – The title of the filter choice.
- **type** (*str*) – The filter type (genre, contentRating, etc).

items()

Returns a list of items for this filter choice.

class plexapi.library.**ManagedHub**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Managed Hub (recommendation) inside a library.

Variables

- **TAG** (*str*) – 'Hub'
- **deletable** (*bool*) – True if the Hub can be deleted (promoted collection).
- **homeVisibility** (*str*) – Promoted home visibility (none, all, admin, or shared).
- **identifier** (*str*) – Hub identifier for the managed hub.
- **promotedToOwnHome** (*bool*) – Promoted to own home.
- **promotedToRecommended** (*bool*) – Promoted to recommended.

- **promotedToSharedHome** (*bool*) – Promoted to shared home.
- **recommendationsVisibility** (*str*) – Promoted recommendation visibility (none or all).
- **title** (*str*) – Title of managed hub.

move(*after=None*)

Move a managed hub to a new position in the library's Managed Recommendations.

Parameters
after (*obj*) – *ManagedHub* object to move the item after in the collection.

Raises
plexapi.exceptions.BadRequest – When trying to move a Hub that is not a Managed Recommendation.

remove()

Removes a managed hub from the library's Managed Recommendations.

Raises
plexapi.exceptions.BadRequest – When trying to remove a Hub that is not a Managed Recommendation or when the Hub cannot be removed.

updateVisibility(*recommended=None*, *home=None*, *shared=None*)

Update the managed hub's visibility settings.

Parameters

- **recommended** (*bool*) – True to make visible on your Library Recommended page. False to hide. Default None.
- **home** (*bool*) – True to make visible on your Home page. False to hide. Default None.
- **shared** (*bool*) – True to make visible on your Friends' Home page. False to hide. Default None.

Example

```
managedHub.updateVisibility(recommended=True, home=True, shared=False).reload()
# or using chained methods
managedHub.promoteRecommended().promoteHome().demoteShared().reload()
```

promoteRecommended()

Show the managed hub on your Library Recommended Page.

demoteRecommended()

Hide the managed hub on your Library Recommended Page.

promoteHome()

Show the managed hub on your Home Page.

demoteHome()

Hide the manged hub on your Home Page.

promoteShared()

Show the managed hub on your Friends' Home Page.

demoteShared()

Hide the managed hub on your Friends' Home Page.

class plexapi.library.**Folder**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Folder inside a library.

Variables

- **key** (*str*) – Url key for folder.
- **title** (*str*) – Title of folder.

subfolders()

Returns a list of available *Folder* for this folder. Continue down subfolders until a mediaType is found.

allSubfolders()

Returns a list of all available *Folder* for this folder. Only returns *Folder*.

class plexapi.library.**FirstCharacter**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a First Character element from a library.

Variables

- **key** (*str*) – Url key for character.
- **size** (*str*) – Total amount of library items starting with this character.
- **title** (*str*) – Character (#, !, A, B, C, ...).

class plexapi.library.**Path**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single directory Path.

Variables

- **TAG** (*str*) – 'Path'
- **home** (*bool*) – True if the path is the home directory
- **key** (*str*) – API URL (/services/browse/<base64path>)
- **network** (*bool*) – True if path is a network location
- **path** (*str*) – Full path to folder
- **title** (*str*) – Folder name

browse(*includeFiles=True*)

Alias for *browse()*.

walk()

Alias for *walk()*.

class plexapi.library.**File**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single File.

Variables

- **TAG** (*str*) – 'File'
- **key** (*str*) – API URL (/services/browse/<base64path>)
- **path** (*str*) – Full path to file

- **title** (*str*) – File name

class plexapi.library.**Common**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Common element from a library. This object lists common fields between multiple objects.

Variables

- **TAG** (*str*) – 'Common'
- **collections** (List<*Collection*>) – List of collection objects.
- **contentRating** (*str*) – Content rating of the items.
- **countries** (List<*Country*>) – List of countries objects.
- **directors** (List<*Director*>) – List of director objects.
- **editionTitle** (*str*) – Edition title of the items.
- **fields** (List<*Field*>) – List of field objects.
- **genres** (List<*Genre*>) – List of genre objects.
- **grandparentRatingKey** (*int*) – Grandparent rating key of the items.
- **grandparentTitle** (*str*) – Grandparent title of the items.
- **guid** (*str*) – Plex GUID of the items.
- **guids** (List<*Guid*>) – List of guid objects.
- **index** (*int*) – Index of the items.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **labels** (List<*Label*>) – List of label objects.
- **mixedFields** (*List<str>*) – List of mixed fields.
- **moods** (List<*Mood*>) – List of mood objects.
- **originallyAvailableAt** (*datetime*) – Datetime of the release date of the items.
- **parentRatingKey** (*int*) – Parent rating key of the items.
- **parentTitle** (*str*) – Parent title of the items.
- **producers** (List<*Producer*>) – List of producer objects.
- **ratingKey** (*int*) – Rating key of the items.
- **ratings** (List<*Rating*>) – List of rating objects.
- **roles** (List<*Role*>) – List of role objects.
- **studio** (*str*) – Studio name of the items.
- **styles** (List<*Style*>) – List of style objects.
- **summary** (*str*) – Summary of the items.
- **tagline** (*str*) – Tagline of the items.
- **tags** (List<*Tag*>) – List of tag objects.
- **title** (*str*) – Title of the items.
- **titleSort** (*str*) – Title to use when sorting of the items.

- **type** (*str*) – Type of the media (common).
- **writers** (List<*Writer*>) – List of writer objects.
- **year** (*int*) – Year of the items.

property commonType

Returns the media type of the common items.

property ratingKeys

Returns a list of rating keys for the common items.

items()

Returns a list of the common items.

CHAPTER

TWELVE

MEDIA PLEXAPI.MEDIA

class plexapi.media.**Media**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Container object for all MediaPart objects. Provides useful data about the video or audio this media belong to such as video framerate, resolution, etc.

Variables

- **TAG** (*str*) – 'Media'
- **aspectRatio** (*float*) – The aspect ratio of the media (ex: 2.35).
- **audioChannels** (*int*) – The number of audio channels of the media (ex: 6).
- **audioCodec** (*str*) – The audio codec of the media (ex: ac3).
- **audioProfile** (*str*) – The audio profile of the media (ex: dts).
- **bitrate** (*int*) – The bitrate of the media (ex: 1624).
- **container** (*str*) – The container of the media (ex: avi).
- **duration** (*int*) – The duration of the media in milliseconds (ex: 6990483).
- **height** (*int*) – The height of the media in pixels (ex: 256).
- **id** (*int*) – The unique ID for this media on the server.
- **has64bitOffsets** (*bool*) – True if video has 64 bit offsets.
- **hasVoiceActivity** (*bool*) – True if video has voice activity analyzed.
- **optimizedForStreaming** (*bool*) – True if video is optimized for streaming.
- **parts** (List<*MediaPart*>) – List of media part objects.
- **proxyType** (*int*) – Equals 42 for optimized versions.
- **target** (*str*) – The media version target name.
- **title** (*str*) – The title of the media.
- **videoCodec** (*str*) – The video codec of the media (ex: ac3).
- **videoFrameRate** (*str*) – The video frame rate of the media (ex: 24p).
- **videoProfile** (*str*) – The video profile of the media (ex: high).
- **videoResolution** (*str*) – The video resolution of the media (ex: sd).
- **width** (*int*) – The width of the video in pixels (ex: 608).
- **Photo_only_attributes** – The following attributes are only available for photos.

- aperture (str): The aperture used to take the photo.
- exposure (str): The exposure used to take the photo.
- iso (int): The iso used to take the photo.
- lens (str): The lens used to take the photo.
- make (str): The make of the camera used to take the photo.
- model (str): The model of the camera used to take the photo.

property isOptimizedVersion

Returns True if the media is a Plex optimized version.

class plexapi.media.**MediaPart**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single media part (often a single file) for the media this belongs to.

Variables

- **TAG** (*str*) – 'Part'
- **accessible** (*bool*) – True if the file is accessible. Requires reloading the media with checkFiles=True. Refer to *reload()*.
- **audioProfile** (*str*) – The audio profile of the file.
- **container** (*str*) – The container type of the file (ex: avi).
- **decision** (*str*) – Unknown.
- **deepAnalysisVersion** (*int*) – The Plex deep analysis version for the file.
- **duration** (*int*) – The duration of the file in milliseconds.
- **exists** (*bool*) – True if the file exists. Requires reloading the media with checkFiles=True. Refer to *reload()*.
- **file** (*str*) – The path to this file on disk (ex: /media/Movies/Cars (2006)/Cars (2006).mkv)
- **has64bitOffsets** (*bool*) – True if the file has 64 bit offsets.
- **hasThumbnail** (*bool*) – True if the file (track) has an embedded thumbnail.
- **id** (*int*) – The unique ID for this media part on the server.
- **indexes** (*str, None*) – sd if the file has generated preview (BIF) thumbnails.
- **key** (*str*) – API URL (ex: /library/parts/46618/1389985872/file.mkv).
- **optimizedForStreaming** (*bool*) – True if the file is optimized for streaming.
- **packetLength** (*int*) – The packet length of the file.
- **requiredBandwidths** (*str*) – The required bandwidths to stream the file.
- **selected** (*bool*) – True if this media part is selected.
- **size** (*int*) – The size of the file in bytes (ex: 733884416).
- **streams** (List<*MediaPartStream*>) – List of stream objects.
- **syncItemId** (*int*) – The unique ID for this media part if it is synced.
- **syncState** (*str*) – The sync state for this media part.
- **videoProfile** (*str*) – The video profile of the file.

property hasPreviewThumbnails

Returns True if the media part has generated preview (BIF) thumbnails.

videoStreams()

Returns a list of *VideoStream* objects in this MediaPart.

audioStreams()

Returns a list of *AudioStream* objects in this MediaPart.

subtitleStreams()

Returns a list of *SubtitleStream* objects in this MediaPart.

lyricStreams()

Returns a list of *LyricStream* objects in this MediaPart.

setSelectedAudioStream(*stream*)

Set the selected *AudioStream* for this MediaPart.

Parameters

stream (*AudioStream*) – Audio stream to set as selected

setSelectedSubtitleStream(*stream*)

Set the selected *SubtitleStream* for this MediaPart.

Parameters

stream (*SubtitleStream*) – Subtitle stream to set as selected.

resetSelectedSubtitleStream()

Set the selected subtitle of this MediaPart to 'None'.

class plexapi.media.**MediaPartStream**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Base class for media streams. These consist of video, audio, subtitles, and lyrics.

Variables

- **bitrate** (*int*) – The bitrate of the stream.
- **codec** (*str*) – The codec of the stream (ex: srt, ac3, mpeg4).
- **default** (*bool*) – True if this is the default stream.
- **displayTitle** (*str*) – The display title of the stream.
- **extendedDisplayTitle** (*str*) – The extended display title of the stream.
- **key** (*str*) – API URL (/library/streams/<id>)
- **id** (*int*) – The unique ID for this stream on the server.
- **index** (*int*) – The index of the stream.
- **language** (*str*) – The language of the stream (ex: English,).
- **languageCode** (*str*) – The ASCII language code of the stream (ex: eng, tha).
- **languageTag** (*str*) – The two letter language tag of the stream (ex: en, fr).
- **requiredBandwidths** (*str*) – The required bandwidths to stream the file.
- **selected** (*bool*) – True if this stream is selected.
- **streamType** (*int*) – The stream type (1= *VideoStream*, 2= *AudioStream*, 3= *SubtitleStream*).

- **title** (*str*) – The title of the stream.
- **type** (*int*) – Alias for streamType.

class plexapi.media.**VideoStream**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaPartStream*

Represents a video stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 1
- **anamorphic** (*str*) – If the video is anamorphic.
- **bitDepth** (*int*) – The bit depth of the video stream (ex: 8).
- **cabac** (*int*) – The context-adaptive binary arithmetic coding.
- **chromaLocation** (*str*) – The chroma location of the video stream.
- **chromaSubsampling** (*str*) – The chroma subsampling of the video stream (ex: 4:2:0).
- **codecID** (*str*) – The codec ID (ex: XVID).
- **codedHeight** (*int*) – The coded height of the video stream in pixels.
- **codedWidth** (*int*) – The coded width of the video stream in pixels.
- **colorPrimaries** (*str*) – The color primaries of the video stream.
- **colorRange** (*str*) – The color range of the video stream.
- **colorSpace** (*str*) – The color space of the video stream (ex: bt2020).
- **colorTrc** (*str*) – The color trc of the video stream.
- **DOVIBLCompatID** (*int*) – Dolby Vision base layer compatibility ID.
- **DOVIBLPresent** (*bool*) – True if Dolby Vision base layer is present.
- **DOVIELPresent** (*bool*) – True if Dolby Vision enhancement layer is present.
- **DOVILevel** (*int*) – Dolby Vision level.
- **DOVIPresent** (*bool*) – True if Dolby Vision is present.
- **DOVIProfile** (*int*) – Dolby Vision profile.
- **DOVIRPUPresent** (*bool*) – True if Dolby Vision reference processing unit is present.
- **DOVIVersion** (*float*) – The Dolby Vision version.
- **duration** (*int*) – The duration of video stream in milliseconds.
- **frameRate** (*float*) – The frame rate of the video stream (ex: 23.976).
- **frameRateMode** (*str*) – The frame rate mode of the video stream.
- **hasScalingMatrix** (*bool*) – True if video stream has a scaling matrix.
- **height** (*int*) – The height of the video stream in pixels (ex: 1080).
- **level** (*int*) – The codec encoding level of the video stream (ex: 41).
- **profile** (*str*) – The profile of the video stream (ex: asp).
- **pixelAspectRatio** (*str*) – The pixel aspect ratio of the video stream.

- **pixelFormat** (*str*) – The pixel format of the video stream.
- **refFrames** (*int*) – The number of reference frames of the video stream.
- **scanType** (*str*) – The scan type of the video stream (ex: progressive).
- **streamIdentifier** (*int*) – The stream identifier of the video stream.
- **width** (*int*) – The width of the video stream in pixels (ex: 1920).

class plexapi.media.**AudioStream**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaPartStream*

Represents a audio stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 2
- **audioChannelLayout** (*str*) – The audio channel layout of the audio stream (ex: 5.1(side)).
- **bitDepth** (*int*) – The bit depth of the audio stream (ex: 16).
- **bitrateMode** (*str*) – The bitrate mode of the audio stream (ex: cbr).
- **channels** (*int*) – The number of audio channels of the audio stream (ex: 6).
- **duration** (*int*) – The duration of audio stream in milliseconds.
- **profile** (*str*) – The profile of the audio stream.
- **samplingRate** (*int*) – The sampling rate of the audio stream (ex: xxx)
- **streamIdentifier** (*int*) – The stream identifier of the audio stream.
- **visualImpaired** (*bool*) – True if this is a visually impaired (AD) audio stream.
- **Track_only_attributes** – The following attributes are only available for tracks.
  - albumGain (float): The gain for the album.
  - albumPeak (float): The peak for the album.
  - albumRange (float): The range for the album.
  - endRamp (str): The end ramp for the track.
  - gain (float): The gain for the track.
  - loudness (float): The loudness for the track.
  - lra (float): The lra for the track.
  - peak (float): The peak for the track.
  - startRamp (str): The start ramp for the track.

setSelected()

Sets this audio stream as the selected audio stream. Alias for *setSelectedAudioStream()*.

levels(*subSample=128*)

Returns a list of *Level* objects for this AudioStream. Only available for Tracks which have been analyzed for loudness.

Variables

subSample (*int*) – The number of loudness samples to return. Default 128.

class plexapi.media.**SubtitleStream**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaPartStream*

Represents a audio stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 3
- **canAutoSync** (*bool*) – True if the subtitle stream can be auto synced.
- **container** (*str*) – The container of the subtitle stream.
- **forced** (*bool*) – True if this is a forced subtitle.
- **format** (*str*) – The format of the subtitle stream (ex: srt).
- **headerCompression** (*str*) – The header compression of the subtitle stream.
- **hearingImpaired** (*bool*) – True if this is a hearing impaired (SDH) subtitle.
- **perfectMatch** (*bool*) – True if the on-demand subtitle is a perfect match.
- **providerTitle** (*str*) – The provider title where the on-demand subtitle is downloaded from.
- **score** (*int*) – The match score (download count) of the on-demand subtitle.
- **sourceKey** (*str*) – The source key of the on-demand subtitle.
- **transient** (*str*) – Unknown.
- **userID** (*int*) – The user id of the user that downloaded the on-demand subtitle.

setSelected()

Sets this subtitle stream as the selected subtitle stream. Alias for *setSelectedSubtitleStream()*.

class plexapi.media.**LyricStream**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaPartStream*

Represents a lyric stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 4
- **format** (*str*) – The format of the lyric stream (ex: lrc).
- **minLines** (*int*) – The minimum number of lines in the (timed) lyric stream.
- **provider** (*str*) – The provider of the lyric stream (ex: com.plexapp.agents.lyricfind).
- **timed** (*bool*) – True if the lyrics are timed to the track.

class plexapi.media.**Session**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a current session.

Variables

- **TAG** (*str*) – 'Session'
- **id** (*str*) – The unique identifier for the session.

- **bandwidth** (*int*) – The Plex streaming brain reserved bandwidth for the session.
- **location** (*str*) – The location of the session (lan, wan, or cellular)

class plexapi.media.**TranscodeSession**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a current transcode session.

Variables

- **TAG** (*str*) – 'TranscodeSession'
- **audioChannels** (*int*) – The number of audio channels of the transcoded media.
- **audioCodec** (*str*) – The audio codec of the transcoded media.
- **audioDecision** (*str*) – The transcode decision for the audio stream.
- **complete** (*bool*) – True if the transcode is complete.
- **container** (*str*) – The container of the transcoded media.
- **context** (*str*) – The context for the transcode session.
- **duration** (*int*) – The duration of the transcoded media in milliseconds.
- **height** (*int*) – The height of the transcoded media in pixels.
- **key** (*str*) – API URL (ex: /transcode/sessions/<id>).
- **maxOffsetAvailable** (*float*) – Unknown.
- **minOffsetAvailable** (*float*) – Unknown.
- **progress** (*float*) – The progress percentage of the transcode.
- **protocol** (*str*) – The protocol of the transcode.
- **remaining** (*int*) – Unknown.
- **size** (*int*) – The size of the transcoded media in bytes.
- **sourceAudioCodec** (*str*) – The audio codec of the source media.
- **sourceVideoCodec** (*str*) – The video codec of the source media.
- **speed** (*float*) – The speed of the transcode.
- **subtitleDecision** (*str*) – The transcode decision for the subtitle stream
- **throttled** (*bool*) – True if the transcode is throttled.
- **timestamp** (*int*) – The epoch timestamp when the transcode started.
- **transcodeHwDecoding** (*str*) – The hardware transcoding decoder engine.
- **transcodeHwDecodingTitle** (*str*) – The title of the hardware transcoding decoder engine.
- **transcodeHwEncoding** (*str*) – The hardware transcoding encoder engine.
- **transcodeHwEncodingTitle** (*str*) – The title of the hardware transcoding encoder engine.
- **transcodeHwFullPipeline** (*str*) – True if hardware decoding and encoding is being used for the transcode.
- **transcodeHwRequested** (*str*) – True if hardware transcoding was requested for the transcode.

- **videoCodec** (*str*) – The video codec of the transcoded media.
- **videoDecision** (*str*) – The transcode decision for the video stream.
- **width** (*str*) – The width of the transcoded media in pixels.

class plexapi.media.**TranscodeJob**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents an Optimizing job. TrancodeJobs are the process for optimizing conversions. Active or paused optimization items. Usually one item as a time.

class plexapi.media.**Optimized**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Optimized item. Optimized items are optimized and queued conversions items.

items()

Returns a list of all `Video` objects in this optimized item.

remove()

Remove an Optimized item

rename(*title*)

Rename an Optimized item

reprocess(*ratingKey*)

Reprocess a removed Conversion item that is still a listed Optimize item

class plexapi.media.**Conversion**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Conversion item. Conversions are items queued for optimization or being actively optimized.

remove()

Remove Conversion from queue

move(*after*)

Move Conversion items position in queue after (int): Place item after specified playQueueItemID. '-1' is the active conversion.

Example:

Move 5th conversion Item to active conversion
conversions[4].move('-1')

Move 4th conversion Item to 3rd in conversion queue
conversions[3].move(conversions[1].playQueueItemID)

class plexapi.media.**MediaTag**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Base class for media tags used for filtering and searching your library items or navigating the metadata of media items in your library. Tags are the construct used for things such as Country, Director, Genre, etc.

Variables

- **filter** (*str*) – The library filter for the tag.
- **id** (*int*) – Tag ID (This seems meaningless except to use it as a unique id).
- **key** (*str*) – API URL (/library/section/<librarySectionID>/all?<filter>).
- **role** (*str*) – The name of the character role for *Role* only.

- **tag** (*str*) – Name of the tag. This will be Animation, SciFi etc for Genres. The name of person for Directors and Roles (ex: Animation, Stephen Graham, etc).
- **tagKey** (*str*) – Plex GUID for the actor/actress for *Role* only.
- **thumb** (*str*) – URL to thumbnail image for *Role* only.

items()

Return the list of items within this tag.

class plexapi.media.**Collection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Collection media tag.

Variables

- **TAG** (*str*) – 'Collection'
- **FILTER** (*str*) – 'collection'

collection()

Return the *Collection* object for this collection tag.

class plexapi.media.**Country**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Country media tag.

Variables

- **TAG** (*str*) – 'Country'
- **FILTER** (*str*) – 'country'

class plexapi.media.**Director**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Director media tag.

Variables

- **TAG** (*str*) – 'Director'
- **FILTER** (*str*) – 'director'

class plexapi.media.**Format**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Format media tag.

Variables

- **TAG** (*str*) – 'Format'
- **FILTER** (*str*) – 'format'

class plexapi.media.**Genre**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Genre media tag.

Variables

- **TAG** (*str*) – 'Genre'
- **FILTER** (*str*) – 'genre'

class plexapi.media.**Label**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Label media tag.

Variables

- **TAG** (*str*) – 'Label'
- **FILTER** (*str*) – 'label'

class plexapi.media.**Mood**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Mood media tag.

Variables

- **TAG** (*str*) – 'Mood'
- **FILTER** (*str*) – 'mood'

class plexapi.media.**Producer**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Producer media tag.

Variables

- **TAG** (*str*) – 'Producer'
- **FILTER** (*str*) – 'producer'

class plexapi.media.**Role**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Role (actor/actress) media tag.

Variables

- **TAG** (*str*) – 'Role'
- **FILTER** (*str*) – 'role'

class plexapi.media.**Similar**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Similar media tag.

Variables

- **TAG** (*str*) – 'Similar'
- **FILTER** (*str*) – 'similar'

class plexapi.media.**Style**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Style media tag.

Variables

- **TAG** (*str*) – 'Style'
- **FILTER** (*str*) – 'style'

class plexapi.media.**Subformat**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Subformat media tag.

Variables

- **TAG** (*str*) – 'Subformat'
- **FILTER** (*str*) – 'subformat'

class plexapi.media.**Tag**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Tag media tag.

Variables

- **TAG** (*str*) – 'Tag'
- **FILTER** (*str*) – 'tag'

class plexapi.media.**Writer**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *MediaTag*

Represents a single Writer media tag.

Variables

- **TAG** (*str*) – 'Writer'
- **FILTER** (*str*) – 'writer'

class plexapi.media.**Guid**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Guid media tag.

Variables

- **TAG** (*str*) – 'Guid'
- **id** (*id*) – The guid for external metadata sources (e.g. IMDB, TMDB, TVDB, MBID).

class plexapi.media.**Image**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Image media tag.

Variables

- **TAG** (*str*) – 'Image'
- **alt** (*str*) – The alt text for the image.
- **type** (*str*) – The type of image (e.g. coverPoster, background, snapshot).
- **url** (*str*) – The API URL (/library/metadata/<ratingKey>/thumb/<thumbid>).

class plexapi.media.**Rating**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Rating media tag.

Variables

- **TAG** (*str*) – 'Rating'

- **image** (*str*) – The uri for the rating image (e.g. imdb://image.rating, rottentomatoes://image.rating.ripe, rottentomatoes://image.rating.upright, themoviedb://image.rating).
- **type** (*str*) – The type of rating (e.g. audience or critic).
- **value** (*float*) – The rating value.

class plexapi.media.**Review**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Review for a Movie.

Variables

- **TAG** (*str*) – 'Review'
- **filter** (*str*) – The library filter for the review.
- **id** (*int*) – The ID of the review.
- **image** (*str*) – The image uri for the review.
- **link** (*str*) – The url to the online review.
- **source** (*str*) – The source of the review.
- **tag** (*str*) – The name of the reviewer.
- **text** (*str*) – The text of the review.

class plexapi.media.**UltraBlurColors**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single UltraBlurColors media tag.

Variables

- **TAG** (*str*) – 'UltraBlurColors'
- **bottomLeft** (*str*) – The bottom left hex color.
- **bottomRight** (*str*) – The bottom right hex color.
- **topLeft** (*str*) – The top left hex color.
- **topRight** (*str*) – The top right hex color.

class plexapi.media.**BaseResource**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Base class for all Art, Poster, and Theme objects.

Variables

- **TAG** (*str*) – 'Photo' or 'Track'
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **provider** (*str*) – The source of the resource. 'local' for local files (e.g. theme.mp3), None if uploaded or agent-/plugin-supplied.
- **ratingKey** (*str*) – Unique key identifying the resource.
- **selected** (*bool*) – True if the resource is currently selected.
- **thumb** (*str*) – The URL to retrieve the resource thumbnail.

property resourceFilepath

Returns the file path to the resource in the Plex Media Server data directory. Note: Returns the URL if the resource is not stored locally.

class plexapi.media.**Art**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *BaseResource*

Represents a single Art object.

class plexapi.media.**Logo**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *BaseResource*

Represents a single Logo object.

class plexapi.media.**Poster**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *BaseResource*

Represents a single Poster object.

class plexapi.media.**SquareArt**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *BaseResource*

Represents a single Square Art object.

class plexapi.media.**Theme**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *BaseResource*

Represents a single Theme object.

class plexapi.media.**Chapter**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Chapter media tag.

Variables

- **TAG** (*str*) – 'Chapter'
- **end** (*int*) – The end time of the chapter in milliseconds.
- **filter** (*str*) – The library filter for the chapter.
- **id** (*int*) – The ID of the chapter.
- **index** (*int*) – The index of the chapter.
- **tag** (*str*) – The name of the chapter.
- **title** (*str*) – The title of the chapter.
- **thumb** (*str*) – The URL to retrieve the chapter thumbnail.
- **start** (*int*) – The start time of the chapter in milliseconds.

class plexapi.media.**Marker**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Marker media tag.

Variables

- **TAG** (*str*) – 'Marker'
- **end** (*int*) – The end time of the marker in milliseconds.
- **final** (*bool*) – True if the marker is the final credits marker.

- **id** (*int*) – The ID of the marker.
- **type** (*str*) – The type of marker.
- **start** (*int*) – The start time of the marker in milliseconds.
- **version** (*int*) – The Plex marker version.

property first

Returns True if the marker in the first credits marker.

class plexapi.media.**Field**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Field.

Variables

- **TAG** (*str*) – 'Field'
- **locked** (*bool*) – True if the field is locked.
- **name** (*str*) – The name of the field.

class plexapi.media.**SearchResult**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single SearchResult.

Variables

TAG (*str*) – 'SearchResult'

class plexapi.media.**Agent**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Agent.

Variables

TAG (*str*) – 'Agent'

class plexapi.media.**AgentMediaType**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Agent*

Represents a single Agent MediaType.

Variables

TAG (*str*) – 'MediaType'

class plexapi.media.**Availability**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single online streaming service Availability.

Variables

- **TAG** (*str*) – 'Availability'
- **country** (*str*) – The streaming service country.
- **offerType** (*str*) – Subscription, buy, or rent from the streaming service.
- **platform** (*str*) – The platform slug for the streaming service.
- **platformColorThumb** (*str*) – Thumbnail icon for the streaming service.
- **platformInfo** (*str*) – The streaming service platform info.

- **platformUrl** (*str*) – The URL to the media on the streaming service.
- **price** (*float*) – The price to buy or rent from the streaming service.
- **priceDescription** (*str*) – The display price to buy or rent from the streaming service.
- **quality** (*str*) – The video quality on the streaming service.
- **title** (*str*) – The title of the streaming service.
- **url** (*str*) – The Plex availability URL.

class plexapi.media.**Level**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single loudness Level sample for an AudioStream.

Variables

loudness (*float*) – Loudness level value

class plexapi.media.**CommonSenseMedia**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single CommonSenseMedia media tag. Note: This object is only loaded with partial data from a Plex Media Server. Call *reload()* to load the full data from Plex Discover (Plex Pass required).

Variables

- **TAG** (*str*) – 'CommonSenseMedia'
- **ageRatings** (List<*AgeRating*>) – List of AgeRating objects.
- **anyGood** (*str*) – A brief description of the media's quality.
- **id** (*int*) – The ID of the CommonSenseMedia tag.
- **key** (*str*) – The unique key for the CommonSenseMedia tag.
- **oneLiner** (*str*) – A brief description of the CommonSenseMedia tag.
- **parentalAdvisoryTopics** (List<*ParentalAdvisoryTopic*>) – List of ParentalAdvisoryTopic objects.
- **parentsNeedToKnow** (*str*) – A brief description of what parents need to know about the media.
- **talkingPoints** (List<*TalkingPoint*>) – List of TalkingPoint objects.

Example

```
from plexapi.server import PlexServer
plex = PlexServer('http://localhost:32400', token='xxxxxxxxxxxxxxxxxxxx')

# Retrieve the Common Sense Media info for a movie
movie = plex.library.section('Movies').get('Cars')
commonSenseMedia = movie.commonSenseMedia
ageRating = commonSenseMedia.ageRatings[0].age

# Load the Common Sense Media info from Plex Discover (Plex Pass required)
commonSenseMedia.reload()
parentalAdvisoryTopics = commonSenseMedia.parentalAdvisoryTopics
talkingPoints = commonSenseMedia.talkingPoints
```

class plexapi.media.**AgeRating**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single AgeRating for a Common Sense Media tag.

Variables

- **TAG** (*str*) – 'AgeRating'
- **age** (*float*) – The age rating (e.g. 13, 17).
- **ageGroup** (*str*) – The age group for the rating (e.g. Little Kids, Teens, etc.).
- **rating** (*float*) – The star rating (out of 5).
- **ratingCount** (*int*) – The number of ratings contributing to the star rating.
- **type** (*str*) – The type of rating (official, adult, child).

class plexapi.media.**TalkingPoint**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single TalkingPoint for a Common Sense Media tag.

Variables

- **TAG** (*str*) – 'TalkingPoint'
- **tag** (*str*) – The description of the talking point.

class plexapi.media.**ParentalAdvisoryTopic**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single ParentalAdvisoryTopic for a Common Sense Media tag.

Variables

- **TAG** (*str*) – 'ParentalAdvisoryTopic'
- **id** (*str*) – The ID of the topic (e.g. violence, language, etc.).
- **label** (*str*) – The label for the topic (e.g. Violence & Scariness, Language, etc.).
- **positive** (*bool*) – Whether the topic is considered positive.
- **rating** (*float*) – The rating of the topic (out of 5).
- **tag** (*str*) – The description of the parental advisory topic.

CHAPTER

THIRTEEN

MIXINS PLEXAPI.MIXINS

PlexAPI Mixins Module

This module contains mixins for Plex objects.

class plexapi.mixins.**AdvancedSettingsMixin**

Bases: object

Mixin for Plex objects that can have advanced settings.

preferences()

Returns a list of *Preferences* objects.

preference(*pref*)

Returns a *Preferences* object for the specified pref.

Parameters

pref (*str*) – The id of the preference to return.

editAdvanced(***kwargs*)

Edit a Plex object's advanced settings.

defaultAdvanced()

Edit all of a Plex object's advanced settings to default.

class plexapi.mixins.**AddedAtMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have an added at date.

editAddedAt(*addedAt*, *locked=True*)

Edit the added at date.

Parameters

- **addedAt** (*int or str or datetime*) – The new value as a unix timestamp (int), "YYYY-MM-DD" (str), or datetime object.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**AudienceRatingMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have an audience rating.

editAudienceRating(*audienceRating*, *locked=True*)

Edit the audience rating.

Parameters

- **audienceRating** (*float*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**CollectionMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have collections.

addCollection(*collections*, *locked=True*)

Add a collection tag(s).

Parameters

- **collections** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeCollection(*collections*, *locked=True*)

Remove a collection tag(s).

Parameters

- **collections** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**ContentRatingMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a content rating.

editContentRating(*contentRating*, *locked=True*)

Edit the content rating.

Parameters

- **contentRating** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**CountryMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have countries.

addCountry(*countries*, *locked=True*)

Add a country tag(s).

Parameters

- **countries** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeCountry(*countries*, *locked=True*)

Remove a country tag(s).

Parameters

- **countries** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**CriticRatingMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a critic rating.

editCriticRating(*criticRating*, *locked=True*)

Edit the critic rating.

Parameters

- **criticRating** (*float*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**DirectorMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have directors.

addDirector(*directors*, *locked=True*)

Add a director tag(s).

Parameters

- **directors** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeDirector(*directors*, *locked=True*)

Remove a director tag(s).

Parameters

- **directors** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**EditionTitleMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have an edition title.

editEditionTitle(*editionTitle*, *locked=True*)

Edit the edition title. Plex Pass is required to edit this field.

Parameters

- **editionTitle** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**EditFieldMixin**

Bases: object

Mixin for editing Plex object fields.

editField(*field*, *value*, *locked=True*, ***kwargs*)

Edit the field of a Plex object. All field editing methods can be chained together. Also see *batchEdits()* for batch editing fields.

Parameters

- **field** (*str*) – The name of the field to edit.
- **value** (*str*) – The value to edit the field to.

- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

Example

```
# Chaining multiple field edits with reloading
Movie.editTitle('A New Title').editSummary('A new summary').editTagline('A new␣
↪tagline').reload()
```

class plexapi.mixins.**EditTagsMixin**

Bases: `object`

Mixin for editing Plex object tags.

editTags(*tag*, *items*, *locked=True*, *remove=False*, ***kwargs*)

Edit the tags of a Plex object. All tag editing methods can be chained together. Also see *batchEdits()* for batch editing tags.

Parameters

- **tag** (*str*) – Name of the tag to edit.
- **items** (List<str> or List<*MediaTag*>) – List of tags to add or remove.
- **locked** (*bool*) – True (default) to lock the tags, False to unlock the tags.
- **remove** (*bool*) – True to remove the tags in items.

Example

```
# Chaining multiple tag edits with reloading
Show.addCollection('New Collection').removeGenre('Action').addLabel('Favorite').
↪reload()
```

class plexapi.mixins.**GenreMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have genres.

addGenre(*genres*, *locked=True*)

Add a genre tag(s).

Parameters

- **genres** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeGenre(*genres*, *locked=True*)

Remove a genre tag(s).

Parameters

- **genres** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**LabelMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have labels.

addLabel(*labels*, *locked=True*)

Add a label tag(s).

Parameters

- **labels** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeLabel(*labels*, *locked=True*)

Remove a label tag(s).

Parameters

- **labels** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**MoodMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have moods.

addMood(*moods*, *locked=True*)

Add a mood tag(s).

Parameters

- **moods** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeMood(*moods*, *locked=True*)

Remove a mood tag(s).

Parameters

- **moods** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**OriginallyAvailableMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have an originally available date.

editOriginallyAvailable(*originallyAvailable*, *locked=True*)

Edit the originally available date.

Parameters

- **originallyAvailable** (*str or datetime*) – The new value “YYYY-MM-DD (str) or datetime object.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**OriginalTitleMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have an original title.

editOriginalTitle(*originalTitle*, *locked=True*)

Edit the original title.

Parameters

- **originalTitle** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**PhotoCapturedTimeMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a captured time.

editCapturedTime(*capturedTime*, *locked=True*)

Edit the photo captured time.

Parameters

- **capturedTime** (*str or datetime*) – The new value "YYYY-MM-DD hh:mm:ss" (str) or datetime object.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**ProducerMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have producers.

addProducer(*producers*, *locked=True*)

Add a producer tag(s).

Parameters

- **producers** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeProducer(*producers*, *locked=True*)

Remove a producer tag(s).

Parameters

- **producers** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**SimilarArtistMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have similar artists.

addSimilarArtist(*artists*, *locked=True*)

Add a similar artist tag(s).

Parameters

- **artists** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeSimilarArtist(*artists*, *locked=True*)

Remove a similar artist tag(s).

Parameters

- **artists** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**SortTitleMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a sort title.

editSortTitle(*sortTitle*, *locked=True*)

Edit the sort title.

Parameters

- **sortTitle** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**StudioMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a studio.

editStudio(*studio*, *locked=True*)

Edit the studio.

Parameters

- **studio** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**StyleMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have styles.

addStyle(*styles*, *locked=True*)

Add a style tag(s).

Parameters

- **styles** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeStyle(*styles*, *locked=True*)

Remove a style tag(s).

Parameters

- **styles** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**SummaryMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a summary.

editSummary(*summary*, *locked=True*)

Edit the summary.

Parameters

- **summary** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**TaglineMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a tagline.

editTagline(*tagline*, *locked=True*)

Edit the tagline.

Parameters

- **tagline** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**TagMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have tags.

addTag(*tags*, *locked=True*)

Add a tag(s).

Parameters

- **tags** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeTag(*tags*, *locked=True*)

Remove a tag(s).

Parameters

- **tags** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**TitleMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a title.

editTitle(*title*, *locked=True*)

Edit the title.

Parameters

- **title** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**TrackArtistMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a track artist.

editTrackArtist(*trackArtist*, *locked=True*)

Edit the track artist.

Parameters

- **trackArtist** (*str*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**TrackDiscNumberMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a track disc number.

editDiscNumber(*discNumber*, *locked=True*)

Edit the track disc number.

Parameters

- **discNumber** (*int*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**TrackNumberMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a track number.

editTrackNumber(*trackNumber*, *locked=True*)

Edit the track number.

Parameters

- **trackNumber** (*int*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**UserRatingMixin**

Bases: *EditFieldMixin*

Mixin for Plex objects that can have a user rating.

editUserRating(*userRating*, *locked=True*)

Edit the user rating.

Parameters

- **userRating** (*float*) – The new value.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**WriterMixin**

Bases: *EditTagsMixin*

Mixin for Plex objects that can have writers.

addWriter(*writers*, *locked=True*)

Add a writer tag(s).

Parameters

- **writers** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

removeWriter(*writers*, *locked=True*)

Remove a writer tag(s).

Parameters

- **writers** (List<str> or List<*MediaTag*>) – List of tags.
- **locked** (*bool*) – True (default) to lock the field, False to unlock the field.

class plexapi.mixins.**ExtrasMixin**

Bases: object

Mixin for Plex objects that can have extras.

extras()

Returns a list of *Extra* objects.

class plexapi.mixins.**HubsMixin**

Bases: object

Mixin for Plex objects that can have related hubs.

hubs()

Returns a list of *Hub* objects.

class plexapi.mixins.**PlayedUnplayedMixin**

Bases: object

Mixin for Plex objects that can be marked played and unplayed.

property isPlayed

Returns True if this video is played.

markPlayed()

Mark the Plex object as played.

markUnplayed()

Mark the Plex object as unplayed.

property isWatched

Alias to self.isPlayed.

markWatched()

Alias to *markPlayed()*.

markUnwatched()

Alias to *markUnplayed()*.

class plexapi.mixins.**RatingMixin**

Bases: object

Mixin for Plex objects that can have user star ratings.

rate(*rating=None*)

Rate the Plex object. Note: Plex ratings are displayed out of 5 stars (e.g. rating 7.0 = 3.5 stars).

Parameters
rating (*float, optional*) – Rating from 0 to 10. Exclude to reset the rating.

Raises
BadRequest – If the rating is invalid.

class plexapi.mixins.**ArtLockMixin**

Bases: object

Mixin for Plex objects that can have a locked background artwork.

lockArt()

Lock the background artwork for a Plex object.

unlockArt()

Unlock the background artwork for a Plex object.

class plexapi.mixins.**ArtMixin**

Bases: *ArtUrlMixin*, *ArtLockMixin*

Mixin for Plex objects that can have background artwork.

arts()

Returns list of available *Art* objects.

uploadArt(*url=None*, *filepath=None*)

Upload a background artwork from a url or filepath.

Parameters

- **url** (*str*) – The full URL to the image to upload.
- **filepath** (*str*) – The full file path to the image to upload or file-like object.

setArt(*art*)

Set the background artwork for a Plex object.

Parameters

art (*Art*) – The art object to select.

deleteArt()

Delete the art from a Plex object.

class plexapi.mixins.**ArtUrlMixin**

Bases: `object`

Mixin for Plex objects that can have a background artwork url.

property artUrl

Return the art url for the Plex object.

class plexapi.mixins.**LogoLockMixin**

Bases: `object`

Mixin for Plex objects that can have a locked logo.

lockLogo()

Lock the logo for a Plex object.

unlockLogo()

Unlock the logo for a Plex object.

class plexapi.mixins.**LogoMixin**

Bases: *LogoUrlMixin*, *LogoLockMixin*

Mixin for Plex objects that can have logos.

logos()

Returns list of available *Logo* objects.

uploadLogo(*url=None*, *filepath=None*)

Upload a logo from a url or filepath.

Parameters

- **url** (*str*) – The full URL to the image to upload.

- **filepath** (*str*) – The full file path to the image to upload or file-like object.

setLogo(*logo*)

Set the logo for a Plex object.

Parameters
logo (*Logo*) – The logo object to select.

deleteLogo()

Delete the logo from a Plex object.

class plexapi.mixins.**LogoUrlMixin**

Bases: object

Mixin for Plex objects that can have a logo url.

property logo

Return the API path to the logo image.

property logoUrl

Return the logo url for the Plex object.

class plexapi.mixins.**PosterLockMixin**

Bases: object

Mixin for Plex objects that can have a locked poster.

lockPoster()

Lock the poster for a Plex object.

unlockPoster()

Unlock the poster for a Plex object.

class plexapi.mixins.**PosterMixin**

Bases: *PosterUrlMixin*, *PosterLockMixin*

Mixin for Plex objects that can have posters.

posters()

Returns list of available *Poster* objects.

uploadPoster(*url=None*, *filepath=None*)

Upload a poster from a url or filepath.

Parameters

- **url** (*str*) – The full URL to the image to upload.
- **filepath** (*str*) – The full file path to the image to upload or file-like object.

setPoster(*poster*)

Set the poster for a Plex object.

Parameters
poster (*Poster*) – The poster object to select.

deletePoster()

Delete the poster from a Plex object.

class plexapi.mixins.**PosterUrlMixin**

Bases: object

Mixin for Plex objects that can have a poster url.

property thumbUrl

Return the thumb url for the Plex object.

property posterUrl

Alias to self.thumbUrl.

class plexapi.mixins.**SquareArtLockMixin**

Bases: object

Mixin for Plex objects that can have a locked square art.

lockSquareArt()

Lock the square art for a Plex object.

unlockSquareArt()

Unlock the square art for a Plex object.

class plexapi.mixins.**SquareArtMixin**

Bases: *SquareArtUrlMixin*, *SquareArtLockMixin*

Mixin for Plex objects that can have square art.

squareArts()

Returns list of available *SquareArt* objects.

uploadSquareArt(*url=None*, *filepath=None*)

Upload a square art from a url or filepath.

Parameters

- **url** (*str*) – The full URL to the image to upload.
- **filepath** (*str*) – The full file path to the image to upload or file-like object.

setSquareArt(*squareArt*)

Set the square art for a Plex object.

Parameters

squareArt (*SquareArt*) – The square art object to select.

deleteSquareArt()

Delete the square art from a Plex object.

class plexapi.mixins.**SquareArtUrlMixin**

Bases: object

Mixin for Plex objects that can have a square art url.

property squareArt

Return the API path to the square art image.

property squareArtUrl

Return the square art url for the Plex object.

class plexapi.mixins.**ThemeLockMixin**

Bases: object

Mixin for Plex objects that can have a locked theme.

lockTheme()

Lock the theme for a Plex object.

unlockTheme()

Unlock the theme for a Plex object.

class plexapi.mixins.**ThemeMixin**

Bases: *ThemeUrlMixin*, *ThemeLockMixin*

Mixin for Plex objects that can have themes.

themes()

Returns list of available *Theme* objects.

uploadTheme(*url=None*, *filepath=None*, *timeout=None*)

Upload a theme from url or filepath.

Warning: Themes cannot be deleted using PlexAPI!

Parameters

- **url** (*str*) – The full URL to the theme to upload.
- **filepath** (*str*) – The full file path to the theme to upload or file-like object.
- **timeout** (*int, optional*) – Timeout, in seconds, to use when uploading themes to the server. (default config.TIMEOUT).

setTheme(*theme*)

Set the theme for a Plex object.

Raises

NotImplementedError – Themes cannot be set through the API.

deleteTheme()

Delete the theme from a Plex object.

class plexapi.mixins.**ThemeUrlMixin**

Bases: object

Mixin for Plex objects that can have a theme url.

property themeUrl

Return the theme url for the Plex object.

class plexapi.mixins.**SmartFilterMixin**

Bases: object

Mixin for Plex objects that can have smart filters.

class plexapi.mixins.**SplitMergeMixin**

Bases: object

Mixin for Plex objects that can be split and merged.

split()

Split duplicated Plex object into separate objects.

merge(*ratingKeys*)

Merge other Plex objects into the current object.

Parameters
ratingKeys (*list*) – A list of rating keys to merge.

class plexapi.mixins.**UnmatchMatchMixin**

Bases: object

Mixin for Plex objects that can be unmatched and matched.

unmatch()

Unmatches metadata match from object.

matches(*agent=None*, *title=None*, *year=None*, *language=None*)

Return list of (*SearchResult*) metadata matches.

Parameters:
agent (str): Agent name to be used (imdb, thetvdb, themoviedb, etc.) title (str): Title of item to search for year (str): Year of item to search in language (str) : Language of item to search in

Examples

1. video.matches()
2. video.matches(title="something", year=2020)
3. video.matches(title="something")
4. video.matches(year=2020)
5. video.matches(title="something", year="")
6. video.matches(title="", year=2020)
7. video.matches(title="", year="")

1. The default behaviour in Plex Web = no params in plexapi
2. Both title and year specified by user
3. Year automatically filled in
4. Title automatically filled in
5. Explicitly searches for title with blank year
6. Explicitly searches for blank title with year
7. I don't know what the user is thinking... return the same result as 1

For 2 to 7, the agent and language is automatically filled in

fixMatch(*searchResult=None*, *auto=False*, *agent=None*)

Use match result to update show metadata.

Parameters

- **auto** (*bool*) – True uses first match from matches False allows user to provide the match
- **searchResult** (*SearchResult*) – Search result from ~plexapi.base.matches()
- **agent** (*str*) – Agent name to be used (imdb, thetvdb, themoviedb, etc.)

class plexapi.mixins.**WatchlistMixin**

Bases: object

Mixin for Plex objects that can be added to a user's watchlist.

onWatchlist(*account=None*)

Returns True if the item is on the user's watchlist. Also see *onWatchlist()*.

Parameters

account (*MyPlexAccount*, optional) – Account to check item on the watchlist. Note: This is required if you are not connected to a Plex server instance using the admin account.

addToWatchlist(*account=None*)

Add this item to the specified user's watchlist. Also see *addToWatchlist()*.

Parameters

account (*MyPlexAccount*, optional) – Account to add item to the watchlist. Note: This is required if you are not connected to a Plex server instance using the admin account.

removeFromWatchlist(*account=None*)

Remove this item from the specified user's watchlist. Also see *removeFromWatchlist()*.

Parameters

account (*MyPlexAccount*, optional) – Account to remove item from the watchlist. Note: This is required if you are not connected to a Plex server instance using the admin account.

streamingServices(*account=None*)

Return a list of *Availability* objects for the available streaming services for this item.

Parameters

account (*MyPlexAccount*, optional) – Account used to retrieve availability. Note: This is required if you are not connected to a Plex server instance using the admin account.

class plexapi.mixins.**AlbumEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *CriticRatingMixin*, *OriginallyAvailableMixin*, *SortTitleMixin*, *StudioMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*, *CollectionMixin*, *GenreMixin*, *LabelMixin*, *MoodMixin*, *StyleMixin*

class plexapi.mixins.**ArtistEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *CriticRatingMixin*, *SortTitleMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*, *CollectionMixin*, *CountryMixin*, *GenreMixin*, *LabelMixin*, *MoodMixin*, *SimilarArtistMixin*, *StyleMixin*

class plexapi.mixins.**CollectionEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *ContentRatingMixin*, *CriticRatingMixin*, *SortTitleMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*, *LabelMixin*

class plexapi.mixins.**EpisodeEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *ContentRatingMixin*, *CriticRatingMixin*, *OriginallyAvailableMixin*, *SortTitleMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*, *CollectionMixin*, *DirectorMixin*, *LabelMixin*, *WriterMixin*

class plexapi.mixins.**MovieEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *ContentRatingMixin*, *CriticRatingMixin*, *EditionTitleMixin*, *OriginallyAvailableMixin*,

OriginalTitleMixin, *SortTitleMixin*, *StudioMixin*, *SummaryMixin*, *TaglineMixin*, *TitleMixin*, *UserRatingMixin*, *CollectionMixin*, *CountryMixin*, *DirectorMixin*, *GenreMixin*, *LabelMixin*, *ProducerMixin*, *WriterMixin*

class plexapi.mixins.**PhotoEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *AddedAtMixin*, *PhotoCapturedTimeMixin*, *SortTitleMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*, *TagMixin*

class plexapi.mixins.**PhotoalbumEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *AddedAtMixin*, *SortTitleMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*

class plexapi.mixins.**PlaylistEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *SortTitleMixin*, *SummaryMixin*, *TitleMixin*

class plexapi.mixins.**SeasonEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *CriticRatingMixin*, *SummaryMixin*, *TitleMixin*, *UserRatingMixin*, *CollectionMixin*, *LabelMixin*

class plexapi.mixins.**ShowEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *ContentRatingMixin*, *CriticRatingMixin*, *OriginallyAvailableMixin*, *OriginalTitleMixin*, *SortTitleMixin*, *StudioMixin*, *SummaryMixin*, *TaglineMixin*, *TitleMixin*, *UserRatingMixin*, *CollectionMixin*, *GenreMixin*, *LabelMixin*

class plexapi.mixins.**TrackEditMixins**

Bases: *ArtLockMixin*, *PosterLockMixin*, *ThemeLockMixin*, *AddedAtMixin*, *AudienceRatingMixin*, *CriticRatingMixin*, *TitleMixin*, *TrackArtistMixin*, *TrackNumberMixin*, *TrackDiscNumberMixin*, *UserRatingMixin*, *CollectionMixin*, *GenreMixin*, *LabelMixin*, *MoodMixin*

class plexapi.mixins.**AlbumMixins**

Bases: *SplitMergeMixin*, *UnmatchMatchMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeUrlMixin*, *AlbumEditMixins*

class plexapi.mixins.**ArtistMixins**

Bases: *AdvancedSettingsMixin*, *SplitMergeMixin*, *UnmatchMatchMixin*, *ExtrasMixin*, *HubsMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeMixin*, *ArtistEditMixins*

class plexapi.mixins.**ClipMixins**

Bases: *ArtUrlMixin*, *LogoUrlMixin*, *PosterUrlMixin*, *SquareArtUrlMixin*

class plexapi.mixins.**CollectionMixins**

Bases: *AdvancedSettingsMixin*, *SmartFilterMixin*, *HubsMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeMixin*, *CollectionEditMixins*

class plexapi.mixins.**EpisodeMixins**

Bases: *ExtrasMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeUrlMixin*, *EpisodeEditMixins*

class plexapi.mixins.**MovieMixins**

Bases: *AdvancedSettingsMixin*, *SplitMergeMixin*, *UnmatchMatchMixin*, *ExtrasMixin*, *HubsMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeMixin*, *MovieEditMixins*, *WatchlistMixin*

class plexapi.mixins.**PhotoMixins**

Bases: *RatingMixin*, *ArtUrlMixin*, *LogoUrlMixin*, *PosterUrlMixin*, *SquareArtUrlMixin*, *PhotoEditMixins*

class plexapi.mixins.**PhotoalbumMixins**

Bases: *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *PhotoalbumEditMixins*

class plexapi.mixins.**PlaylistMixins**

Bases: *SmartFilterMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *PlaylistEditMixins*

class plexapi.mixins.**SeasonMixins**

Bases: *AdvancedSettingsMixin*, *ExtrasMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeUrlMixin*, *SeasonEditMixins*

class plexapi.mixins.**ShowMixins**

Bases: *AdvancedSettingsMixin*, *SplitMergeMixin*, *UnmatchMatchMixin*, *ExtrasMixin*, *HubsMixin*, *RatingMixin*, *ArtMixin*, *LogoMixin*, *PosterMixin*, *SquareArtMixin*, *ThemeMixin*, *ShowEditMixins*, *WatchlistMixin*

class plexapi.mixins.**TrackMixins**

Bases: *ExtrasMixin*, *RatingMixin*, *ArtUrlMixin*, *LogoUrlMixin*, *PosterUrlMixin*, *SquareArtUrlMixin*, *ThemeUrlMixin*, *TrackEditMixins*

CHAPTER

FOURTEEN

MYPLEX PLEXAPI.MYPLEX

class plexapi.myplex.**MyPlexAccount**(*username=None*, *password=None*, *token=None*, *session=None*, *timeout=None*, *code=None*, *remember=True*)

Bases: *PlexObject*

MyPlex account and profile information. This object represents the data found Account on the myplex.tv servers at the url https://plex.tv/api/v2/user. You may create this object directly by passing in your username & password (or token). There is also a convenience method provided at *myPlexAccount()* which will create and return this object.

Parameters

- **username** (*str*) – Plex login username if not using a token.
- **password** (*str*) – Plex login password if not using a token.
- **token** (*str*) – Plex authentication token instead of username and password.
- **session** (*requests.Session, optional*) – Use your own session object if you want to cache the http responses from PMS.
- **timeout** (*int*) – timeout in seconds on initial connect to myplex (default config.TIMEOUT).
- **code** (*str*) – Two-factor authentication code to use when logging in with username and password.
- **remember** (*bool*) – Remember the account token for 14 days (Default True).

Variables

- **key** (*str*) – 'https://plex.tv/api/v2/user'
- **adsConsent** (*str*) – Unknown.
- **adsConsentReminderAt** (*str*) – Unknown.
- **adsConsentSetAt** (*str*) – Unknown.
- **anonymous** (*str*) – Unknown.
- **authToken** (*str*) – The account token.
- **backupCodesCreated** (*bool*) – If the two-factor authentication backup codes have been created.
- **confirmed** (*bool*) – If the account has been confirmed.
- **country** (*str*) – The account country.
- **email** (*str*) – The account email address.

- **emailOnlyAuth** (*bool*) – If login with email only is enabled.
- **experimentalFeatures** (*bool*) – If experimental features are enabled.
- **friendlyName** (*str*) – Your account full name.
- **entitlements** (*List<str>*) – List of devices your allowed to use with this account.
- **guest** (*bool*) – If the account is a Plex Home guest user.
- **hasPassword** (*bool*) – If the account has a password.
- **home** (*bool*) – If the account is a Plex Home user.
- **homeAdmin** (*bool*) – If the account is the Plex Home admin.
- **homeSize** (*int*) – The number of accounts in the Plex Home.
- **id** (*int*) – The Plex account ID.
- **joinedAt** (*datetime*) – Date the account joined Plex.
- **locale** (*str*) – the account locale
- **mailingListActive** (*bool*) – If you are subscribed to the Plex newsletter.
- **mailingListStatus** (*str*) – Your current mailing list status.
- **maxHomeSize** (*int*) – The maximum number of accounts allowed in the Plex Home.
- **pin** (*str*) – The hashed Plex Home PIN.
- **profileAutoSelectAudio** (*bool*) – If the account has automatically select audio and subtitle tracks enabled.
- **profileDefaultAudioLanguage** (*str*) – The preferred audio language for the account.
- **profileDefaultSubtitleLanguage** (*str*) – The preferred subtitle language for the account.
- **profileAutoSelectSubtitle** (*int*) – The auto-select subtitle mode (0 = Manually selected, 1 = Shown with foreign audio, 2 = Always enabled).
- **profileDefaultSubtitleAccessibility** (*int*) – The subtitles for the deaf or hard-of-hearing (SDH) searches mode (0 = Prefer non-SDH subtitles, 1 = Prefer SDH subtitles, 2 = Only show SDH subtitles, 3 = Only shown non-SDH subtitles).
- **profileDefaultSubtitleForced** (*int*) – The forced subtitles searches mode (0 = Prefer non-forced subtitles, 1 = Prefer forced subtitles, 2 = Only show forced subtitles, 3 = Only show non-forced subtitles).
- **protected** (*bool*) – If the account has a Plex Home PIN enabled.
- **rememberExpiresAt** (*datetime*) – Date the token expires.
- **restricted** (*bool*) – If the account is a Plex Home managed user.
- **roles** – (List<str>) Lit of account roles. Plexpass membership listed here.
- **scrobbleTypes** (*List<int>*) – Unknown.
- **subscriptionActive** (*bool*) – If the account's Plex Pass subscription is active.
- **subscriptionDescription** (*str*) – Description of the Plex Pass subscription.
- **subscriptionFeatures** – (List<str>) List of features allowed on your Plex Pass subscription.

- **subscriptionPaymentService** (*str*) – Payment service used for your Plex Pass subscription.
- **subscriptionPlan** (*str*) – Name of Plex Pass subscription plan.
- **subscriptionStatus** (*str*) – String representation of `subscriptionActive`.
- **subscriptionSubscribedAt** (*datetime*) – Date the account subscribed to Plex Pass.
- **thumb** (*str*) – URL of the account thumbnail.
- **title** (*str*) – The title of the account (username or friendly name).
- **twoFactorEnabled** (*bool*) – If two-factor authentication is enabled.
- **username** (*str*) – The account username.
- **uuid** (*str*) – The account UUID.

signout()

Sign out of the Plex account. Invalidates the authentication token.

property authenticationToken

Returns the authentication token for the account. Alias for `authToken`.

ping()

Ping the Plex.tv API. This will refresh the authentication token to prevent it from expiring.

device(*name=None*, *clientId=None*)

Returns the *MyPlexDevice* that matches the name specified.

Parameters

- **name** (*str*) – Name to match against.
- **clientId** (*str*) – clientIdentifier to match against.

devices()

Returns a list of all *MyPlexDevice* objects connected to the server.

resource(*name*)

Returns the *MyPlexResource* that matches the name specified.

Parameters

name (*str*) – Name or machine identifier to match against.

resources()

Returns a list of all *MyPlexResource* objects connected to the server.

inviteFriend(*user*, *server*, *sections=None*, *allowSync=False*, *allowCameraUpload=False*, *allowChannels=False*, *filterMovies=None*, *filterTelevision=None*, *filterMusic=None*)

Share library content with the specified user.

Parameters

- **user** (*MyPlexUser*) – *MyPlexUser* object, username, or email of the user to be added.
- **server** (*PlexServer*) – *PlexServer* object, or machineIdentifier containing the library sections to share.
- **sections** (List<*LibrarySection*>) – List of *LibrarySection* objects, or names to be shared (default None). *sections* must be defined in order to update shared libraries.
- **allowSync** (*Bool*) – Set True to allow user to sync content.

- **allowCameraUpload** (*Bool*) – Set True to allow user to upload photos.
- **allowChannels** (*Bool*) – Set True to allow user to utilize installed channels.
- **filterMovies** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterTelevision** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterMusic** (*Dict*) – Dict containing key 'label' set to a list of values to be filtered. ex: *{'label':['foo']}*

createHomeUser(*user*, *server*, *sections=None*, *allowSync=False*, *allowCameraUpload=False*, *allowChannels=False*, *filterMovies=None*, *filterTelevision=None*, *filterMusic=None*)

Share library content with the specified user.

Parameters

- **user** (*MyPlexUser*) – *MyPlexUser* object, username, or email of the user to be added.
- **server** (*PlexServer*) – *PlexServer* object, or machineIdentifier containing the library sections to share.
- **sections** (List<*LibrarySection*>) – List of *LibrarySection* objects, or names to be shared (default None). *sections* must be defined in order to update shared libraries.
- **allowSync** (*Bool*) – Set True to allow user to sync content.
- **allowCameraUpload** (*Bool*) – Set True to allow user to upload photos.
- **allowChannels** (*Bool*) – Set True to allow user to utilize installed channels.
- **filterMovies** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterTelevision** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterMusic** (*Dict*) – Dict containing key 'label' set to a list of values to be filtered. ex: *{'label':['foo']}*

createExistingUser(*user*, *server*, *sections=None*, *allowSync=False*, *allowCameraUpload=False*, *allowChannels=False*, *filterMovies=None*, *filterTelevision=None*, *filterMusic=None*)

Share library content with the specified user.

Parameters

- **user** (*MyPlexUser*) – *MyPlexUser* object, username, or email of the user to be added.
- **server** (*PlexServer*) – *PlexServer* object, or machineIdentifier containing the library sections to share.
- **sections** (List<*LibrarySection*>) – List of *LibrarySection* objects, or names to be shared (default None). *sections* must be defined in order to update shared libraries.
- **allowSync** (*Bool*) – Set True to allow user to sync content.
- **allowCameraUpload** (*Bool*) – Set True to allow user to upload photos.
- **allowChannels** (*Bool*) – Set True to allow user to utilize installed channels.
- **filterMovies** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*

- **filterTelevision** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterMusic** (*Dict*) – Dict containing key 'label' set to a list of values to be filtered. ex: *{'label':['foo']}*

removeFriend(*user*)

Remove the specified user from your friends.

Parameters

user (*MyPlexUser* or str) – *MyPlexUser*, username, or email of the user to be removed.

removeHomeUser(*user*)

Remove the specified user from your home users.

Parameters

user (*MyPlexUser* or str) – *MyPlexUser*, username, or email of the user to be removed.

switchHomeUser(*user*, *pin=None*)

Returns a new *MyPlexAccount* object switched to the given home user.

Parameters

- **user** (*MyPlexUser* or str) – *MyPlexUser*, username, or email of the home user to switch to.
- **pin** (*str*) – PIN for the home user (required if the home user has a PIN set).

Example

```
from plexapi.myplex import MyPlexAccount
# Login to a Plex Home account
account = MyPlexAccount('<USERNAME>', '<PASSWORD>')
# Switch to a different Plex Home user
userAccount = account.switchHomeUser('Username')
```

setPin(*newPin*, *currentPin=None*)

Set a new Plex Home PIN for the account.

Parameters

- **newPin** (*str*) – New PIN to set for the account.
- **currentPin** (*str*) – Current PIN for the account (required to change the PIN).

removePin(*currentPin*)

Remove the Plex Home PIN for the account.

Parameters

currentPin (*str*) – Current PIN for the account (required to remove the PIN).

setManagedUserPin(*user*, *newPin*)

Set a new Plex Home PIN for a managed home user. This must be done from the Plex Home admin account.

Parameters

- **user** (*MyPlexUser* or str) – *MyPlexUser* or username of the managed home user.
- **newPin** (*str*) – New PIN to set for the managed home user.

removeManagedUserPin(*user*)

Remove the Plex Home PIN for a managed home user. This must be done from the Plex Home admin account.

Parameters
user (*MyPlexUser* or str) – *MyPlexUser* or username of the managed home user.

acceptInvite(*user*)

Accept a pending friend invite from the specified user.

Parameters
user (*MyPlexInvite* or str) – *MyPlexInvite*, username, or email of the friend invite to accept.

cancelInvite(*user*)

Cancel a pending firend invite for the specified user.

Parameters
user (*MyPlexInvite* or str) – *MyPlexInvite*, username, or email of the friend invite to cancel.

updateFriend(*user*, *server*, *sections=None*, *removeSections=False*, *allowSync=None*, *allowCameraUpload=None*, *allowChannels=None*, *filterMovies=None*, *filterTelevision=None*, *filterMusic=None*)

Update the specified user's share settings.

Parameters

- **user** (*MyPlexUser*) – *MyPlexUser* object, username, or email of the user to be updated.
- **server** (*PlexServer*) – *PlexServer* object, or machineIdentifier containing the library sections to share.
- **sections** (List<*LibrarySection*>) – List of *LibrarySection* objects, or names to be shared (default None). *sections* must be defined in order to update shared libraries.
- **removeSections** (*Bool*) – Set True to remove all shares. Supersedes sections.
- **allowSync** (*Bool*) – Set True to allow user to sync content.
- **allowCameraUpload** (*Bool*) – Set True to allow user to upload photos.
- **allowChannels** (*Bool*) – Set True to allow user to utilize installed channels.
- **filterMovies** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterTelevision** (*Dict*) – Dict containing key 'contentRating' and/or 'label' each set to a list of values to be filtered. ex: *{'contentRating':['G'], 'label':['foo']}*
- **filterMusic** (*Dict*) – Dict containing key 'label' set to a list of values to be filtered. ex: *{'label':['foo']}*

user(*username*)

Returns the *MyPlexUser* that matches the specified username or email.

Parameters
username (*str*) – Username, email or id of the user to return.

users()

Returns a list of all *MyPlexUser* objects connected to your account.

pendingInvite(*username*, *includeSent=True*, *includeReceived=True*)

Returns the *MyPlexInvite* that matches the specified username or email. Note: This can be a pending invite sent from your account or received to your account.

Parameters

- **username** (*str*) – Username, email or id of the user to return.
- **includeSent** (*bool*) – True to include sent invites.
- **includeReceived** (*bool*) – True to include received invites.

pendingInvites(*includeSent=True*, *includeReceived=True*)

Returns a list of all *MyPlexInvite* objects connected to your account. Note: This includes all pending invites sent from your account and received to your account.

Parameters

- **includeSent** (*bool*) – True to include sent invites.
- **includeReceived** (*bool*) – True to include received invites.

optOut(*playback=None*, *library=None*)

Opt in or out of sharing stuff with plex. See: https://www.plex.tv/about/privacy-legal/

syncItems(*client=None*, *clientId=None*)

Returns an instance of *SyncList* for specified client.

Parameters

- **client** (*MyPlexDevice*) – a client to query SyncItems for.
- **clientId** (*str*) – an identifier of a client to query SyncItems for.

If both *client* and *clientId* provided the client would be preferred. If neither *client* nor *clientId* provided the clientId would be set to current clients's identifier.

sync(*sync_item*, *client=None*, *clientId=None*)

Adds specified sync item for the client. It's always easier to use methods defined directly in the media objects, e.g. *sync()*, *sync()*.

Parameters

- **client** (*MyPlexDevice*) – a client for which you need to add SyncItem to.
- **clientId** (*str*) – an identifier of a client for which you need to add SyncItem to.
- **sync_item** (*SyncItem*) – prepared SyncItem object with all fields set.

If both *client* and *clientId* provided the client would be preferred. If neither *client* nor *clientId* provided the clientId would be set to current clients's identifier.

Returns
an instance of created syncItem.

Return type
SyncItem

Raises

- ***BadRequest*** – When client with provided clientId wasn't found.
- ***BadRequest*** – Provided client doesn't provides *sync-target*.

claimToken()

Returns a str, a new "claim-token", which you can use to register your new Plex Server instance to your account. See: https://hub.docker.com/r/plexinc/pms-docker/, https://www.plex.tv/claim/

history(*maxresults=None*, *mindate=None*)

Get Play History for all library sections on all servers for the owner.

Parameters

- **maxresults** (*int*) – Only return the specified number of results (optional).
- **mindate** (*datetime*) – Min datetime to return results from.

onlineMediaSources()

Returns a list of user account Online Media Sources settings *AccountOptOut*

videoOnDemand()

Returns a list of VOD Hub items *Hub*

tidal()

Returns a list of tidal Hub items *Hub*

watchlist(*filter=None*, *sort=None*, *libtype=None*, *maxresults=None*, ***kwargs*)

Returns a list of *Movie* and *Show* items in the user's watchlist. Note: The objects returned are from Plex's online metadata. To get the matching item on a Plex server, search for the media using the guid.

Parameters

- **filter** (*str, optional*) – 'available' or 'released' to only return items that are available or released, otherwise return all items.
- **sort** (*str, optional*) – In the format `field:dir`. Available fields are `watchlistedAt` (Added At), `titleSort` (Title), `originallyAvailableAt` (Release Date), or `rating` (Critic Rating). `dir` can be `asc` or `desc`.
- **libtype** (*str, optional*) – 'movie' or 'show' to only return movies or shows, otherwise return all items.
- **maxresults** (*int, optional*) – Only return the specified number of results.
- ****kwargs** (*dict*) – Additional custom filters to apply to the search results.

Example

```
# Watchlist for released movies sorted by critic rating in descending order
watchlist = account.watchlist(filter='released', sort='rating:desc', libtype=
↪'movie')
item = watchlist[0]  # First item in the watchlist

# Search for the item on a Plex server
result = plex.library.search(guid=item.guid, libtype=item.type)
```

onWatchlist(*item*)

Returns True if the item is on the user's watchlist.

Parameters

item (*Movie* or *Show*) – Item to check if it is on the user's watchlist.

addToWatchlist(*items*)

Add media items to the user's watchlist

Parameters
items (*List*) – List of *Movie* or *Show* objects to be added to the watchlist.

Raises
BadRequest – When trying to add invalid or existing media to the watchlist.

removeFromWatchlist(*items*)

Remove media items from the user's watchlist

Parameters
items (*List*) – List of *Movie* or *Show* objects to be added to the watchlist.

Raises
BadRequest – When trying to remove invalid or non-existing media to the watchlist.

userState(*item*)

Returns a *UserState* object for the specified item.

Parameters
item (*Movie* or *Show*) – Item to return the user state.

isPlayed(*item*)

Return True if the item is played on Discover.

:param item (*Movie*: :param : :param *Show*: :param *Season* or: :param *Episode*): Object from searchDiscover(). :param Can be also result from Plex Movie or Plex TV Series agent.:

markPlayed(*item*)

Mark the Plex object as played on Discover.

:param item (*Movie*: :param : :param *Show*: :param *Season* or: :param *Episode*): Object from searchDiscover(). :param Can be also result from Plex Movie or Plex TV Series agent.:

markUnplayed(*item*)

Mark the Plex object as unplayed on Discover.

:param item (*Movie*: :param : :param *Show*: :param *Season* or: :param *Episode*): Object from searchDiscover(). :param Can be also result from Plex Movie or Plex TV Series agent.:

searchDiscover(*query*, *limit=30*, *libtype=None*, *providers='discover'*)

Search for movies and TV shows in Discover. Returns a list of *Movie* and *Show* objects.

Parameters

- **query** (*str*) – Search query.
- **limit** (*int, optional*) – Limit to the specified number of results. Default 30.
- **libtype** (*str, optional*) – 'movie' or 'show' to only return movies or shows, otherwise return all items.
- **providers** (*str, optional*) – 'discover' for default behavior or 'discover,PLEXAVOD' to also include the Plex ad-suported video service or 'discover,PLEXAVOD,PLEXTVOD' to also include the Plex video rental service

property viewStateSync

Returns True or False if syncing of watch state and ratings is enabled or disabled, respectively, for the account.

enableViewStateSync()

Enable syncing of watch state and ratings for the account.

disableViewStateSync()

Disable syncing of watch state and ratings for the account.

link(*pin*)

Link a device to the account using a pin code.

Parameters
pin (*str*) – The 4 digit link pin code.

publicIP()

Returns your public IP address.

geoip(*ip_address*)

Returns a *GeoLocation* object with geolocation information for an IP address using Plex's GeoIP database.

Parameters
ip_address (*str*) – IP address to lookup.

class plexapi.myplex.**MyPlexUser**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

This object represents non-signed in users such as friends and linked accounts. NOTE: This should not be confused with the *MyPlexAccount* which is your specific account. The raw xml for the data presented here can be found at: https://plex.tv/api/users/

Variables

- **TAG** (*str*) – 'User'
- **key** (*str*) – 'https://plex.tv/api/users/'
- **allowCameraUpload** (*bool*) – True if this user can upload images.
- **allowChannels** (*bool*) – True if this user has access to channels.
- **allowSync** (*bool*) – True if this user can sync.
- **email** (*str*) – User's email address (user@gmail.com).
- **filterAll** (*str*) – Unknown.
- **filterMovies** (*str*) – Unknown.
- **filterMusic** (*str*) – Unknown.
- **filterPhotos** (*str*) – Unknown.
- **filterTelevision** (*str*) – Unknown.
- **home** (*bool*) – Unknown.
- **id** (*int*) – User's Plex account ID.
- **protected** (*False*) – Unknown (possibly SSL enabled?).
- **recommendationsPlaylistId** (*str*) – Unknown.
- **restricted** (*str*) – Unknown.
- **servers** (List<<MyPlexServerShare>)) – Servers shared with the user.
- **thumb** (*str*) – Link to the users avatar.
- **title** (*str*) – Seems to be an alias for username.

- **username** (*str*) – User's username.

server(*name*)

Returns the *MyPlexServerShare* that matches the name specified.

Parameters
name (*str*) – Name of the server to return.

history(*maxresults=None*, *mindate=None*)

Get all Play History for a user in all shared servers. :param maxresults: Only return the specified number of results (optional). :type maxresults: int :param mindate: Min datetime to return results from. :type mindate: datetime

class plexapi.myplex.**MyPlexInvite**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

This object represents pending friend invites.

Variables

- **TAG** (*str*) – 'Invite'
- **createdAt** (*datetime*) – Datetime the user was invited.
- **email** (*str*) – User's email address (user@gmail.com).
- **friend** (*bool*) – True or False if the user is invited as a friend.
- **friendlyName** (*str*) – The user's friendly name.
- **home** (*bool*) – True or False if the user is invited to a Plex Home.
- **id** (*int*) – User's Plex account ID.
- **server** (*bool*) – True or False if the user is invited to any servers.
- **servers** (List<<MyPlexServerShare>)) – Servers shared with the user.
- **thumb** (*str*) – Link to the users avatar.
- **username** (*str*) – User's username.

class plexapi.myplex.**Section**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

This refers to a shared section. The raw xml for the data presented here can be found at: https://plex.tv/api/servers/{machineId}/shared_servers

Variables

- **TAG** (*str*) – section
- **id** (*int*) – The shared section ID
- **key** (*int*) – The shared library section key
- **shared** (*bool*) – If this section is shared with the user
- **title** (*str*) – Title of the section
- **type** (*str*) – movie, tvshow, artist

history(*maxresults=None*, *mindate=None*)

Get all Play History for a user for this section in this shared server. :param maxresults: Only return the specified number of results (optional). :type maxresults: int :param mindate: Min datetime to return results from. :type mindate: datetime

class plexapi.myplex.**MyPlexServerShare**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single user's server reference. Used for library sharing.

Variables

- **id** (*int*) – id for this share
- **serverId** (*str*) – what id plex uses for this.
- **machineIdentifier** (*str*) – The servers machineIdentifier
- **name** (*str*) – The servers name
- **lastSeenAt** (*datetime*) – Last connected to the server?
- **numLibraries** (*int*) – Total number of libraries
- **allLibraries** (*bool*) – True if all libraries is shared with this user.
- **owned** (*bool*) – 1 if the server is owned by the user
- **pending** (*bool*) – True if the invite is pending.

section(*name*)

Returns the *Section* that matches the name specified.

Parameters

name (*str*) – Name of the section to return.

sections()

Returns a list of all *Section* objects shared with this user.

history(*maxresults=9999999*, *mindate=None*)

Get all Play History for a user in this shared server. :param maxresults: Only return the specified number of results (optional). :type maxresults: int :param mindate: Min datetime to return results from. :type mindate: datetime

class plexapi.myplex.**MyPlexResource**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

This object represents resources connected to your Plex server that can provide content such as Plex Media Servers, iPhone or Android clients, etc. The raw xml for the data presented here can be found at: https://plex.tv/api/v2/resources?includeHttps=1&includeRelay=1

Variables

- **TAG** (*str*) – 'Device'
- **key** (*str*) – 'https://plex.tv/api/v2/resources?includeHttps=1&includeRelay=1'
- **accessToken** (*str*) – This resource's Plex access token.
- **clientIdentifier** (*str*) – Unique ID for this resource.
- **connections** (*list*) – List of *ResourceConnection* objects for this resource.
- **createdAt** (*datetime*) – Timestamp this resource first connected to your server.
- **device** (*str*) – Best guess on the type of device this is (PS, iPhone, Linux, etc).
- **dnsRebindingProtection** (*bool*) – True if the server had DNS rebinding protection.
- **home** (*bool*) – Unknown
- **httpsRequired** (*bool*) – True if the resource requires https.

- **lastSeenAt** (*datetime*) – Timestamp this resource last connected.
- **name** (*str*) – Descriptive name of this resource.
- **natLoopbackSupported** (*bool*) – True if the resource supports NAT loopback.
- **owned** (*bool*) – True if this resource is one of your own (you logged into it).
- **ownerId** (*int*) – ID of the user that owns this resource (shared resources only).
- **platform** (*str*) – OS the resource is running (Linux, Windows, Chrome, etc.)
- **platformVersion** (*str*) – Version of the platform.
- **presence** (*bool*) – True if the resource is online
- **product** (*str*) – Plex product (Plex Media Server, Plex for iOS, Plex Web, etc.)
- **productVersion** (*str*) – Version of the product.
- **provides** (*str*) – List of services this resource provides (client, server, player, pubsub-player, etc.)
- **publicAddressMatches** (*bool*) – True if the public IP address matches the client's public IP address.
- **relay** (*bool*) – True if this resource has the Plex Relay enabled.
- **sourceTitle** (*str*) – Username of the user that owns this resource (shared resources only).
- **synced** (*bool*) – Unknown (possibly True if the resource has synced content?)

preferred_connections(*ssl=None*, *locations=None*, *schemes=None*)

Returns a sorted list of the available connection addresses for this resource. Often times there is more than one address specified for a server or client. Default behavior will prioritize local connections before remote or relay and HTTPS before HTTP.

Parameters

ssl (*bool, optional*) – Set True to only connect to HTTPS connections. Set False to only connect to HTTP connections. Set None (default) to connect to any HTTP or HTTPS connection.

connect(*ssl=None*, *timeout=None*, *locations=None*, *schemes=None*)

Returns a new *PlexServer* or *PlexClient* object. Uses *MyPlexResource.preferred_connections()* to generate the priority order of connection addresses. After trying to connect to all available addresses for this resource and assuming at least one connection was successful, the PlexServer object is built and returned.

Parameters

- **ssl** (*bool, optional*) – Set True to only connect to HTTPS connections. Set False to only connect to HTTP connections. Set None (default) to connect to any HTTP or HTTPS connection.
- **timeout** (*int, optional*) – The timeout in seconds to attempt each connection.

Raises

NotFound – When unable to connect to any addresses for this resource.

class plexapi.myplex.**ResourceConnection**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Resource Connection object found within the *MyPlexResource* objects.

Variables

- **TAG** (*str*) – 'Connection'

- **address** (*str*) – The connection IP address
- **httpuri** (*str*) – Full HTTP URL
- **ipv6** (*bool*) – True if the address is IPv6
- **local** (*bool*) – True if the address is local
- **port** (*int*) – The connection port
- **protocol** (*str*) – HTTP or HTTPS
- **relay** (*bool*) – True if the address uses the Plex Relay
- **uri** (*str*) – Full connetion URL

class plexapi.myplex.**MyPlexDevice**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

This object represents resources connected to your Plex server that provide playback ability from your Plex Server, iPhone or Android clients, Plex Web, this API, etc. The raw xml for the data presented here can be found at: https://plex.tv/devices.xml

Variables

- **TAG** (*str*) – 'Device'
- **key** (*str*) – 'https://plex.tv/devices.xml'
- **clientIdentifier** (*str*) – Unique ID for this resource.
- **connections** (*list*) – List of connection URIs for the device.
- **device** (*str*) – Best guess on the type of device this is (Linux, iPad, AFTB, etc).
- **id** (*str*) – MyPlex ID of the device.
- **model** (*str*) – Model of the device (bueller, Linux, x86_64, etc.)
- **name** (*str*) – Hostname of the device.
- **platform** (*str*) – OS the resource is running (Linux, Windows, Chrome, etc.)
- **platformVersion** (*str*) – Version of the platform.
- **product** (*str*) – Plex product (Plex Media Server, Plex for iOS, Plex Web, etc.)
- **productVersion** (*string*) – Version of the product.
- **provides** (*str*) – List of services this resource provides (client, controller, sync-target, player, pubsub-player).
- **publicAddress** (*str*) – Public IP address.
- **screenDensity** (*str*) – Unknown
- **screenResolution** (*str*) – Screen resolution (750x1334, 1242x2208, etc.)
- **token** (*str*) – Plex authentication token for the device.
- **vendor** (*str*) – Device vendor (ubuntu, etc).
- **version** (*str*) – Unknown (1, 2, 1.3.3.3148-b38628e, 1.3.15, etc.)

connect(*timeout=None*)

Returns a new *PlexClient* or *PlexServer* Sometimes there is more than one address specified for a server or client. After trying to connect to all available addresses for this client and assuming at least one connection was successful, the PlexClient object is built and returned.

Raises
NotFound – When unable to connect to any addresses for this device.

delete()

Remove this device from your account.

syncItems()

Returns an instance of *SyncList* for current device.

Raises
BadRequest – when the device doesn't provides *sync-target*.

class plexapi.myplex.**MyPlexPinLogin**(*session=None*, *requestTimeout=None*, *headers=None*, *oauth=False*)

Bases: object

MyPlex PIN login class which supports getting a token for authenticating the client and providing an access token to create a *MyPlexAccount* instance. The login can be done using the four character PIN which the user must enter at https://plex.tv/link or using Plex OAuth.

This helper class supports a polling, threaded and callback approach.

- The polling approach expects the developer to periodically check if the PIN login was successful using *checkLogin()*.
- The threaded approach expects the developer to call *run()* and then at a later time call *waitForLogin()* to wait for and check the result.
- The callback approach is an extension of the threaded approach and expects the developer to pass the `callback` parameter to the call to *run()*. The callback will be called when the thread waiting for the PIN login to succeed either finishes or expires. The parameter passed to the callback is the received authentication token or `None` if the login expired.

Parameters

- **session** (*requests.Session, optional*) – Use your own session object if you want to cache the http responses from Plex.
- **requestTimeout** (*int, optional*) – Timeout in seconds on initial connect to plex.tv (default config.TIMEOUT).
- **headers** (*dict, optional*) – A dict of X-Plex headers to send with requests.
- **oauth** (*bool, optional*) – True to use Plex OAuth instead of PIN login.

Variables

- **PINS** (*str*) – 'https://plex.tv/api/v2/pins'
- **POLLINTERVAL** (*int*) – 1
- **pin** (*str*) – Four character PIN to use for the login at https://plex.tv/link.
- **finished** (*bool*) – Whether the pin login has finished or not.
- **expired** (*bool*) – Whether the pin login has expired or not.
- **token** (*str*) – Token retrieved after login.

Example

```
from plexapi.myplex import MyPlexAccount, MyPlexPinLogin

pinlogin = MyPlexPinLogin(oauth=True)
pinlogin.run()
print(f'Login to Plex at the following url:\n{pinlogin.oauthUrl()}')
pinlogin.waitForLogin()
token = pinlogin.token

account = MyPlexAccount(token=token)
```

property pin

Return the four character PIN used for linking a device at https://plex.tv/link.

oauthUrl(*forwardUrl=None*)

Return the Plex OAuth url for login.

Parameters
forwardUrl (*str, optional*) – The url to redirect the client to after login.

run(*callback=None*, *timeout=120*)

Starts the thread which monitors the PIN login state.

Parameters

- **callback** (*Callable[str], optional*) – Callback called with the received authentication token.
- **timeout** (*int, optional*) – Timeout in seconds to wait for user login. Default 120 seconds.

Raises

- **RuntimeError** – If the thread is already running.
- **RuntimeError** – If the PIN login for the current PIN has expired.

waitForLogin()

Waits for the PIN login to succeed or expire.

Returns
`True` if the PIN login succeeded or `False` otherwise.

Return type
bool

stop()

Stops the thread monitoring the PIN login state.

checkLogin()

Returns `True` if the PIN login has succeeded.

class plexapi.myplex.**MyPlexJWTLogin**(*session=None*, *requestTimeout=None*, *headers=None*, *oauth=False*, *token=None*, *jwtToken=None*, *keypair=(None, None)*, *scopes=None*)

Bases: `object`

MyPlex JWT login class which supports getting a JWT for authenticating the client and providing an access token to create a *MyPlexAccount* instance. The login can be done using the four character PIN which the user must enter at https://plex.tv/link or using Plex OAuth. This class can also be used to refresh or verify an existing JWT.

See: https://developer.plex.tv/pms/#section/API-Info/Authenticating-with-Plex

Using this class requires the `PyJWT` with `cryptography` packages to be installed (`pyjwt[crypto]`).

This helper class supports a polling, threaded and callback approach.

- The polling approach expects the developer to periodically check if the PIN login was successful using *checkLogin()*.
- The threaded approach expects the developer to call *run()* and then at a later time call *waitForLogin()* to wait for and check the result.
- The callback approach is an extension of the threaded approach and expects the developer to pass the `callback` parameter to the call to *run()*. The callback will be called when the thread waiting for the PIN login to succeed either finishes or expires. The parameter passed to the callback is the received authentication token or `None` if the login expired.

Parameters

- **session** (*requests.Session, optional*) – Use your own session object if you want to cache the http responses from Plex.
- **requestTimeout** (*int, optional*) – Timeout in seconds on initial connect to plex.tv (default config.TIMEOUT).
- **headers** (*dict, optional*) – A dict of X-Plex headers to send with requests.
- **oauth** (*bool, optional*) – True to use Plex OAuth instead of PIN login.
- **token** (*str, optional*) – Plex token only required to register the device initially if not using OAuth.
- **jwtToken** (*str, optional*) – Existing Plex JWT to verify or refresh.
- **keypair** (*tuple[str|bytes], optional*) – A tuple of the full file paths (str) to the ED25519 private and public key pair or the raw keys themselves (bytes) to use for signing the JWT. Use *generateKeypair()* to generate a new keypair if not provided.
- **scope** (*list[str], optional*) – List of scopes to request in the new token. Default is all available scopes.

Variables

- **PINS** (*str*) – 'https://plex.tv/api/v2/pins'
- **AUTH** (*str*) – 'https://clients.plex.tv/api/v2/auth'
- **POLLINTERVAL** (*int*) – 1
- **SCOPES** (*list*) – List of all available scopes to request for the JWT.
- **pin** (*str*) – Four character PIN to use for the login at https://plex.tv/link.
- **finished** (*bool*) – Whether the JWT login has finished or not.
- **expired** (*bool*) – Whether the JWT login has expired or not.
- **jwtToken** (*str*) – The Plex JWT received after login or refreshing.
- **decodedJWT** (*dict*) – The decoded Plex JWT payload.

Example

```
from plexapi.myplex import MyPlexAccount, MyPlexJWTLogin

# Method 1: Generate a new Plex JWT using Plex OAuth
jwtlogin = MyPlexJWTLogin(
    oauth=True,
    scopes=['username', 'email', 'friendly_name']
)
jwtlogin.generateKeypair(keyfiles=('private.key', 'public.key'))
jwtlogin.run()
print(f'Login to Plex at the following url:\n{jwtlogin.oauthUrl()}')
jwtlogin.waitForLogin()
jwtToken = jwtlogin.jwtToken

account = MyPlexAccount(token=jwtToken)

# Method 2: Generate a new Plex JWT using an existing Plex token and keypair
jwtlogin = MyPlexJWTLogin(
    token='2ffLuB84dqLswk9skLos',
    keypair=('private.key', 'public.key'),
    scopes=['username', 'email', 'friendly_name']
)
jwtlogin.registerDevice()
jwtToken = jwtlogin.refreshJWT()

account = MyPlexAccount(token=jwtToken)

# Refresh an existing Plex JWT
jwtlogin = MyPlexJWTLogin(
    jwtToken=jwtToken,
    keypair=('private.key', 'public.key'),
    scopes=['username', 'email', 'friendly_name']
)
if not jwtlogin.verifyJWT():
    jwtToken = jwtlogin.refreshJWT()

account = MyPlexAccount(token=jwtToken)
```

generateKeypair(*keyfiles=(None, None)*, *overwrite=False*)

Generates a new ED25519 private/public keypair for signing the JWT and saves them to files. Requires the `cryptography` package to be installed.

Parameters

- **keyfiles** (*tuple[str]*) – A tuple of the full file paths to write the private and public keypair to.
- **overwrite** (*bool*) – Set to True to overwrite existing keypair files. Default is False.

Raises

FileExistsError – when keypair files already exist and overwrite is False.

decodePlexJWT(*verify_signature=True*)

Returns the decoded Plex JWT with optional signature verification using the Plex public JWK.

Parameters

verify_signature (*bool*) – Whether to verify the JWT signature and required claims. Defaults to True. Set to False to skip signature verification and required-claim enforcement.

property decodedJWT

Returns the decoded Plex JWT with signature verification and required-claim enforcement.

registerDevice()

Registers the device with Plex using the provided token and private/public keypair. This must be done once if OAuth was not used before the Plex JWT can be refreshed.

Raises

BadRequest – when token or keypair is missing.

refreshJWT()

Refreshes the Plex JWT using the existing private/public keypair.

Returns

The new Plex JWT.

Return type

str

Raises

- *BadRequest* – when keypair is missing.
- *BadRequest* – when the newly obtained JWT cannot be verified.

verifyJWT(*refreshWithinDays=1*)

Verifies the existing Plex JWT is valid and not expiring within the specified number of days.

Parameters

refreshWithinDays (*int*) – Number of days before expiration to consider the JWT invalid and in need of refresh. Default is 1 day.

property pin

Return the four character PIN used for linking a device at https://plex.tv/link.

oauthUrl(*forwardUrl=None*)

Return the Plex OAuth url for login.

Parameters

forwardUrl (*str, optional*) – The url to redirect the client to after login.

run(*callback=None*, *timeout=120*)

Starts the thread which monitors the PIN login state.

Parameters

- **callback** (*Callable[str], optional*) – Callback called with the received authentication token.
- **timeout** (*int, optional*) – Timeout in seconds to wait for user login. Default 120 seconds.

Raises

- **RuntimeError** – If the thread is already running.
- **RuntimeError** – If the PIN login for the current PIN has expired.

waitForLogin()

Waits for the user login to succeed or expire.

Returns
`True` if the user login succeeded or `False` otherwise.

Return type
bool

stop()

Stops the thread monitoring the user login state.

checkLogin()

Returns `True` if the user login has succeeded.

class plexapi.myplex.**AccountOptOut**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single AccountOptOut 'https://plex.tv/api/v2/user/{userUUID}/settings/opt_outs'

Variables

- **TAG** (*str*) – optOut
- **key** (*str*) – Online Media Source key
- **value** (*str*) – Online Media Source opt_in, opt_out, or opt_out_managed

optIn()

Sets the Online Media Source to "Enabled".

optOut()

Sets the Online Media Source to "Disabled".

optOutManaged()

Sets the Online Media Source to "Disabled for Managed Users".

Raises
BadRequest – When trying to opt out music.

class plexapi.myplex.**UserState**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single UserState

Variables

- **TAG** (*str*) – UserState
- **lastViewedAt** (*datetime*) – Datetime the item was last played.
- **ratingKey** (*str*) – Unique key identifying the item.
- **type** (*str*) – The media type of the item.
- **viewCount** (*int*) – Count of times the item was played.
- **viewedLeafCount** (*int*) – Number of items marked as played in the show/season.
- **viewOffset** (*int*) – Time offset in milliseconds from the start of the content
- **viewState** (*bool*) – True or False if the item has been played.
- **watchlistedAt** (*datetime*) – Datetime the item was added to the watchlist.

class plexapi.myplex.**GeoLocation**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a signle IP address geolocation

Variables

- **TAG** (*str*) – location
- **city** (*str*) – City name
- **code** (*str*) – Country code
- **continentCode** (*str*) – Continent code
- **coordinates** (*Tuple<float>*) – Latitude and longitude
- **country** (*str*) – Country name
- **europeanUnionMember** (*bool*) – True if the country is a member of the European Union
- **inPrivacyRestrictedCountry** (*bool*) – True if the country is privacy restricted
- **postalCode** (*str*) – Postal code
- **subdivisions** (*str*) – Subdivision name
- **timezone** (*str*) – Timezone

CHAPTER

FIFTEEN

PHOTO PLEXAPI.PHOTO

class plexapi.photo.**Photoalbum**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexPartialObject*, *PhotoalbumMixins*

Represents a single Photoalbum (collection of photos).

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'photo'
- **addedAt** (*datetime*) – Datetime the photo album was added to the library.
- **art** (*str*) – URL to artwork image (/library/metadata/<ratingKey>/art/<artid>).
- **composite** (*str*) – URL to composite image (/library/metadata/<ratingKey>/composite/<compositeid>)
- **fields** (List<*Field*>) – List of field objects.
- **guid** (*str*) – Plex GUID for the photo album (local://229674).
- **images** (List<*Image*>) – List of image objects.
- **index** (*sting*) – Plex index number for the photo album.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **lastRatedAt** (*datetime*) – Datetime the photo album was last rated.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **librarySectionKey** (*str*) – *LibrarySection* key.
- **librarySectionTitle** (*str*) – *LibrarySection* title.
- **listType** (*str*) – Hardcoded as 'photo' (useful for search filters).
- **ratingKey** (*int*) – Unique key identifying the photo album.
- **summary** (*str*) – Summary of the photoalbum.
- **thumb** (*str*) – URL to thumbnail image (/library/metadata/<ratingKey>/thumb/<thumbid>).
- **title** (*str*) – Name of the photo album. (Trip to Disney World)
- **titleSort** (*str*) – Title to use when sorting (defaults to title).
- **type** (*str*) – 'photo'
- **updatedAt** (*datetime*) – Datetime the photo album was updated.
- **userRating** (*float*) – Rating of the photo album (0.0 - 10.0) equaling (0 stars - 5 stars).

album(*title*)

Returns the *Photoalbum* that matches the specified title.

Parameters
title (*str*) – Title of the photo album to return.

albums(***kwargs*)

Returns a list of *Photoalbum* objects in the album.

photo(*title*)

Returns the *Photo* that matches the specified title.

Parameters
title (*str*) – Title of the photo to return.

photos(***kwargs*)

Returns a list of *Photo* objects in the album.

clip(*title*)

Returns the *Clip* that matches the specified title.

Parameters
title (*str*) – Title of the clip to return.

clips(***kwargs*)

Returns a list of *Clip* objects in the album.

get(*title*)

Alias to *photo()*.

download(*savepath=None*, *keep_original_name=False*, *subfolders=False*)

Download all photos and clips from the photo album. See *download()* for details.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **keep_original_name** (*bool*) – True to keep the original filename otherwise a friendlier filename is generated.
- **subfolders** (*bool*) – True to separate photos/clips in to photo album folders.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.photo.**Photo**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexPartialObject*, *Playable*, *PhotoMixins*

Represents a single Photo.

Variables

- **TAG** (*str*) – 'Photo'
- **TYPE** (*str*) – 'photo'
- **addedAt** (*datetime*) – Datetime the photo was added to the library.
- **createdAtAccuracy** (*str*) – Unknown (local).
- **createdAtTZOffset** (*int*) – Unknown (-25200).
- **fields** (List<*Field*>) – List of field objects.

- **guid** (*str*) – Plex GUID for the photo (com.plexapp.agents.none://231714?lang=xn).
- **images** (List<*Image*>) – List of image objects.
- **index** (*sting*) – Plex index number for the photo.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **lastRatedAt** (*datetime*) – Datetime the photo was last rated.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **librarySectionKey** (*str*) – *LibrarySection* key.
- **librarySectionTitle** (*str*) – *LibrarySection* title.
- **listType** (*str*) – Hardcoded as 'photo' (useful for search filters).
- **media** (List<*Media*>) – List of media objects.
- **originallyAvailableAt** (*datetime*) – Datetime the photo was added to Plex.
- **parentGuid** (*str*) – Plex GUID for the photo album (local://229674).
- **parentIndex** (*int*) – Plex index number for the photo album.
- **parentKey** (*str*) – API URL of the photo album (/library/metadata/<parentRatingKey>).
- **parentRatingKey** (*int*) – Unique key identifying the photo album.
- **parentThumb** (*str*) – URL to photo album thumbnail image (/library/metadata/<parentRatingKey>/thumb/<thumbid>).
- **parentTitle** (*str*) – Name of the photo album for the photo.
- **ratingKey** (*int*) – Unique key identifying the photo.
- **sourceURI** (*str*) – Remote server URI (server://<machineIdentifier>/com.plexapp.plugins.library) (remote playlist item only).
- **summary** (*str*) – Summary of the photo.
- **tags** (List<*Tag*>) – List of tag objects.
- **thumb** (*str*) – URL to thumbnail image (/library/metadata/<ratingKey>/thumb/<thumbid>).
- **title** (*str*) – Name of the photo.
- **titleSort** (*str*) – Title to use when sorting (defaults to title).
- **type** (*str*) – 'photo'
- **updatedAt** (*datetime*) – Datetime the photo was updated.
- **userRating** (*float*) – Rating of the photo (0.0 - 10.0) equaling (0 stars - 5 stars).
- **year** (*int*) – Year the photo was taken.

photoalbum()

Return the photo's *Photoalbum*.

section()

Returns the *LibrarySection* the item belongs to.

property locations

This does not exist in plex xml response but is added to have a common interface to get the locations of the photo.

Returns
List<str> of file paths where the photo is found on disk.

sync(*resolution*, *client=None*, *clientId=None*, *limit=None*, *title=None*)

Add current photo as sync item for specified device. See *sync()* for possible exceptions.

Parameters

- **resolution** (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module *sync*.
- **client** (*MyPlexDevice*) – sync destination, see *sync()*.
- **clientId** (*str*) – sync destination, see *sync()*.
- **limit** (*int*) – maximum count of items to sync, unlimited if *None*.
- **title** (*str*) – descriptive title for the new *SyncItem*, if empty the value would be generated from metadata of current photo.

Returns
an instance of created syncItem.

Return type
SyncItem

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.photo.**PhotoSession**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexSession*, *Photo*

Represents a single Photo session loaded from *sessions()*.

CHAPTER

SIXTEEN

PLAYLIST PLEXAPI.PLAYLIST

class plexapi.playlist.**Playlist**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexPartialObject*, *Playable*, *PlaylistMixins*

Represents a single Playlist.

Variables

- **TAG** (*str*) – 'Playlist'
- **TYPE** (*str*) – 'playlist'
- **addedAt** (*datetime*) – Datetime the playlist was added to the server.
- **allowSync** (*bool*) – True if you allow syncing playlists.
- **composite** (*str*) – URL to composite image (/playlist/<ratingKey>/composite/<compositeid>)
- **content** (*str*) – The filter URI string for smart playlists.
- **duration** (*int*) – Duration of the playlist in milliseconds.
- **durationInSeconds** (*int*) – Duration of the playlist in seconds.
- **fields** (List<*Field*>) – List of field objects.
- **guid** (*str*) – Plex GUID for the playlist (com.plexapp.agents.none://XXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXX).
- **icon** (*str*) – Icon URI string for smart playlists.
- **key** (*str*) – API URL (/playlist/<ratingkey>).
- **leafCount** (*int*) – Number of items in the playlist view.
- **librarySectionID** (*int*) – Library section identifier (radio only)
- **librarySectionKey** (*str*) – Library section key (radio only)
- **librarySectionTitle** (*str*) – Library section title (radio only)
- **playlistType** (*str*) – 'audio', 'video', or 'photo'
- **radio** (*bool*) – If this playlist represents a radio station
- **ratingKey** (*int*) – Unique key identifying the playlist.
- **smart** (*bool*) – True if the playlist is a smart playlist.
- **summary** (*str*) – Summary of the playlist.
- **title** (*str*) – Name of the playlist.
- **titleSort** (*str*) – Title to use when sorting (defaults to title).

- **type** (*str*) – 'playlist'
- **updatedAt** (*datetime*) – Datetime the playlist was updated.

property thumb

Alias to self.composite.

property metadataType

Returns the type of metadata in the playlist (movie, track, or photo).

property isVideo

Returns True if this is a video playlist.

property isAudio

Returns True if this is an audio playlist.

property isPhoto

Returns True if this is a photo playlist.

filters()

Returns the search filter dict for smart playlist. The filter dict be passed back into *search()* to get the list of items.

section()

Returns the *LibrarySection* this smart playlist belongs to.

Raises

- ***plexapi.exceptions.BadRequest*** – When trying to get the section for a regular playlist.
- ***plexapi.exceptions.Unsupported*** – When unable to determine the library section.

item(*title*)

Returns the item in the playlist that matches the specified title.

Parameters
title (*str*) – Title of the item to return.

Raises
plexapi.exceptions.NotFound – When the item is not found in the playlist.

items()

Returns a list of all items in the playlist.

get(*title*)

Alias to *item()*.

addItems(*items*)

Add items to the playlist.

Parameters
items (*List*) – List of *Audio*, *Video*, or *Photo* objects to be added to the playlist.

Raises
plexapi.exceptions.BadRequest – When trying to add items to a smart playlist.

removeItems(*items*)

Remove items from the playlist.

Parameters
items (*List*) – List of *Audio*, *Video*, or *Photo* objects to be removed from the playlist.

Raises

- *plexapi.exceptions.BadRequest* – When trying to remove items from a smart playlist.
- *plexapi.exceptions.NotFound* – When the item does not exist in the playlist.

moveItem(*item*, *after=None*)

Move an item to a new position in the playlist.

Parameters

- **items** (*obj*) – *Audio*, *Video*, or *Photo* objects to be moved in the playlist.
- **after** (*obj*) – *Audio*, *Video*, or *Photo* objects to move the item after in the playlist.

Raises

- *plexapi.exceptions.BadRequest* – When trying to move items in a smart playlist.
- *plexapi.exceptions.NotFound* – When the item or item after does not exist in the playlist.

updateFilters(*limit=None*, *sort=None*, *filters=None*, ***kwargs*)

Update the filters for a smart playlist.

Parameters

- **limit** (*int*) – Limit the number of items in the playlist.
- **sort** (*str or list, optional*) – A string of comma separated sort fields or a list of sort fields in the format `column:dir`. See *search()* for more info.
- **filters** (*dict*) – A dictionary of advanced filters. See *search()* for more info.
- ****kwargs** (*dict*) – Additional custom filters to apply to the search results. See *search()* for more info.

Raises

plexapi.exceptions.BadRequest – When trying update filters for a regular playlist.

delete()

Delete the playlist.

classmethod create(*server*, *title*, *section=None*, *items=None*, *smart=False*, *limit=None*, *libtype=None*, *sort=None*, *filters=None*, *m3ufilepath=None*, ***kwargs*)

Create a playlist.

Parameters

- **server** (*PlexServer*) – Server to create the playlist on.
- **title** (*str*) – Title of the playlist.
- **section** (*LibrarySection*, str) – Smart playlists and m3u import only, the library section to create the playlist in.
- **items** (*List*) – Regular playlists only, list of *Audio*, *Video*, or *Photo* objects to be added to the playlist.
- **smart** (*bool*) – True to create a smart playlist. Default False.
- **limit** (*int*) – Smart playlists only, limit the number of items in the playlist.
- **libtype** (*str*) – Smart playlists only, the specific type of content to filter (movie, show, season, episode, artist, album, track, photoalbum, photo).

- **sort** (*str or list, optional*) – Smart playlists only, a string of comma separated sort fields or a list of sort fields in the format `column:dir`. See *search()* for more info.
- **filters** (*dict*) – Smart playlists only, a dictionary of advanced filters. See *search()* for more info.
- **m3ufilepath** (*str*) – Music playlists only, the full file path to an m3u file to import. Note: This will overwrite any playlist previously created from the same m3u file.
- ****kwargs** (*dict*) – Smart playlists only, additional custom filters to apply to the search results. See *search()* for more info.

Raises

- *plexapi.exceptions.BadRequest* – When no items are included to create the playlist.
- *plexapi.exceptions.BadRequest* – When mixing media types in the playlist.
- *plexapi.exceptions.BadRequest* – When attempting to import m3u file into non-music library.
- *plexapi.exceptions.BadRequest* – When failed to import m3u file.

Returns
A new instance of the created Playlist.

Return type
Playlist

copyToUser(*user*)

Copy playlist to another user account.

Parameters
user (*MyPlexUser* or str) – *MyPlexUser* object, username, email, or user id of the user to copy the playlist to.

sync(*videoQuality=None*, *photoResolution=None*, *audioBitrate=None*, *client=None*, *clientId=None*, *limit=None*, *unwatched=False*, *title=None*)

Add the playlist as a sync item for the specified device. See *sync()* for possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *sync* module. Used only when playlist contains video.
- **photoResolution** (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module *sync*. Used only when playlist contains photos.
- **audioBitrate** (*int*) – maximum bitrate for synchronized music, better use one of MUSIC_BITRATE_* values from the module *sync*. Used only when playlist contains audio.
- **client** (*MyPlexDevice*) – sync destination, see *sync()*.
- **clientId** (*str*) – sync destination, see *sync()*.
- **limit** (*int*) – maximum count of items to sync, unlimited if *None*.
- **unwatched** (*bool*) – if *True* watched videos wouldn't be synced.
- **title** (*str*) – descriptive title for the new *SyncItem*, if empty the value would be generated from metadata of current photo.

Raises

- *BadRequest* – When playlist is not allowed to sync.
- *Unsupported* – When playlist content is unsupported.

Returns
A new instance of the created sync item.

Return type
SyncItem

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

CHAPTER

SEVENTEEN

PLAYQUEUE PLEXAPI.PLAYQUEUE

class plexapi.playqueue.**PlayQueue**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Control a PlayQueue.

Variables

- **TAG** (*str*) – 'PlayQueue'
- **TYPE** (*str*) – 'playqueue'
- **identifier** (*str*) – com.plexapp.plugins.library
- **items** (*list*) – List of *Playable* or *Playlist*
- **mediaTagPrefix** (*str*) – Fx /system/bundle/media/flags/
- **mediaTagVersion** (*int*) – Fx 1485957738
- **playQueueID** (*int*) – ID of the PlayQueue.
- **playQueueLastAddedItemID** (*int*) – Defines where the "Up Next" region starts. Empty unless PlayQueue is modified after creation.
- **playQueueSelectedItemID** (*int*) – The queue item ID of the currently selected item.
- **playQueueSelectedItemOffset** (*int*) – The offset of the selected item in the PlayQueue, from the beginning of the queue.
- **playQueueSelectedMetadataItemID** (*int*) – ID of the currently selected item, matches ratingKey.
- **playQueueShuffled** (*bool*) – True if shuffled.
- **playQueueSourceURI** (*str*) – Original URI used to create the PlayQueue.
- **playQueueTotalCount** (*int*) – How many items in the PlayQueue.
- **playQueueVersion** (*int*) – Version of the PlayQueue. Increments every time a change is made to the PlayQueue.
- **selectedItem** (*Playable*) – Media object for the currently selected item.
- **_server** (*PlexServer*) – PlexServer associated with the PlayQueue.
- **size** (*int*) – Alias for playQueueTotalCount.

getQueueItem(*item*)

Accepts a media item and returns a similar object from this PlayQueue. Useful for looking up playQueueItemIDs using items obtained from the Library.

classmethod get(*server*, *playQueueID*, *own=False*, *center=None*, *window=50*, *includeBefore=True*, *includeAfter=True*)

Retrieve an existing *PlayQueue* by identifier.

Parameters

- **server** (*PlexServer*) – Server you are connected to.
- **playQueueID** (*int*) – Identifier of an existing PlayQueue.
- **own** (*bool, optional*) – If server should transfer ownership.
- **center** (*int, optional*) – The playQueueItemID of the center of the window. Does not change selectedItem.
- **window** (*int, optional*) – Number of items to return from each side of the center item.
- **includeBefore** (*bool, optional*) – Include items before the center, defaults True. Does not include center if False.
- **includeAfter** (*bool, optional*) – Include items after the center, defaults True. Does not include center if False.

classmethod create(*server*, *items*, *startItem=None*, *shuffle=0*, *repeat=0*, *includeChapters=1*, *includeRelated=1*, *continuous=0*)

Create and return a new *PlayQueue*.

Parameters

- **server** (*PlexServer*) – Server you are connected to.
- **items** (*PlexPartialObject*) – A media item or a list of media items.
- **startItem** (*Playable*, optional) – Media item in the PlayQueue where playback should begin.
- **shuffle** (*int, optional*) – Start the playqueue shuffled.
- **repeat** (*int, optional*) – Start the playqueue shuffled.
- **includeChapters** (*int, optional*) – include Chapters.
- **includeRelated** (*int, optional*) – include Related.
- **continuous** (*int, optional*) – include additional items after the initial item. For a show this would be the next episodes, for a movie it does nothing.

classmethod fromStationKey(*server*, *key*)

Create and return a new *PlayQueue*.

This is a convenience method to create a *PlayQueue* for radio stations when only the *key* string is available.

Parameters

- **server** (*PlexServer*) – Server you are connected to.
- **key** (*str*) – A station key as provided by *hubs()* or *station()*

Example

```
from plexapi.playqueue import PlayQueue
music = server.library.section("Music")
artist = music.get("Artist Name")
station = artist.station()
```

(continues on next page)

(continued from previous page)

```
key = station.key  # "/library/metadata/12855/station/8bd39616-dbdb-459e-b8da-
↪f46d0b170af4?type=10"
pq = PlayQueue.fromStationKey(server, key)
client = server.clients()[0]
client.playMedia(pq)
```

addItem(*item*, *playNext=False*, *refresh=True*)

Append the provided item to the "Up Next" section of the PlayQueue. Items can only be added to the section immediately following the current playing item.

Parameters

- **item** (*Playable* or *Playlist*) – Single media item or Playlist.
- **playNext** (*bool, optional*) – If True, add this item to the front of the "Up Next" section. If False, the item will be appended to the end of the "Up Next" section. Only has an effect if an item has already been added to the "Up Next" section. See https://support.plex.tv/articles/202188298-play-queues/ for more details.
- **refresh** (*bool, optional*) – Refresh the PlayQueue from the server before updating.

moveItem(*item*, *after=None*, *refresh=True*)

Moves an item to the beginning of the PlayQueue. If *after* is provided, the item will be placed immediately after the specified item.

Parameters

- **item** (*Playable*) – An existing item in the PlayQueue to move.
- **afterItemID** (*Playable*, optional) – A different item in the PlayQueue. If provided, *item* will be placed in the PlayQueue after this item.
- **refresh** (*bool, optional*) – Refresh the PlayQueue from the server before updating.

removeItem(*item*, *refresh=True*)

Remove an item from the PlayQueue.

Parameters

- **item** (*Playable*) – An existing item in the PlayQueue to move.
- **refresh** (*bool, optional*) – Refresh the PlayQueue from the server before updating.

clear()

Remove all items from the PlayQueue.

refresh()

Refresh the PlayQueue from the Plex server.

CHAPTER

EIGHTEEN

SERVER PLEXAPI.SERVER

class plexapi.server.**PlexServer**(*baseurl=None*, *token=None*, *session=None*, *timeout=None*)

Bases: *PlexObject*

This is the main entry point to interacting with a Plex server. It allows you to list connected clients, browse your library sections and perform actions such as emptying trash. If you do not know the auth token required to access your Plex server, or simply want to access your server with your username and password, you can also create an PlexServer instance from *MyPlexAccount*.

Parameters

- **baseurl** (*str*) – Base url for to access the Plex Media Server (default: 'http://localhost:32400').
- **token** (*str*) – Required Plex authentication token to access the server.
- **session** (*requests.Session, optional*) – Use your own session object if you want to cache the http responses from the server.
- **timeout** (*int, optional*) – Timeout in seconds on initial connection to the server (default config.TIMEOUT).

Variables

- **allowCameraUpload** (*bool*) – True if server allows camera upload.
- **allowChannelAccess** (*bool*) – True if server allows channel access (iTunes?).
- **allowMediaDeletion** (*bool*) – True is server allows media to be deleted.
- **allowSharing** (*bool*) – True is server allows sharing.
- **allowSync** (*bool*) – True is server allows sync.
- **backgroundProcessing** (*bool*) – Unknown
- **certificate** (*bool*) – True if server has an HTTPS certificate.
- **companionProxy** (*bool*) – Unknown
- **diagnostics** (*bool*) – Unknown
- **eventStream** (*bool*) – Unknown
- **friendlyName** (*str*) – Human friendly name for this server.
- **hubSearch** (*bool*) – True if Hub Search is enabled. I believe this is enabled for everyone
- **machineIdentifier** (*str*) – Unique ID for this server (looks like an md5).
- **multiuser** (*bool*) – True if multiusers are enabled.
- **myPlex** (*bool*) – Unknown (True if logged into myPlex?).

- **myPlexMappingState** (*str*) – Unknown (ex: mapped).
- **myPlexSigninState** (*str*) – Unknown (ex: ok).
- **myPlexSubscription** (*bool*) – True if you have a myPlex subscription.
- **myPlexUsername** (*str*) – Email address if signed into myPlex (user@example.com)
- **ownerFeatures** (*list*) – List of features allowed by the server owner. This may be based on your PlexPass subscription. Features include: camera_upload, cloudsync, content_filter, dvr, hardware_transcoding, home, lyrics, music_videos, pass, photo_autotags, premium_music_metadata, session_bandwidth_restrictions, sync, trailers, webhooks (and maybe more).
- **photoAutoTag** (*bool*) – True if photo auto-tagging is enabled.
- **platform** (*str*) – Platform the server is hosted on (ex: Linux)
- **platformVersion** (*str*) – Platform version (ex: '6.1 (Build 7601)', '4.4.0-59-generic').
- **pluginHost** (*bool*) – Unknown
- **readOnlyLibraries** (*bool*) – Unknown
- **requestParametersInCookie** (*bool*) – Unknown
- **streamingBrainVersion** (*bool*) – Current Streaming Brain version.
- **sync** (*bool*) – True if syncing to a device is enabled.
- **transcoderActiveVideoSessions** (*int*) – Number of active video transcoding sessions.
- **transcoderAudio** (*bool*) – True if audio transcoding audio is available.
- **transcoderLyrics** (*bool*) – True if audio transcoding lyrics is available.
- **transcoderPhoto** (*bool*) – True if audio transcoding photos is available.
- **transcoderSubtitles** (*bool*) – True if audio transcoding subtitles is available.
- **transcoderVideo** (*bool*) – True if audio transcoding video is available.
- **transcoderVideoBitrates** (*bool*) – List of video bitrates.
- **transcoderVideoQualities** (*bool*) – List of video qualities.
- **transcoderVideoResolutions** (*bool*) – List of video resolutions.
- **updatedAt** (*int*) – Datetime the server was updated.
- **updater** (*bool*) – Unknown
- **version** (*str*) – Current Plex version (ex: 1.3.2.3112-1751929)
- **voiceSearch** (*bool*) – True if voice search is enabled. (is this Google Voice search?)
- **_baseurl** (*str*) – HTTP address of the client.
- **_token** (*str*) – Token used to access this client.
- **_session** (*obj*) – Requests session object used to access this client.

property library

Library to browse or search your media.

property settings

Returns a list of all server settings.

identity()

Returns the Plex server identity.

account()

Returns the *Account* object this server belongs to.

claim(*account*)

Claim the Plex server using a *MyPlexAccount*. This will only work with an unclaimed server on localhost or the same subnet.

Parameters

account (*MyPlexAccount*) – The account used to claim the server.

unclaim()

Unclaim the Plex server. This will remove the server from your *MyPlexAccount*.

property activities

Returns all current PMS activities.

agents(*mediaType=None*)

Returns a list of *Agent* objects this server has available.

createToken(*type='delegation'*, *scope='all'*)

Create a temp access token for the server.

switchUser(*user*, *session=None*, *timeout=None*)

Returns a new *PlexServer* object logged in as the given username. Note: Only the admin account can switch to other users.

Parameters

- **user** (*MyPlexUser* or str) – *MyPlexUser* object, username, email, or user id of the user to log in to the server.
- **session** (*requests.Session, optional*) – Use your own session object if you want to cache the http responses from the server. This will default to the same session as the admin account if no new session is provided.
- **timeout** (*int, optional*) – Timeout in seconds on initial connection to the server. This will default to the same timeout as the admin account if no new timeout is provided.

Example

```
from plexapi.server import PlexServer
# Login to the Plex server using the admin token
plex = PlexServer('http://plexserver:32400', token='2ffLuB84dqLswk9skLos')
# Login to the same Plex server using a different account
userPlex = plex.switchUser("Username")
```

systemAccounts()

Returns a list of *SystemAccount* objects this server contains.

systemAccount(*accountID*)

Returns the *SystemAccount* object for the specified account ID.

Parameters

accountID (*int*) – The *SystemAccount* ID.

systemDevices()

Returns a list of *SystemDevice* objects this server contains.

systemDevice(*deviceID*)

Returns the *SystemDevice* object for the specified device ID.

Parameters
deviceID (*int*) – The *SystemDevice* ID.

myPlexAccount()

Returns a *MyPlexAccount* object using the same token to access this server. If you are not the owner of this PlexServer you're likely to receive an authentication error calling this.

browse(*path=None*, *includeFiles=True*)

Browse the system file path using the Plex API. Returns list of *Path* and *File* objects.

Parameters

- **path** (*Path* or str, optional) – Full path to browse.
- **includeFiles** (*bool*) – True to include files when browsing (Default). False to only return folders.

walk(*path=None*)

Walk the system file tree using the Plex API similar to *os.walk*. Yields a 3-tuple *(path, paths, files)* where *path* is a string of the directory path, *paths* is a list of *Path* objects, and *files* is a list of *File* objects.

Parameters
path (*Path* or str, optional) – Full path to walk.

isBrowsable(*path*)

Returns True if the Plex server can browse the given path.

Parameters
path (*Path* or str) – Full path to browse.

clients()

Returns list of all *PlexClient* objects connected to server.

client(*name*)

Returns the *PlexClient* that matches the specified name or machine identifier.

Parameters
name (*str*) – Name or machine identifier of the client to return.

Raises
NotFound – Unknown client name.

createCollection(*title*, *section*, *items=None*, *smart=False*, *limit=None*, *libtype=None*, *sort=None*, *filters=None*, ***kwargs*)

Creates and returns a new *Collection*.

Parameters

- **title** (*str*) – Title of the collection.
- **section** (*LibrarySection*, str) – The library section to create the collection in.
- **items** (*List*) – Regular collections only, list of *Audio*, *Video*, or *Photo* objects to be added to the collection.
- **smart** (*bool*) – True to create a smart collection. Default False.

- **limit** (*int*) – Smart collections only, limit the number of items in the collection.
- **libtype** (*str*) – Smart collections only, the specific type of content to filter (movie, show, season, episode, artist, album, track, photoalbum, photo).
- **sort** (*str or list, optional*) – Smart collections only, a string of comma separated sort fields or a list of sort fields in the format `column:dir`. See *search()* for more info.
- **filters** (*dict*) – Smart collections only, a dictionary of advanced filters. See *search()* for more info.
- ****kwargs** (*dict*) – Smart collections only, additional custom filters to apply to the search results. See *search()* for more info.

Raises

- ***plexapi.exceptions.BadRequest*** – When no items are included to create the collection.
- ***plexapi.exceptions.BadRequest*** – When mixing media types in the collection.

Returns
A new instance of the created Collection.

Return type
Collection

Example

```
# Create a regular collection
movies = plex.library.section("Movies")
movie1 = movies.get("Big Buck Bunny")
movie2 = movies.get("Sita Sings the Blues")
collection = plex.createCollection(
    title="Favorite Movies",
    section=movies,
    items=[movie1, movie2]
)

# Create a smart collection
collection = plex.createCollection(
    title="Recently Aired Comedy TV Shows",
    section="TV Shows",
    smart=True,
    sort="episode.originallyAvailableAt:desc",
    filters={"episode.originallyAvailableAt>>": "4w", "genre": "comedy"}
)
```

createPlaylist(*title*, *section=None*, *items=None*, *smart=False*, *limit=None*, *libtype=None*, *sort=None*, *filters=None*, *m3ufilepath=None*, ***kwargs*)

Creates and returns a new *Playlist*.

Parameters

- **title** (*str*) – Title of the playlist.
- **section** (*LibrarySection*, str) – Smart playlists and m3u import only, the library section to create the playlist in.

- **items** (*List*) – Regular playlists only, list of *Audio*, *Video*, or *Photo* objects to be added to the playlist.
- **smart** (*bool*) – True to create a smart playlist. Default False.
- **limit** (*int*) – Smart playlists only, limit the number of items in the playlist.
- **libtype** (*str*) – Smart playlists only, the specific type of content to filter (movie, show, season, episode, artist, album, track, photoalbum, photo).
- **sort** (*str or list, optional*) – Smart playlists only, a string of comma separated sort fields or a list of sort fields in the format `column:dir`. See *search()* for more info.
- **filters** (*dict*) – Smart playlists only, a dictionary of advanced filters. See *search()* for more info.
- **m3ufilepath** (*str*) – Music playlists only, the full file path to an m3u file to import. Note: This will overwrite any playlist previously created from the same m3u file.
- ****kwargs** (*dict*) – Smart playlists only, additional custom filters to apply to the search results. See *search()* for more info.

Raises

- *plexapi.exceptions.BadRequest* – When no items are included to create the playlist.
- *plexapi.exceptions.BadRequest* – When mixing media types in the playlist.
- *plexapi.exceptions.BadRequest* – When attempting to import m3u file into non-music library.
- *plexapi.exceptions.BadRequest* – When failed to import m3u file.

Returns

A new instance of the created Playlist.

Return type

Playlist

Example

```
# Create a regular playlist
episodes = plex.library.section("TV Shows").get("Game of Thrones").episodes()
playlist = plex.createPlaylist(
    title="GoT Episodes",
    items=episodes
)

# Create a smart playlist
playlist = plex.createPlaylist(
    title="Top 10 Unwatched Movies",
    section="Movies",
    smart=True,
    limit=10,
    sort="audienceRating:desc",
    filters={"audienceRating>>": 8.0, "unwatched": True}
)

# Create a music playlist from an m3u file
playlist = plex.createPlaylist(
```

(continues on next page)

(continued from previous page)

```
    title="Favorite Tracks",
    section="Music",
    m3ufilepath="/path/to/playlist.m3u"
)
```

createPlayQueue(*item*, ***kwargs*)

Creates and returns a new *PlayQueue*.

Parameters

- **item** (*Media or Playlist*) – Media or playlist to add to PlayQueue.
- **kwargs** (*dict*) – See *~plexapi.playqueue.PlayQueue.create*.

downloadDatabases(*savepath=None*, *unpack=False*, *showstatus=False*)

Download databases.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **unpack** (*bool*) – Unpack the zip file.
- **showstatus** (*bool*) – Display a progressbar.

downloadLogs(*savepath=None*, *unpack=False*, *showstatus=False*)

Download server logs.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **unpack** (*bool*) – Unpack the zip file.
- **showstatus** (*bool*) – Display a progressbar.

butlerTasks()

Return a list of `ButlerTask` objects.

runButlerTask(*task*)

Manually run a butler task immediately instead of waiting for the scheduled task to run. Note: The butler task is run asynchronously. Check Plex Web to monitor activity.

Parameters

task (*str*) – The name of the task to run. (e.g. 'BackupDatabase')

Example

```
availableTasks = [task.name for task in plex.butlerTasks()]
print("Available butler tasks:", availableTasks)
```

checkForUpdate(*force=True*, *download=False*)

Returns a *Release* object containing release info if an update is available or None if no update is available.

Parameters

- **force** (*bool*) – Force server to check for new releases
- **download** (*bool*) – Download if a update is available.

isLatest()

Returns True if the installed version of Plex Media Server is the latest.

canInstallUpdate()

Returns True if the newest version of Plex Media Server can be installed automatically. (e.g. Windows and Mac can install updates automatically, but Docker and NAS devices cannot.)

installUpdate()

Automatically install the newest version of Plex Media Server.

history(*maxresults=None*, *mindate=None*, *ratingKey=None*, *accountID=None*, *librarySectionID=None*)

Returns a list of media items from watched history. If there are many results, they will be fetched from the server in batches of X_PLEX_CONTAINER_SIZE amounts. If you're only looking for the first <num> results, it would be wise to set the maxresults option to that amount so this functions doesn't iterate over all results on the server.

Parameters

- **maxresults** (*int*) – Only return the specified number of results (optional).
- **mindate** (*datetime*) – Min datetime to return results from. This really helps speed up the result listing. For example: datetime.now() - timedelta(days=7)
- **ratingKey** (*int/str*)
- **accountID** (*int/str*)
- **librarySectionID** (*int/str*)

playlists(*playlistType=None*, *sectionId=None*, *title=None*, *sort=None*, ***kwargs*)

Returns a list of all *Playlist* objects on the server.

Parameters

- **playlistType** (*str, optional*) – The type of playlists to return (audio, video, photo). Default returns all playlists.
- **sectionId** (*int, optional*) – The section ID (key) of the library to search within.
- **title** (*str, optional*) – General string query to search for. Partial string matches are allowed.
- **sort** (*str or list, optional*) – A string of comma separated sort fields in the format `column:dir`.

playlist(*title*)

Returns the `Playlist` that matches the specified title.

Parameters

title (*str*) – Title of the playlist to return.

Raises

NotFound – Unable to find playlist.

optimizedItems(*removeAll=None*)

Returns list of all *Optimized* objects connected to server.

conversions(*pause=None*)

Returns list of all *Conversion* objects connected to server.

currentBackgroundProcess()

Returns list of all *TranscodeJob* objects running or paused on server.

query(*key*, *method=None*, *headers=None*, *params=None*, *timeout=None*, ***kwargs*)

Main method used to handle HTTPS requests to the Plex server. This method helps by encoding the response to utf-8 and parsing the returned XML into and ElementTree object. Returns None if no data exists in the response.

search(*query*, *mediatype=None*, *limit=None*, *sectionId=None*)

Returns a list of media items or filter categories from the resulting Hub Search against all items in your Plex library. This searches genres, actors, directors, playlists, as well as all the obvious media titles. It performs spell-checking against your search terms (because KUROSAWA is hard to spell). It also provides contextual search results. So for example, if you search for 'Pernice', it'll return 'Pernice Brothers' as the artist result, but we'll also go ahead and return your most-listened to albums and tracks from the artist. If you type 'Arnold' you'll get a result for the actor, but also the most recently added movies he's in.

Parameters

- **query** (*str*) – Query to use when searching your library.
- **mediatype** (*str, optional*) – Limit your search to the specified media type. actor, album, artist, autotag, collection, director, episode, game, genre, movie, photo, photoalbum, place, playlist, shared, show, tag, track
- **limit** (*int, optional*) – Limit to the specified number of results per Hub.
- **sectionId** (*int, optional*) – The section ID (key) of the library to search within.

continueWatching()

Return a list of all items in the Continue Watching hub.

sessions()

Returns a list of all active session (currently playing) media objects.

transcodeSessions()

Returns a list of all active *TranscodeSession* objects.

startAlertListener(*callback=None*, *callbackError=None*)

Creates a websocket connection to the Plex Server to optionally receive notifications. These often include messages from Plex about media scans as well as updates to currently running Transcode Sessions.

Returns a new *AlertListener* object.

Note: `websocket-client` must be installed in order to use this feature.

```
>> pip install websocket-client
```

Parameters

- **callback** (*func*) – Callback function to call on received messages.
- **callbackError** (*func*) – Callback function to call on errors.

Raises

Unsupported – Websocket-client not installed.

transcodeImage(*imageUrl*, *height*, *width*, *opacity=None*, *saturation=None*, *blur=None*, *background=None*, *blendColor=None*, *minSize=True*, *upscale=True*, *imageFormat=None*)

Returns the URL for a transcoded image.

Parameters

- **imageUrl** (*str*) – The URL to the image (eg. returned by *thumbUrl()* or *artUrl()*). The URL can be an online image.

- **height** (*int*) – Height to transcode the image to.
- **width** (*int*) – Width to transcode the image to.
- **opacity** (*int, optional*) – Change the opacity of the image (0 to 100)
- **saturation** (*int, optional*) – Change the saturation of the image (0 to 100).
- **blur** (*int, optional*) – The blur to apply to the image in pixels (e.g. 3).
- **background** (*str, optional*) – The background hex colour to apply behind the opacity (e.g. '000000').
- **blendColor** (*str, optional*) – The hex colour to blend the image with (e.g. '000000').
- **minSize** (*bool, optional*) – Maintain smallest dimension. Default True.
- **upscale** (*bool, optional*) – Upscale the image if required. Default True.
- **imageFormat** (*str, optional*) – 'jpeg' (default) or 'png'.

url(*key*, *includeToken=None*)

Build a URL string with proper token argument. Token will be appended to the URL if either includeToken is True or CONFIG.log.show_secrets is 'true'.

refreshSynclist()

Force PMS to download new SyncList from Plex.tv.

refreshContent()

Force PMS to refresh content for known SyncLists.

refreshSync()

Calls *refreshSynclist()* and *refreshContent()*, just like the Plex Web UI does when you click 'refresh'.

bandwidth(*timespan=None*, ***kwargs*)

Returns a list of *StatisticsBandwidth* objects with the Plex server dashboard bandwidth data.

Parameters

- **timespan** (*str, optional*) – The timespan to bin the bandwidth data. Default is seconds. Available timespans: seconds, hours, days, weeks, months.
- ****kwargs** (*dict, optional*) – Any of the available filters that can be applied to the bandwidth data. The time frame (at) and bytes can also be filtered using less than or greater than (see examples below).
  - accountID (int): The *SystemAccount* ID to filter.
  - **at (datetime): The time frame to filter (inclusive). The time frame can be either:**
    1. An exact time frame (e.g. Only December 1st 2020 *at=datetime(2020, 12, 1)*).
    2. Before a specific time (e.g. Before and including December 2020 *at<=datetime(2020, 12, 1)*).
    3. After a specific time (e.g. After and including January 2021 *at>=datetime(2021, 1, 1)*).
  - **bytes (int): The amount of bytes to filter (inclusive). The bytes can be either:**
    1. An exact number of bytes (not very useful) (e.g. *bytes=1024**3*).
    2. Less than or equal number of bytes (e.g. *bytes<=1024**3*).
    3. Greater than or equal number of bytes (e.g. *bytes>=1024**3*).

- deviceID (int): The *SystemDevice* ID to filter.
- **lan (bool): True to only retrieve local bandwidth, False to only retrieve remote bandwidth.**
  Default returns all local and remote bandwidth.

Raises

BadRequest – When applying an invalid timespan or unknown filter.

Example

```
from plexapi.server import PlexServer
plex = PlexServer('http://localhost:32400', token='xxxxxxxxxxxxxxxxxxxx')

# Filter bandwidth data for December 2020 and later, and more than 1 GB used.
filters = {
    'at>': datetime(2020, 12, 1),
    'bytes>': 1024**3
}

# Retrieve bandwidth data in one day timespans.
bandwidthData = plex.bandwidth(timespan='days', **filters)

# Print out bandwidth usage for each account and device combination.
for bandwidth in sorted(bandwidthData, key=lambda x: x.at):
    account = bandwidth.account()
    device = bandwidth.device()
    gigabytes = round(bandwidth.bytes / 1024**3, 3)
    local = 'local' if bandwidth.lan else 'remote'
    date = bandwidth.at.strftime('%Y-%m-%d')
    print(f'{account.name} used {gigabytes} GB of {local} bandwidth on {date}␣
↪from {device.name}')
```

resources()

Returns a list of *StatisticsResources* objects with the Plex server dashboard resources data.

getWebURL(*base=None*, *playlistTab=None*)

Returns the Plex Web URL for the server.

Parameters

- **base** (*str*) – The base URL before the fragment (`#!`). Default is https://app.plex.tv/desktop.
- **playlistTab** (*str*) – The playlist tab (audio, video, photo). Only used for the playlist URL.

class plexapi.server.**Account**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Contains the locally cached MyPlex account information. The properties provided don't match the *MyPlexAccount* object very well. I believe this exists because access to myplex is not required to get basic plex information. I can't imagine object is terribly useful except unless you were needed this information while offline.

Parameters

- **server** (*PlexServer*) – PlexServer this account is connected to (optional)
- **data** (*ElementTree*) – Response from PlexServer used to build this object (optional).

Variables

- **mappingError** (*str*) – Unknown
- **mappingErrorMessage** (*str*) – Unknown
- **mappingState** (*str*) – Unknown
- **privateAddress** (*str*) – Local IP address of the Plex server.
- **privatePort** (*str*) – Local port of the Plex server.
- **publicAddress** (*str*) – Public IP address of the Plex server.
- **publicPort** (*str*) – Public port of the Plex server.
- **signInState** (*str*) – Signin state for this account (ex: ok).
- **subscriptionActive** (*str*) – True if the account subscription is active.
- **subscriptionFeatures** (*str*) – List of features allowed by the server for this account. This may be based on your PlexPass subscription. Features include: camera_upload, cloudsync, content_filter, dvr, hardware_transcoding, home, lyrics, music_videos, pass, photo_autotags, premium_music_metadata, session_bandwidth_restrictions, sync, trailers, webhooks' (and maybe more).
- **subscriptionState** (*str*) – 'Active' if this subscription is active.
- **username** (*str*) – Plex account username (user@example.com).

class plexapi.server.**Activity**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

A currently running activity on the PlexServer.

class plexapi.server.**Release**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

class plexapi.server.**SystemAccount**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single system account.

Variables

- **TAG** (*str*) – 'Account'
- **autoSelectAudio** (*bool*) – True or False if the account has automatic audio language enabled.
- **defaultAudioLanguage** (*str*) – The default audio language code for the account.
- **defaultSubtitleLanguage** (*str*) – The default subtitle language code for the account.
- **id** (*int*) – The Plex account ID.
- **key** (*str*) – API URL (/accounts/<id>)
- **name** (*str*) – The username of the account.
- **subtitleMode** (*bool*) – The subtitle mode for the account.
- **thumb** (*str*) – URL for the account thumbnail.

class plexapi.server.**SystemDevice**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single system device.

Variables

- **TAG** (*str*) – 'Device'
- **clientIdentifier** (*str*) – The unique identifier for the device.
- **createdAt** (*datetime*) – Datetime the device was created.
- **id** (*int*) – The ID of the device (not the same as *MyPlexDevice* ID).
- **key** (*str*) – API URL (/devices/<id>)
- **name** (*str*) – The name of the device.
- **platform** (*str*) – OS the device is running (Linux, Windows, Chrome, etc.)

class plexapi.server.**StatisticsBandwidth**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single statistics bandwidth data.

Variables

- **TAG** (*str*) – 'StatisticsBandwidth'
- **accountID** (*int*) – The associated *SystemAccount* ID.
- **at** (*datetime*) – Datetime of the bandwidth data.
- **bytes** (*int*) – The total number of bytes for the specified time span.
- **deviceID** (*int*) – The associated *SystemDevice* ID.
- **lan** (*bool*) – True or False whether the bandwidth is local or remote.
- **timespan** (*int*) – The time span for the bandwidth data. 1: months, 2: weeks, 3: days, 4: hours, 6: seconds.

account()

Returns the *SystemAccount* associated with the bandwidth data.

device()

Returns the *SystemDevice* associated with the bandwidth data.

class plexapi.server.**StatisticsResources**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single statistics resources data.

Variables

- **TAG** (*str*) – 'StatisticsResources'
- **at** (*datetime*) – Datetime of the resource data.
- **hostCpuUtilization** (*float*) – The system CPU usage %.
- **hostMemoryUtilization** (*float*) – The Plex Media Server CPU usage %.
- **processCpuUtilization** (*float*) – The system RAM usage %.
- **processMemoryUtilization** (*float*) – The Plex Media Server RAM usage %.
- **timespan** (*int*) – The time span for the resource data (6: seconds).

class plexapi.server.**ButlerTask**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single scheduled butler task.

Variables

- **TAG** (*str*) – 'ButlerTask'
- **description** (*str*) – The description of the task.
- **enabled** (*bool*) – Whether the task is enabled.
- **interval** (*int*) – The interval the task is run in days.
- **name** (*str*) – The name of the task.
- **scheduleRandomized** (*bool*) – Whether the task schedule is randomized.
- **title** (*str*) – The title of the task.

class plexapi.server.**Identity**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a server identity.

Variables

- **claimed** (*bool*) – True or False if the server is claimed.
- **machineIdentifier** (*str*) – The Plex server machine identifier.
- **version** (*str*) – The Plex server version.

CHAPTER

NINETEEN

SETTINGS PLEXAPI.SETTINGS

class plexapi.settings.**Settings**(*server*, *data*, *initpath=None*)

Bases: *PlexObject*

Container class for all settings. Allows getting and setting PlexServer settings.

Variables

key (*str*) – '/:/prefs'

all()

Returns a list of all *Setting* objects available.

get(*id*)

Return the *Setting* object with the specified id.

groups()

Returns a dict of lists for all *Setting* objects grouped by setting group.

group(*group*)

Return a list of all *Setting* objects in the specified group.

Parameters

group (*str*) – Group to return all settings.

save()

Save any outstanding setting changes to the *PlexServer*. This performs a full reload() of Settings after complete.

class plexapi.settings.**Setting**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a single Plex setting.

Variables

- **id** (*str*) – Setting id (or name).
- **label** (*str*) – Short description of what this setting is.
- **summary** (*str*) – Long description of what this setting is.
- **type** (*str*) – Setting type (text, int, double, bool).
- **default** (*str*) – Default value for this setting.
- **value** (*str,bool,int,float*) – Current value for this setting.
- **hidden** (*bool*) – True if this is a hidden setting.
- **advanced** (*bool*) – True if this is an advanced setting.
- **group** (*str*) – Group name this setting is categorized as.

- **enumValues** (*list,dict*) – List or dictionary of valid values for this setting.

set(*value*)

Set a new value for this setting. NOTE: You must call plex.settings.save() for before any changes to setting values are persisted to the *PlexServer*.

toUrl()

Helper for urls

class plexapi.settings.**Preferences**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Setting*

Represents a single Preferences.

Variables

- **TAG** (*str*) – 'Setting'
- **FILTER** (*str*) – 'preferences'

General Settings

butlerUpdateChannel (text)
Update Channel. (default: 16; choices: 16:Public|8:Plex Pass)

collectUsageData (bool)
Send anonymous usage data to Plex. This helps us improve your experience (for example, to help us match movies and TV shows). (default: True)

friendlyName (text)
Friendly name. This name will be used to identify this media server to other computers on your network. If you leave it blank, your computer's name will be used instead.

logDebug (bool)
Enable Plex Media Server debug logging. (default: True)

logTokens (bool)
Allow Plex Media Server tokens in logs. Media server tokens can be used to gain access to library content. Don't share logs containing tokens publicly. A server restart is required for a change to take effect.

logVerbose (bool)
Enable Plex Media Server verbose logging.

Scheduled Task Settings

butlerDatabaseBackupPath (text)
Backup directory. The directory in which database backups are stored. (default: /var/lib/plexmediaserver/Library/Application Support/Plex Media Server/Plug-in Support/Databases)

butlerEndHour (int)
Time at which tasks stop running. The time at which the background maintenance tasks stop running. (default: 5; choices: 0:Midnight|1:1 am|2:2 am|3:3 am|4:4 am|5:5 am|6:6 am|7:7 am|8:8 am|9:9 am|10:10 am|11:11 am|12:Noon|13:1 pm|14:2 pm|15:3 pm|16:4 pm|17:5 pm|18:6 pm|19:7 pm|20:8 pm|21:9 pm|22:10 pm|23:11 pm)

butlerStartHour (int)
Time at which tasks start to run. The time at which the server starts running background maintenance tasks. (default: 2; choices: 0:Midnight|1:1 am|2:2 am|3:3 am|4:4 am|5:5 am|6:6 am|7:7 am|8:8 am|9:9 am|10:10 am|11:11 am|12:Noon|13:1 pm|14:2 pm|15:3 pm|16:4 pm|17:5 pm|18:6 pm|19:7 pm|20:8 pm|21:9 pm|22:10 pm|23:11 pm)

butlerTaskBackupDatabase (bool)
Backup database every three days. (default: True)

butlerTaskCleanOldBundles (bool)
Remove old bundles every week. (default: True)

butlerTaskCleanOldCacheFiles (bool)
Remove old cache files every week. (default: True)

butlerTaskDeepMediaAnalysis (bool)
Perform extensive media analysis during maintenance. (default: True)

butlerTaskOptimizeDatabase (bool)
Optimize database every week. (default: True)

butlerTaskRefreshEpgGuides (bool)
Perform refresh of program guide data.. (default: True)

butlerTaskRefreshLibraries (bool)
Update all libraries during maintenance.

butlerTaskRefreshLocalMedia (bool)
Refresh local metadata every three days. (default: True)

butlerTaskRefreshPeriodicMetadata (bool)
Refresh metadata periodically. (default: True)

butlerTaskUpgradeMediaAnalysis (bool)
Upgrade media analysis during maintenance. (default: True)

Channels Settings

disableCapabilityChecking (bool)
Disable capability checking. Capability checking ensures that plug-ins that are incompatible with this version of the server or the current client application you are using are hidden. Disabling capability checking is useful during development, but will enable access to plug-ins that may perform unreliably with certain client applications.

iTunesLibraryXmlPath (text)
iTunes library XML path.

iTunesSharingEnabled (bool)
Enable iTunes channel. A server restart is required for a change to take effect.

pluginsLaunchTimeout (int)
Number of seconds to wait before a plugin times out. (default: 180)

DLNA Settings

dlnaAnnouncementLeaseTime (int)
DLNA server announcement lease time. Duration in seconds of DLNA Server SSDP announcement lease time. (default: 1800)

dlnaClientPreferences (text)
DLNA client preferences. Client-specific configuration settings for the DLNA server.

dlnaDefaultProtocolInfo (text)
DLNA default protocol info. Protocol info string used in GetProtocolInfo responses by the DLNA server. (default: http-get:*:video/mpeg:*,http-get:*:video/mp4:*,http-get:*:video/vnd.dlna.mpeg-tts:*,http-get:*:video/avi:*,http-get:*:video/x-matroska:*,http-get:*:video/x-ms-wmv:*,http-get:*:video/wtv:*,http-get:*:audio/mpeg:*,http-get:*:audio/mp3:*,http-get:*:audio/mp4:*,http-get:*:audio/x-ms-wma*,http-get:*:audio/wav:*,http-get:*:audio/L16:*,http-get:*image/jpeg:*,http-get:*image/png:*,http-get:*image/gif:*,http-get:*image/tiff:*)

dlnaDescriptionIcons (text)
DLNA server description icons. Icons offered by DLNA server when devices request server description. (default: png,jpeg;260x260,120x120,48x48)

dlnaDeviceDiscoveryInterval (int)
DLNA media renderer discovery interval. Number of seconds between DLNA media renderer discovery requests. (default: 60)

dlnaEnabled (bool)
Enable the DLNA server. This allows the server to stream media to DLNA (Digital Living Network Alliance) devices. (default: True)

dlnaPlatinumLoggingLevel (text)
DLNA server logging level. (default: OFF; choices: OFF|FATAL|SEVERE|WARNING|INFO|FINE|FINER|FINEST|ALL)

dlnaReportTimeline (bool)
DLNA server timeline reporting. Enable the DLNA server to report timelines for video play activity. (default: True)

Extras Settings

cinemaTrailersFromBluRay (bool)
Include Cinema Trailers from new and upcoming movies on Blu-ray. This feature is Plex Pass only.

cinemaTrailersFromLibrary (bool)
Include Cinema Trailers from movies in my library. (default: True)

cinemaTrailersFromTheater (bool)
Include Cinema Trailers from new and upcoming movies in theaters. This feature is Plex Pass only.

cinemaTrailersPrerollID (text)
Cinema Trailers pre-roll video. Copy and paste the video's detail page URL into this field.

cinemaTrailersType (int)
Choose Cinema Trailers from. (default: 1; choices: 0:All movies|1:Only unwatched movies)

Library Settings

allowMediaDeletion (bool)
Allow media deletion. The owner of the server will be allowed to delete media files from disk. (default: True)

autoEmptyTrash (bool)
Empty trash automatically after every scan. (default: True)

fSEventLibraryPartialScanEnabled (bool)
Run a partial scan when changes are detected. When changes to library folders are detected, only scan the folder that changed.

fSEventLibraryUpdatesEnabled (bool)
Update my library automatically. Your library will be updated automatically when changes to library folders are detected.

generateBIFBehavior (text)
Generate video preview thumbnails. Video preview thumbnails provide live updates in Now Playing and while seeking on supported apps. Thumbnail generation may take a long time, cause high CPU usage, and consume additional disk space. You can turn off thumbnail generation for individual libraries in the library's advanced settings. (default: never; choices: never:never|scheduled:as a scheduled task|asap:as a scheduled task and when media is added)

generateChapterThumbBehavior (text)
Generate chapter thumbnails. Chapter thumbnails provide images in the chapter view on supported apps. They can take a long time to generate and consume additional disk space. (default: scheduled; choices: never:never|scheduled:as a scheduled task|asap:as a scheduled task and when media is added)

onDeckWindow (int)
Weeks to consider for On Deck. Shows that have not been watched in this many weeks will not appear in On Deck. (default: 16)

scannerLowPriority (bool)
Run scanner tasks at a lower priority.

scheduledLibraryUpdateInterval (int)
Library update interval. (default: 3600; choices: 900:every 15 minutes|1800:every 30 minutes|3600:hourly|7200:every 2 hours|21600:every 6 hours|43200:every 12 hours|86400:daily)

scheduledLibraryUpdatesEnabled (bool)
Update my library periodically.

watchMusicSections (bool)
Include music libraries in automatic updates. Linux systems limit the maximum number of watched directories; this may cause problems with large music libraries.

Network Settings

allowedNetworks (text)
List of IP addresses and networks that are allowed without auth. Comma separated list of IP addresses or IP/netmask entries for networks that are allowed to access Plex Media Server without logging in. When the server is signed out and this value is set, only localhost and addresses on this list will be allowed.

configurationUrl (text)
Web Manager URL. (default: http://127.0.0.1:32400/web)

customCertificateDomain (text)
Custom certificate domain. Domain name to be published to plex.tv using your mapped port; must match a name from the custom certificate file.

customCertificateKey (text)
Custom certificate encryption key.

customCertificatePath (text)
Custom certificate location. Path to a PKCS #12 file containing a certificate and private key to enable TLS support on a custom domain.

customConnections (text)
Custom server access URLs. A comma-separated list of URLs (http or https) which are published up to plex.tv for server discovery.

enableHttpPipelining (bool)
Enable HTTP Pipelining. This feature can enable higher performance in the HTTP server component. A server restart is required for a change to take effect. (default: True)

enableIPv6 (bool)
Enable server support for IPv6.

gdmEnabled (bool)
Enable local network discovery (GDM). This enables the media server to discover other servers and players on the local network. (default: True)

lanNetworksBandwidth (text)
LAN Networks. Comma separated list of IP addresses or IP/netmask entries for networks that will be considered to be on the local network when enforcing bandwidth restrictions. If set, all other IP addresses will be considered to be on the external network and and will be subject to external network bandwidth restrictions. If left blank, only the server's subnet is considered to be on the local network.

secureConnections (int)
Secure connections. When set to "Required", some unencrypted connections (originating from the Media Server

computer) will still be allowed and apps that don't support secure connections will not be able to connect at all. (default: 1; choices: 0:Required|1:Preferred|2:Disabled)

wanPerUserStreamCount (int)
Remote streams allowed per user. Maximum number of simultaneous streams each user is allowed when not on the local network. (choices: 0:Unlimited|1:1|2:2|3:3|4:4|5:5|6:6|7:7|8:8|9:9|10:10|11:11|12:12|13:13|14:14|15:15|16:16|17:17|18:18|19:19|20:20)

webHooksEnabled (bool)
Webhooks. This feature enables your server to send events to external services. (default: True)

Transcoder Settings

hardwareAcceleratedCodecs (bool)
Use hardware acceleration when available (Experimental). Plex Media Server will attempt to use hardware-accelerated video codecs when encoding and decoding video. Hardware acceleration can make transcoding faster and allow more simultaneous video transcodes, but it can also reduce video quality and compatibility.

segmentedTranscoderTimeout (int)
Segmented transcoder timeout. Timeout in seconds segmented transcodes wait for the transcoder to begin writing data. (default: 20)

transcodeCountLimit (int)
Maximum simultaneous video transcode. Limit the number of simultaneous video transcode streams your server can utilize (choices: 0:Unlimited|1:1|2:2|3:3|4:4|5:5|6:6|7:7|8:8|9:9|10:10|11:11|12:12|13:13|14:14|15:15|16:16|17:17|18:18|19:19|20:20)

transcoderDefaultDuration (int)
Transcoder default duration. Duration in minutes to use when transcoding something with an unknown duration. (default: 120)

transcoderH264BackgroundPreset (text)
Background transcoding x264 preset. The x264 preset value used for background transcoding (Sync and Media Optimizer). Slower values will result in better video quality and smaller file sizes, but will take significantly longer to complete processing. (default: veryfast; choices: ultrafast:Ultra fast|superfast:Super fast|veryfast:Very fast|faster:Faster|fast:Fast|medium:Medium|slow:Slow|slower:Slower|veryslow:Very slow)

transcoderPruneBuffer (int)
Transcoder default prune buffer. Amount in past seconds to retain before pruning segments from a transcode. (default: 300)

transcoderQuality (int)
Transcoder quality. Quality profile used by the transcoder. (choices: 0:Automatic|1:Prefer higher speed encoding|2:Prefer higher quality encoding|3:Make my CPU hurt)

transcoderTempDirectory (text)
Transcoder temporary directory. Directory to use when transcoding for temporary files.

transcoderThrottleBuffer (int)
Transcoder default throttle buffer. Amount in seconds to buffer before throttling the transcoder. (default: 60)

Misc Settings

acceptedEULA (bool)
Has the user accepted the EULA.

articleStrings (text)
Comma-separated list of strings considered articles when sorting titles. A server restart is required for a change to take effect.. (default: the,das,der,a,an,el,la)

languageInCloud (bool)
Use language preferences from plex.tv.

machineIdentifier (text)
A unique identifier for the machine.

publishServerOnPlexOnlineKey (bool)
Publish server on Plex Online. Publishing a server makes it automatically available on your client devices without any configuration of your router.

transcoderCanOnlyRemuxVideo (bool)
The transcoder can only remux video.

transcoderVideoResolutionLimit (text)
Maximum video output resolution for the transcoder. (default: 0x0)

wanPerStreamMaxUploadRate (int)
Limit remote stream bitrate. Set the maximum bitrate of a remote stream from this server. (choices: 0:Original (No limit)|20000:20 Mbps (1080p)|12000:12 Mbps (1080p)|10000:10 Mbps (1080p)|8000:8 Mbps (1080p)|4000:4 Mbps (720p)|3000:3 Mbps (720p)|2000:2 Mbps (480p)|1500:1.5 Mbps (480p)|720:720 kbps|320:320 kbps)

wanTotalMaxUploadRate (int)
External network total upload limit (kbps). Speed at which to limit the total bandwidth not on the local network in kilobits per second. Use 0 to set no limit.

Undocumented Settings

- **aBRKeepOldTranscodes (bool)**
- **allowHighOutputBitrates (bool)**
- **backgroundQueueIdlePaused (bool)**
- **butlerTaskGarbageCollectBlobs (bool)**
- **butlerTaskGenerateMediaIndexFiles (bool)**
- **certificateVersion (int)**: default: 2
- **dvrShowUnsupportedDevices (bool)**
- **enableABRDebugOverlay (bool)**
- **enableAirplay (bool)**
- **eyeQUser (text)**
- **forceAutoAdjustQuality (bool)**
- **generateIndexFilesDuringAnalysis (bool)**
- **gracenoteUser (text)**
- **hardwareDevicePath (text)**: default: /dev/dri/renderD128
- **lastAutomaticMappedPort (int)**
- **manualPortMappingMode (bool)**
- **manualPortMappingPort (int)**: default: 32400
- **minimumProgressTime (int)**: default: 60000
- **plexMetricsUrl (text)**: default: https://metrics.plex.tv
- **plexOnlineMail (text)**
- **plexOnlineUrl (text)**: default: https://plex.tv
- **syncMyPlexLoginGCDeferral (int)**: default: 14400

- **syncPagingItemsLimit (int)**: default: 100
- **systemAudioCodecs (bool)**: default: True
- **transcoderH264MinimumCRF (double)**: default: 16.0
- **transcoderH264Options (text)**
- **transcoderH264OptionsOverride (text)**
- **transcoderH264Preset (text)**: default: veryfast
- **transcoderLivePruneBuffer (int)**: default: 5400
- **transcoderLogLevel (text)**: default: error

CHAPTER

TWENTY

SONOS PLEXAPI.SONOS

class plexapi.sonos.**PlexSonosClient**(*account*, *data*, *timeout=None*)

Bases: *PlexClient*

Class for interacting with a Sonos speaker via the Plex API. This class makes requests to an external Plex API which then forwards the Sonos-specific commands back to your Plex server & Sonos speakers. Use of this feature requires an active Plex Pass subscription and Sonos speakers linked to your Plex account. It also requires remote access to be working properly.

More details on the Sonos integration are available here: https://support.plex.tv/articles/218237558-requirements-for-using-plex-for-sonos/

The Sonos API emulates the Plex player control API closely: https://github.com/plexinc/plex-media-player/wiki/Remote-control-API

Parameters

- **account** (PlexAccount) – PlexAccount instance this Sonos speaker is associated with.
- **data** (*ElementTree*) – Response from Plex Sonos API used to build this client.

Variables

- **deviceClass** (*str*) – "speaker"
- **lanIP** (*str*) – Local IP address of speaker.
- **machineIdentifier** (*str*) – Unique ID for this device.
- **platform** (*str*) – "Sonos"
- **platformVersion** (*str*) – Build version of Sonos speaker firmware.
- **product** (*str*) – "Sonos"
- **protocol** (*str*) – "plex"
- **protocolCapabilities** (*list<str>*) – List of client capabilities (timeline, playback, playqueues, provider-playback)
- **server** (*PlexServer*) – Server this client is connected to.
- **session** (Session) – Session object used for connection.
- **title** (*str*) – Name of this Sonos speaker.
- **token** (*str*) – X-Plex-Token used for authentication
- **_baseurl** (*str*) – Address of public Plex Sonos API endpoint.
- **_commandId** (*int*) – Counter for commands sent to Plex API.
- **_token** (*str*) – Token associated with linked Plex account.

- **_session** (*obj*) – Requests session object used to access this client.

playMedia(*media*, *offset=0*, ***params*)

Start playback of the specified media item. See also:

Parameters

- **media** (*Media*) – Media item to be played back (movie, music, photo, playlist, playqueue).
- **offset** (*int*) – Number of milliseconds at which to start playing with zero representing the beginning (default 0).
- ****params** (*dict*) – Optional additional parameters to include in the playback request. See also: https://github.com/plexinc/plex-media-player/wiki/Remote-control-API#modified-commands

CHAPTER

TWENTYONE

SYNC PLEXAPI.SYNC

You can work with Mobile Sync on other devices straight away, but if you'd like to use your app as a *sync-target* (when you can set items to be synced to your app) you need to init some variables.

```
def init_sync():
    import plexapi
    plexapi.X_PLEX_PROVIDES = 'sync-target'
    plexapi.BASE_HEADERS['X-Plex-Sync-Version'] = '2'
    plexapi.BASE_HEADERS['X-Plex-Provides'] = plexapi.X_PLEX_PROVIDES

    # mimic iPhone SE
    plexapi.X_PLEX_PLATFORM = 'iOS'
    plexapi.X_PLEX_PLATFORM_VERSION = '11.4.1'
    plexapi.X_PLEX_DEVICE = 'iPhone'

    plexapi.BASE_HEADERS['X-Plex-Platform'] = plexapi.X_PLEX_PLATFORM
    plexapi.BASE_HEADERS['X-Plex-Platform-Version'] = plexapi.X_PLEX_PLATFORM_VERSION
    plexapi.BASE_HEADERS['X-Plex-Device'] = plexapi.X_PLEX_DEVICE
```

You have to fake platform/device/model because transcoding profiles are hardcoded in Plex, and you obviously have to explicitly specify that your app supports *sync-target*.

class plexapi.sync.**SyncItem**(*server*, *data*, *initpath=None*, *clientIdentifier=None*)

Bases: *PlexObject*

Represents single sync item, for specified server and client. When you saying in the UI to sync "this" to "that" you're basically creating a sync item.

Variables

- **id** (*int*) – unique id of the item.
- **clientIdentifier** (*str*) – an identifier of Plex Client device, to which the item is belongs.
- **machineIdentifier** (*str*) – the id of server which holds all this content.
- **version** (*int*) – current version of the item. Each time you modify the item (e.g. by changing amount if media to sync) the new version is created.
- **rootTitle** (*str*) – the title of library/media from which the sync item was created. E.g.:
  - when you create an item for an episode 3 of season 3 of show Example, the value would be *Title of Episode 3*
  - when you create an item for a season 3 of show Example, the value would be *Season 3*

- when you set to sync all your movies in library named "My Movies" to value would be *My Movies*.

- **title** (*str*) – the title which you've set when created the sync item.
- **metadataType** (*str*) – the type of media which hides inside, can be *episode*, *movie*, etc.
- **contentType** (*str*) – basic type of the content: *video* or *audio*.
- **status** (*Status*) – current status of the sync.
- **mediaSettings** (*MediaSettings*) – media transcoding settings used for the item.
- **policy** (*Policy*) – the policy of which media to sync.
- **location** (*str*) – plex-style library url with all required filters / sorting.

server()

Returns *MyPlexResource* with server of current item.

getMedia()

Returns list of *Playable* which belong to this sync item.

markDownloaded(*media*)

Mark the file as downloaded (by the nature of Plex it will be marked as downloaded within any SyncItem where it presented).

Parameters
media (base.Playable) – the media to be marked as downloaded.

delete()

Removes current SyncItem

class plexapi.sync.**SyncList**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexObject*

Represents a Mobile Sync state, specific for single client, within one SyncList may be presented items from different servers.

Variables

- **clientId** (*str*) – an identifier of the client.
- **items** (List<*SyncItem*>) – list of registered items to sync.

class plexapi.sync.**Status**(*itemsCount*, *itemsCompleteCount*, *state*, *totalSize*, *itemsDownloadedCount*, *itemsReadyCount*, *itemsSuccessfulCount*, *failureCode*, *failure*)

Bases: object

Represents a current status of specific *SyncItem*.

Variables

- **failureCode** – unknown, never got one yet.
- **failure** – unknown.
- **state** (*str*) – server-side status of the item, can be *completed*, *pending*, empty, and probably something else.
- **itemsCount** (*int*) – total items count.
- **itemsCompleteCount** (*int*) – count of transcoded and/or downloaded items.
- **itemsDownloadedCount** (*int*) – count of downloaded items.

- **itemsReadyCount** (*int*) – count of transcoded items, which can be downloaded.
- **totalSize** (*int*) – total size in bytes of complete items.
- **itemsSuccessfulCount** (*int*) – unknown, in my experience it always was equal to *itemsCompleteCount*.

class plexapi.sync.**MediaSettings**(*maxVideoBitrate=4000*, *videoQuality=100*, *videoResolution='1280x720'*, *audioBoost=100*, *musicBitrate=192*, *photoQuality=74*, *photoResolution='1920x1080'*, *subtitleSize=100*)

Bases: object

Transcoding settings used for all media within *SyncItem*.

Variables

- **audioBoost** (*int*) – unknown.
- **maxVideoBitrate** (*int* | *str*) – maximum bitrate for video, may be empty string.
- **musicBitrate** (*int* | *str*) – maximum bitrate for music, may be an empty string.
- **photoQuality** (*int*) – photo quality on scale 0 to 100.
- **photoResolution** (*str*) – maximum photo resolution, formatted as WxH (e.g. *1920x1080*).
- **videoResolution** (*str*) – maximum video resolution, formatted as WxH (e.g. *1280x720*, may be empty).
- **subtitleSize** (*int*) – subtitle size on scale 0 to 100.
- **videoQuality** (*int*) – video quality on scale 0 to 100.

static createVideo(*videoQuality*)

Returns a *MediaSettings* object, based on provided video quality value.

Parameters

videoQuality (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in this module.

Raises

BadRequest – When provided unknown video quality.

static createMusic(*bitrate*)

Returns a *MediaSettings* object, based on provided music quality value

Parameters

bitrate (*int*) – maximum bitrate for synchronized music, better use one of MUSIC_BITRATE_* values from the module

static createPhoto(*resolution*)

Returns a *MediaSettings* object, based on provided photo quality value.

Parameters

resolution (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module.

Raises

BadRequest – When provided unknown video quality.

class plexapi.sync.**Policy**(*scope*, *unwatched*, *value=0*)

Bases: object

Policy of syncing the media (how many items to sync and process watched media or not).

Variables

- **scope** (*str*) – type of limitation policy, can be *count* or *all*.
- **value** (*int*) – amount of media to sync, valid only when *scope=count*.
- **unwatched** (*bool*) – True means disallow to sync watched media.

static create(*limit=None*, *unwatched=False*)

Creates a *Policy* object for provided options and automatically sets proper *scope* value.

Parameters

- **limit** (*int*) – limit items by count.
- **unwatched** (*bool*) – if True then watched items wouldn't be synced.

Returns

Policy.

CHAPTER

TWENTYTWO

UTILS PLEXAPI.UTILS

class plexapi.utils.**SecretsFilter**(*secrets=None*)

Bases: Filter

Logging filter to hide secrets.

filter(*record*)

Determine if the specified record is to be logged.

Returns True if the record should be logged, or False otherwise. If deemed appropriate, the record may be modified in-place.

plexapi.utils.**registerPlexObject**(*cls*)

Registry of library types we may come across when parsing XML. This allows us to define a few helper functions to dynamically convert the XML into objects. See buildItem() below for an example.

plexapi.utils.**getPlexObject**(*ehash*, *default*)

Return the PlexObject class for the specified ehash. This recursively looks up the class with the highest specificity, falling back to the default class if not found.

plexapi.utils.**cast**(*func*, *value*)

Cast the specified value to the specified type (returned by func). Currently this only support str, int, float, bool. Should be extended if needed.

Parameters

- **func** (*func*) – Callback function to used cast to type (int, bool, float).
- **value** (*any*) – value to be cast and returned.

plexapi.utils.**joinArgs**(*args*)

Returns a query string (uses for HTTP URLs) where only the value is URL encoded. Example return value: '?genre=action&type=1337'.

Parameters

args (*dict*) – Arguments to include in query string.

plexapi.utils.**rget**(*obj*, *attrstr*, *default=None*, *delim='.'*)

Returns the value at the specified attrstr location within a nested tree of dicts, lists, tuples, functions, classes, etc. The lookup is done recursively for each key in attrstr (split by by the delimiter) This function is heavily influenced by the lookups used in Django templates.

Parameters

- **obj** (*any*) – Object to start the lookup in (dict, obj, list, tuple, etc).
- **attrstr** (*str*) – String to lookup (ex: 'foo.bar.baz.value')
- **default** (*any*) – Default value to return if not found.

- **delim** (*str*) – Delimiter separating keys in attrstr.

plexapi.utils.**searchType**(*libtype*)

Returns the integer value of the library string type.

Parameters

libtype (*str*) – LibType to lookup (See SEARCHTYPES)

Raises

NotFound – Unknown libtype

plexapi.utils.**reverseSearchType**(*libtype*)

Returns the string value of the library type.

Parameters

libtype (*int*) – Integer value of the library type.

Raises

NotFound – Unknown libtype

plexapi.utils.**tagType**(*tag*)

Returns the integer value of the library tag type.

Parameters

tag (*str*) – Tag to lookup (See TAGTYPES)

Raises

NotFound – Unknown tag

plexapi.utils.**reverseTagType**(*tag*)

Returns the string value of the library tag type.

Parameters

tag (*int*) – Integer value of the library tag type.

Raises

NotFound – Unknown tag

plexapi.utils.**threaded**(*callback*, *listargs*)

Returns the result of <callback> for each set of **args* in listargs. Each call to <callback> is called concurrently in their own separate threads.

Parameters

- **callback** (*func*) – Callback function to apply to each set of **args*.
- **listargs** (*list*) – List of lists; **args* to pass each thread.

plexapi.utils.**setDatetimeTimezone**(*value*)

Sets the timezone to use when converting values with `toDatetime()`.

Parameters

value (*bool, str*) –

- `False` or `None` to disable timezone (default).
- `True` or `"local"` to use the local timezone.
- A valid IANA timezone (e.g. `UTC` or `America/New_York`).

Returns

Resolved timezone object or `None` if disabled or invalid.

Return type

datetime.tzinfo

plexapi.utils.**toDatetime**(*value*, *format=None*)

Returns a datetime object from the specified value.

Parameters

- **value** (*str*) – value to return as a datetime
- **format** (*str*) – Format to pass strftime (optional; if value is a str).

plexapi.utils.**millisecondToHumanstr**(*milliseconds*)

Returns human readable time duration [D day[s],]HH:MM:SS.UUU from milliseconds.

Parameters

milliseconds (*str, int*) – time duration in milliseconds.

plexapi.utils.**toList**(*value*, *itemcast=None*, *delim=','*)

Returns a list of strings from the specified value.

Parameters

- **value** (*str*) – comma delimited string to convert to list.
- **itemcast** (*func*) – Function to cast each list item to (default str).
- **delim** (*str*) – string delimiter (optional; default ‘,’).

plexapi.utils.**downloadSessionImages**(*server*, *filename=None*, *height=150*, *width=150*, *opacity=100*, *saturation=100*)

Helper to download a bif image or thumb.url from plex.server.sessions.

Parameters

- **filename** (*str*) – default to None,
- **height** (*int*) – Height of the image.
- **width** (*int*) – width of the image.
- **opacity** (*int*) – Opacity of the resulting image (possibly deprecated).
- **saturation** (*int*) – Saturating of the resulting image.

Returns

{‘filepath’: ‘<filepath>’, ‘url’: ‘http://<url>’}, {‘<username>’: {filepath, url}}, …

Return type

{‘hellowlol’

plexapi.utils.**download**(*url*, *token*, *filename=None*, *savepath=None*, *session=None*, *chunksize=4096*, *unpack=False*, *mocked=False*, *showstatus=False*)

Helper to download a thumb, videofile or other media item. Returns the local path to the downloaded file.

Parameters

- **url** (*str*) – URL where the content be reached.
- **token** (*str*) – Plex auth token to include in headers.
- **filename** (*str*) – Filename of the downloaded file, default None.
- **savepath** (*str*) – Defaults to current working dir.
- **chunksize** (*int*) – What chunksize read/write at the time.
- **mocked** (*bool*) – Helper to do everything except write the file.
- **unpack** (*bool*) – Unpack the zip file.

- **showstatus** – Display a progressbar.

plexapi.utils.**getMyPlexAccount**(*opts=None*)

Helper function tries to get a MyPlex Account instance by checking the the following locations for a username and password. This is useful to create user-friendly command line tools. 1. command-line options (opts). 2. environment variables and config.ini 3. Prompt on the command line.

plexapi.utils.**createMyPlexDevice**(*headers=None*, *account=None*, *timeout=10*)

Helper function to create a new MyPlexDevice. Returns a new MyPlexDevice instance.

Parameters

- **headers** (*dict*) – Provide the X-Plex- headers for the new device. A unique X-Plex-Client-Identifier is required or one will be generated if not provided.
- **account** (*MyPlexAccount*) – The Plex account to create the device on.
- **timeout** (*int*) – Timeout in seconds to wait for device login.

plexapi.utils.**plexOAuth**(*headers*, *forwardUrl=None*, *timeout=120*)

Helper function for Plex OAuth login. Returns a new MyPlexAccount instance.

Parameters

- **headers** (*dict*) – Provide the X-Plex- headers for the new device. A unique X-Plex-Client-Identifier is required or one will be generated if not provided.
- **forwardUrl** (*str, optional*) – The url to redirect the client to after login.
- **timeout** (*int, optional*) – Timeout in seconds to wait for user login. Default 120 seconds.

plexapi.utils.**plexJWTAuth**(*headers=None*, *forwardUrl=None*, *timeout=120*, *keypair=(None, None)*, *scopes=None*)

Helper function for Plex JWT authentication using Plex OAuth. Returns a new MyPlexAccount instance.

Parameters

- **headers** (*dict, optional*) – Provide the X-Plex- headers for the new device. A unique X-Plex-Client-Identifier is required or one will be generated if not provided.
- **forwardUrl** (*str, optional*) – The url to redirect the client to after login.
- **timeout** (*int, optional*) – Timeout in seconds to wait for user login. Default 120 seconds.
- **keypair** (*tuple, optional*) – A tuple of the ED25519 (privateKey, publicKey) to use for signing the JWT. If not provided, a new keypair will be generated and saved to 'private.key' and 'public.key'.
- **scopes** (*list[str], optional*) – List of scopes to request in the JWT.

plexapi.utils.**choose**(*msg*, *items*, *attr*)

Command line helper to display a list of choices, asking the user to choose one of the options.

plexapi.utils.**getAgentIdentifier**(*section*, *agent*)

Return the full agent identifier from a short identifier, name, or confirm full identifier.

plexapi.utils.**iterXMLBFS**(*root*, *tag=None*)

Iterate through an XML tree using a breadth-first search. If tag is specified, only return nodes with that tag.

plexapi.utils.**toJson**(*obj*, ***kwargs*)

Convert an object to a JSON string.

Parameters

- **obj** (*object*) – The object to convert.
- ****kwargs** (*dict*) – Keyword arguments to pass to `json.dumps()`.

plexapi.utils.**sha1hash**(*guid*)

Return the SHA1 hash of a guid.

plexapi.utils.**parseXMLString**(*s: str*)

Parse an XML string and return an ElementTree object.

CHAPTER

TWENTYTHREE

VIDEO PLEXAPI.VIDEO

class plexapi.video.**Video**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexPartialObject*, *PlayedUnplayedMixin*

Base class for all video objects including *Movie*, *Show*, *Season*, *Episode*, and *Clip*.

Variables

- **addedAt** (*datetime*) – Datetime the item was added to the library.
- **art** (*str*) – URL to artwork image (/library/metadata/<ratingKey>/art/<artid>).
- **artBlurHash** (*str*) – BlurHash string for artwork image.
- **fields** (List<*Field*>) – List of field objects.
- **guid** (*str*) – Plex GUID for the movie, show, season, episode, or clip (plex://movie/5d776b59ad5437001f79c6f8).
- **images** (List<*Image*>) – List of image objects.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **lastRatedAt** (*datetime*) – Datetime the item was last rated.
- **lastViewedAt** (*datetime*) – Datetime the item was last played.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **librarySectionKey** (*str*) – *LibrarySection* key.
- **librarySectionTitle** (*str*) – *LibrarySection* title.
- **listType** (*str*) – Hardcoded as 'video' (useful for search filters).
- **ratingKey** (*int*) – Unique key identifying the item.
- **summary** (*str*) – Summary of the movie, show, season, episode, or clip.
- **thumb** (*str*) – URL to thumbnail image (/library/metadata/<ratingKey>/thumb/<thumbid>).
- **thumbBlurHash** (*str*) – BlurHash string for thumbnail image.
- **title** (*str*) – Name of the movie, show, season, episode, or clip.
- **titleSort** (*str*) – Title to use when sorting (defaults to title).
- **type** (*str*) – 'movie', 'show', 'season', 'episode', or 'clip'.
- **updatedAt** (*datetime*) – Datetime the item was updated.
- **userRating** (*float*) – Rating of the item (0.0 - 10.0) equaling (0 stars - 5 stars).
- **viewCount** (*int*) – Count of times the item was played.

url(*part*)

Returns the full url for something. Typically used for getting a specific image.

augmentation()

Returns a list of *Hub* objects. Augmentation returns hub items relating to online media sources such as Tidal Music "Track from {item}" or "Soundtrack of {item}". Plex Pass and linked Tidal account are required.

uploadSubtitles(*filepath*)

Upload a subtitle file for the video.

Parameters
filepath (*str*) – Path to subtitle file.

searchSubtitles(*language='en'*, *hearingImpaired=0*, *forced=0*)

Search for on-demand subtitles for the video. See https://support.plex.tv/articles/subtitle-search/.

Parameters

- **language** (*str, optional*) – Language code (ISO 639-1) of the subtitles to search for. Default 'en'.
- **hearingImpaired** (*int, optional*) – Search option for SDH subtitles. Default 0. (0 = Prefer non-SDH subtitles, 1 = Prefer SDH subtitles, 2 = Only show SDH subtitles, 3 = Only show non-SDH subtitles)
- **forced** (*int, optional*) – Search option for forced subtitles. Default 0. (0 = Prefer non-forced subtitles, 1 = Prefer forced subtitles, 2 = Only show forced subtitles, 3 = Only show non-forced subtitles)

Returns
List of SubtitleStream objects.

Return type
List<*SubtitleStream*>

downloadSubtitles(*subtitleStream*)

Download on-demand subtitles for the video. See https://support.plex.tv/articles/subtitle-search/.

Note: This method is asynchronous and returns immediately before subtitles are fully downloaded.

Parameters
subtitleStream (*SubtitleStream*) – Subtitle object returned from *searchSubtitles()*.

removeSubtitles(*subtitleStream=None*, *streamID=None*, *streamTitle=None*)

Remove an upload or downloaded subtitle from the video.

Note: If the subtitle file is located inside video directory it will be deleted. Files outside of video directory are not affected. Embedded subtitles cannot be removed.

Parameters

- **subtitleStream** (*SubtitleStream*, optional) – Subtitle object to remove.
- **streamID** (*int, optional*) – ID of the subtitle stream to remove.
- **streamTitle** (*str, optional*) – Title of the subtitle stream to remove.

optimize(*title=''*, *target=''*, *deviceProfile=''*, *videoQuality=None*, *locationID=-1*, *limit=None*, *unwatched=False*)

Create an optimized version of the video.

Parameters

- **title** (*str, optional*) – Title of the optimized video.
- **target** (*str, optional*) – Target quality profile: "Optimized for Mobile" ("mobile"), "Optimized for TV" ("tv"), "Original Quality" ("original"), or custom quality profile name (default "Custom: {deviceProfile}").
- **deviceProfile** (*str, optional*) – Custom quality device profile: "Android", "iOS", "Universal Mobile", "Universal TV", "Windows Phone", "Windows", "Xbox One". Required if `target` is custom.
- **videoQuality** (*int, optional*) – Index of the quality profile, one of `VIDEO_QUALITY_*` values defined in the *sync* module. Only used if `target` is custom.
- **locationID** (int or *Location*, optional) – Default -1 for "In folder with original items", otherwise a *Location* object or ID. See examples below.
- **limit** (*int, optional*) – Maximum count of items to optimize, unlimited if `None`.
- **unwatched** (*bool, optional*) – `True` to only optimized unwatched videos.

Raises

- ***BadRequest*** – Unknown quality profile target or missing deviceProfile and videoQuality.
- ***BadRequest*** – Unknown location ID.

Example

```
# Optimize for mobile using defaults
video.optimize(target="mobile")

# Optimize for Android at 10 Mbps 1080p
from plexapi.sync import VIDEO_QUALITY_10_MBPS_1080p
video.optimize(deviceProfile="Android", videoQuality=sync.VIDEO_QUALITY_10_
↪MBPS_1080p)

# Optimize for iOS at original quality in library location
from plexapi.sync import VIDEO_QUALITY_ORIGINAL
locations = plex.library.section("Movies")._locations()
video.optimize(deviceProfile="iOS", videoQuality=VIDEO_QUALITY_ORIGINAL,␣
↪locationID=locations[0])

# Optimize for tv the next 5 unwatched episodes
show.optimize(target="tv", limit=5, unwatched=True)
```

sync(*videoQuality*, *client=None*, *clientId=None*, *limit=None*, *unwatched=False*, *title=None*)

Add current video (movie, tv-show, season or episode) as sync item for specified device. See *sync()* for possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *sync* module.
- **client** (*MyPlexDevice*) – sync destination, see *sync()*.

- **clientId** (*str*) – sync destination, see *sync()*.
- **limit** (*int*) – maximum count of items to sync, unlimited if *None*.
- **unwatched** (*bool*) – if *True* watched videos wouldn't be synced.
- **title** (*str*) – descriptive title for the new *SyncItem*, if empty the value would be generated from metadata of current media.

Returns
an instance of created syncItem.

Return type
SyncItem

class plexapi.video.**Movie**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Video*, *Playable*, *MovieMixins*

Represents a single Movie.

Variables

- **TAG** (*str*) – 'Video'
- **TYPE** (*str*) – 'movie'
- **audienceRating** (*float*) – Audience rating (usually from Rotten Tomatoes).
- **audienceRatingImage** (*str*) – Key to audience rating image (rottentomatoes://image.rating.spilled).
- **chapters** (List<*Chapter*>) – List of chapter objects.
- **chapterSource** (*str*) – Chapter source (agent; media; mixed).
- **collections** (List<*Collection*>) – List of collection objects.
- **commonSenseMedia** (*CommonSenseMedia*) – Common Sense Media object.
- **contentRating** (*str*) *Content rating (PG-13; NR; TV-G)*
- **countries** (List<*Country*>) – List of country objects.
- **directors** (List<*Director*>) – List of director objects.
- **duration** (*int*) – Duration of the movie in milliseconds.
- **editionTitle** (*str*) – The edition title of the movie (e.g. Director's Cut, Extended Edition, etc.).
- **enableCreditsMarkerGeneration** (*int*) – Setting that indicates if credits markers detection is enabled. (-1 = Library default, 0 = Disabled)
- **genres** (List<*Genre*>) – List of genre objects.
- **guids** (List<*Guid*>) – List of guid objects.
- **labels** (List<*Label*>) – List of label objects.
- **languageOverride** (*str*) – Setting that indicates if a language is used to override metadata (eg. en-CA, None = Library default).
- **markers** (List<*Marker*>) – List of marker objects.
- **media** (List<*Media*>) – List of media objects.
- **originallyAvailableAt** (*datetime*) – Datetime the movie was released.
- **originalTitle** (*str*) – Original title, often the foreign title (;).

- **primaryExtraKey** (*str) Primary extra key (/library/metadata/66351*)
- **producers** (List<*Producer*>) – List of producers objects.
- **rating** (*float*) – Movie critic rating (7.9; 9.8; 8.1).
- **ratingImage** (*str*) – Key to critic rating image (rottentomatoes://image.rating.rotten).
- **ratings** (List<*Rating*>) – List of rating objects.
- **roles** (List<*Role*>) – List of role objects.
- **slug** (*str*) – The clean watch.plex.tv URL identifier for the movie.
- **similar** (List<*Similar*>) – List of Similar objects.
- **sourceURI** (*str*) – Remote server URI (server://<machineIdentifier>/com.plexapp.plugins.library) (remote playlist item only).
- **studio** (*str*) – Studio that created movie (Di Bonaventura Pictures; 21 Laps Entertainment).
- **tagline** (*str*) – Movie tag line (Back 2 Work; Who says men can't change?).
- **theme** (*str*) – URL to theme resource (/library/metadata/<ratingkey>/theme/<themeid>).
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.
- **useOriginalTitle** (*int*) – Setting that indicates if the original title is used for the movie (-1 = Library default, 0 = No, 1 = Yes).
- **viewOffset** (*int*) – View offset in milliseconds.
- **writers** (List<*Writer*>) – List of writers objects.
- **year** (*int*) – Year movie was released.

property actors

Alias to self.roles.

property locations

This does not exist in plex xml response but is added to have a common interface to get the locations of the movie.

Returns

List<str> of file paths where the movie is found on disk.

property hasCreditsMarker

Returns True if the movie has a credits marker.

property hasVoiceActivity

Returns True if any of the media has voice activity analyzed.

property hasPreviewThumbnails

Returns True if any of the media parts has generated preview (BIF) thumbnails.

reviews()

Returns a list of *Review* objects.

editions()

Returns a list of *Movie* objects for other editions of the same movie.

removeFromContinueWatching()

Remove the movie from continue watching.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.video.Show(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Video*, *ShowMixins*

Represents a single Show (including all seasons and episodes).

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'show'
- **audienceRating** (*float*) – Audience rating (TMDB or TVDB).
- **audienceRatingImage** (*str*) – Key to audience rating image (tmdb://image.rating).
- **audioLanguage** (*str*) – Setting that indicates the preferred audio language.
- **autoDeletionItemPolicyUnwatchedLibrary** (*int*) – Setting that indicates the number of unplayed episodes to keep for the show (0 = All episodes, 5 = 5 latest episodes, 3 = 3 latest episodes, 1 = 1 latest episode, -3 = Episodes added in the past 3 days, -7 = Episodes added in the past 7 days, -30 = Episodes added in the past 30 days).
- **autoDeletionItemPolicyWatchedLibrary** (*int*) – Setting that indicates if episodes are deleted after being watched for the show (0 = Never, 1 = After a day, 7 = After a week, 100 = On next refresh).
- **childCount** (*int*) – Number of seasons (including Specials) in the show.
- **collections** (List<*Collection*>) – List of collection objects.
- **commonSenseMedia** (*CommonSenseMedia*) – Common Sense Media object.
- **contentRating** (*str) Content rating (PG-13; NR; TV-G*)
- **duration** (*int*) – Typical duration of the show episodes in milliseconds.
- **enableCreditsMarkerGeneration** (*int*) – Setting that indicates if credits markers detection is enabled. (-1 = Library default, 0 = Disabled).
- **episodeSort** (*int*) – Setting that indicates how episodes are sorted for the show (-1 = Library default, 0 = Oldest first, 1 = Newest first).
- **flattenSeasons** (*int*) – Setting that indicates if seasons are set to hidden for the show (-1 = Library default, 0 = Hide, 1 = Show).
- **genres** (List<*Genre*>) – List of genre objects.
- **guids** (List<*Guid*>) – List of guid objects.
- **index** (*int*) – Plex index number for the show.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **labels** (List<*Label*>) – List of label objects.
- **languageOverride** (*str*) – Setting that indicates if a language is used to override metadata (eg. en-CA, None = Library default).
- **leafCount** (*int*) – Number of items in the show view.
- **locations** (*List<str>*) – List of folder paths where the show is found on disk.
- **network** (*str*) – The network that distributed the show.
- **originallyAvailableAt** (*datetime*) – Datetime the show was released.

- **originalTitle** (*str*) – The original title of the show.
- **rating** (*float*) – Show rating (7.9; 9.8; 8.1).
- **ratings** (List<*Rating*>) – List of rating objects.
- **roles** (List<*Role*>) – List of role objects.
- **seasonCount** (*int*) – Number of seasons (excluding Specials) in the show.
- **showOrdering** (*str*) – Setting that indicates the episode ordering for the show (None = Library default, tmdbAiring = The Movie Database (Aired), aired = TheTVDB (Aired), dvd = TheTVDB (DVD), absolute = TheTVDB (Absolute)).
- **similar** (List<*Similar*>) – List of Similar objects.
- **slug** (*str*) – The clean watch.plex.tv URL identifier for the show.
- **studio** (*str*) – Studio that created show (Di Bonaventura Pictures; 21 Laps Entertainment).
- **subtitleLanguage** (*str*) – Setting that indicates the preferred subtitle language.
- **subtitleMode** (*int*) – Setting that indicates the auto-select subtitle mode. (-1 = Account default, 0 = Manually selected, 1 = Shown with foreign audio, 2 = Always enabled).
- **tagline** (*str*) – Show tag line.
- **theme** (*str*) – URL to theme resource (/library/metadata/<ratingkey>/theme/<themeid>).
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.
- **useOriginalTitle** (*int*) – Setting that indicates if the original title is used for the show (-1 = Library default, 0 = No, 1 = Yes).
- **viewedLeafCount** (*int*) – Number of items marked as played in the show view.
- **year** (*int*) – Year the show was released.

property actors

Alias to self.roles.

property isPlayed

Returns True if the show is fully played.

onDeck()

Returns show's On Deck *Video* object or *None*. If show is unwatched, return will likely be the first episode.

season(*title=None*, *season=None*)

Returns the season with the specified title or number.

Parameters

- **title** (*str*) – Title of the season to return.
- **season** (*int*) – Season number (default: None; required if title not specified).

Raises

BadRequest – If title or season parameter is missing.

seasons(***kwargs*)

Returns a list of *Season* objects in the show.

episode(*title=None*, *season=None*, *episode=None*)

Find a episode using a title or season and episode.

Parameters

- **title** (*str*) – Title of the episode to return
- **season** (*int*) – Season number (default: None; required if title not specified).
- **episode** (*int*) – Episode number (default: None; required if title not specified).

Raises

BadRequest – If title or season and episode parameters are missing.

episodes(***kwargs*)

Returns a list of *Episode* objects in the show.

get(*title=None*, *season=None*, *episode=None*)

Alias to *episode()*.

watched()

Returns list of watched *Episode* objects.

unwatched()

Returns list of unwatched *Episode* objects.

download(*savepath=None*, *keep_original_name=False*, *subfolders=False*, ***kwargs*)

Download all episodes from the show. See *download()* for details.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **keep_original_name** (*bool*) – True to keep the original filename otherwise a friendlier filename is generated.
- **subfolders** (*bool*) – True to separate episodes in to season folders.
- ****kwargs** – Additional options passed into `getStreamURL()`.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.video.**Season**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Video*, *SeasonMixins*

Represents a single Season.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'season'
- **audienceRating** (*float*) – Audience rating.
- **audioLanguage** (*str*) – Setting that indicates the preferred audio language.
- **collections** (List<*Collection*>) – List of collection objects.
- **guids** (List<*Guid*>) – List of guid objects.
- **index** (*int*) – Season number.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **labels** (List<*Label*>) – List of label objects.

- **leafCount** (*int*) – Number of items in the season view.
- **parentGuid** (*str*) – Plex GUID for the show (plex://show/5d9c086fe9d5a1001f4d9fe6).
- **parentIndex** (*int*) – Plex index number for the show.
- **parentKey** (*str*) – API URL of the show (/library/metadata/<parentRatingKey>).
- **parentRatingKey** (*int*) – Unique key identifying the show.
- **parentSlug** (*str*) – The clean watch.plex.tv URL identifier for the show.
- **parentStudio** (*str*) – Studio that created show.
- **parentTheme** (*str*) – URL to show theme resource (/library/metadata/<parentRatingkey>/theme/<themeid>).
- **parentThumb** (*str*) – URL to show thumbnail image (/library/metadata/<parentRatingKey>/thumb/<thumbid>).
- **parentTitle** (*str*) – Name of the show for the season.
- **rating** (*float*) – Season rating (7.9; 9.8; 8.1).
- **ratings** (List<*Rating*>) – List of rating objects.
- **subtitleLanguage** (*str*) – Setting that indicates the preferred subtitle language.
- **subtitleMode** (*int*) – Setting that indicates the auto-select subtitle mode. (-1 = Series default, 0 = Manually selected, 1 = Shown with foreign audio, 2 = Always enabled).
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.
- **viewedLeafCount** (*int*) – Number of items marked as played in the season view.
- **year** (*int*) – Year the season was released.

property isPlayed

Returns True if the season is fully played.

property seasonNumber

Returns the season number.

onDeck()

Returns season's On Deck *Video* object or *None*. Will only return a match if the show's On Deck episode is in this season.

episode(*title=None*, *episode=None*)

Returns the episode with the given title or number.

Parameters

- **title** (*str*) – Title of the episode to return.
- **episode** (*int*) – Episode number (default: None; required if title not specified).

Raises

BadRequest – If title or episode parameter is missing.

episodes(***kwargs*)

Returns a list of *Episode* objects in the season.

get(*title=None*, *episode=None*)

Alias to *episode()*.

show()

Return the season's *Show*.

watched()

Returns list of watched *Episode* objects.

unwatched()

Returns list of unwatched *Episode* objects.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)

Download all episodes from the season. See *download()* for details.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **keep_original_name** (*bool*) – True to keep the original filename otherwise a friendlier filename is generated.
- ****kwargs** – Additional options passed into getStreamURL().

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.video.**Episode**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Video*, *Playable*, *EpisodeMixins*

Represents a single Episode.

Variables

- **TAG** (*str*) – 'Video'
- **TYPE** (*str*) – 'episode'
- **audienceRating** (*float*) – Audience rating (TMDB or TVDB).
- **audienceRatingImage** (*str*) – Key to audience rating image (tmdb://image.rating).
- **chapters** (List<*Chapter*>) – List of chapter objects.
- **chapterSource** (*str*) – Chapter source (agent; media; mixed).
- **collections** (List<*Collection*>) – List of collection objects.
- **contentRating** (*str*) *Content rating (PG-13; NR; TV-G*)
- **directors** (List<*Director*>) – List of director objects.
- **duration** (*int*) – Duration of the episode in milliseconds.
- **grandparentArt** (*str*) – URL to show artwork (/library/metadata/<grandparentRatingKey>/art/<artid>).
- **grandparentGuid** (*str*) – Plex GUID for the show (plex://show/5d9c086fe9d5a1001f4d9fe6).
- **grandparentKey** (*str*) – API URL of the show (/library/metadata/<grandparentRatingKey>).
- **grandparentRatingKey** (*int*) – Unique key identifying the show.
- **grandparentSlug** (*str*) – The clean watch.plex.tv URL identifier for the show.
- **grandparentTheme** (*str*) – URL to show theme resource (/library/metadata/<grandparentRatingkey>/theme/<themeid>).

- **grandparentThumb** (*str*) – URL to show thumbnail image (/library/metadata/<grandparentRatingKey>/thumb/<thumbid>).
- **grandparentTitle** (*str*) – Name of the show for the episode.
- **guids** (List<*Guid*>) – List of guid objects.
- **index** (*int*) – Episode number.
- **labels** (List<*Label*>) – List of label objects.
- **markers** (List<*Marker*>) – List of marker objects.
- **media** (List<*Media*>) – List of media objects.
- **originallyAvailableAt** (*datetime*) – Datetime the episode was released.
- **parentGuid** (*str*) – Plex GUID for the season (plex://season/5d9c09e42df347001e3c2a72).
- **parentIndex** (*int*) – Season number of episode.
- **parentKey** (*str*) – API URL of the season (/library/metadata/<parentRatingKey>).
- **parentRatingKey** (*int*) – Unique key identifying the season.
- **parentThumb** (*str*) – URL to season thumbnail image (/library/metadata/<parentRatingKey>/thumb/<thumbid>).
- **parentTitle** (*str*) – Name of the season for the episode.
- **parentYear** (*int*) – Year the season was released.
- **producers** (List<*Producer*>) – List of producers objects.
- **rating** (*float*) – Episode rating (7.9; 9.8; 8.1).
- **ratings** (List<*Rating*>) – List of rating objects.
- **roles** (List<*Role*>) – List of role objects.
- **skipParent** (*bool*) – True if the show's seasons are set to hidden.
- **sourceURI** (*str*) – Remote server URI (server://<machineIdentifier>/com.plexapp.plugins.library) (remote playlist item only).
- **ultraBlurColors** (*UltraBlurColors*) – Ultra blur color object.
- **viewOffset** (*int*) – View offset in milliseconds.
- **writers** (List<*Writer*>) – List of writers objects.
- **year** (*int*) – Year the episode was released.

property parentKey

Returns the parentKey. Refer to the Episode attributes.

property parentRatingKey

Returns the parentRatingKey. Refer to the Episode attributes.

property parentThumb

Returns the parentThumb. Refer to the Episode attributes.

property actors

Alias to self.roles.

property locations

This does not exist in plex xml response but is added to have a common interface to get the locations of the episode.

Returns

List<str> of file paths where the episode is found on disk.

property episodeNumber

Returns the episode number.

property seasonNumber

Returns the episode's season number.

property seasonEpisode

Returns the s00e00 string containing the season and episode numbers.

property hasCommercialMarker

Returns True if the episode has a commercial marker.

property hasIntroMarker

Returns True if the episode has an intro marker.

property hasCreditsMarker

Returns True if the episode has a credits marker.

property hasVoiceActivity

Returns True if any of the media has voice activity analyzed.

property hasPreviewThumbnails

Returns True if any of the media parts has generated preview (BIF) thumbnails.

season()

" Return the episode's *Season*.

show()

" Return the episode's *Show*.

removeFromContinueWatching()

Remove the movie from continue watching.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.video.**Clip**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Video*, *Playable*, *ClipMixins*

Represents a single Clip.

Variables

- **TAG** (*str*) – 'Video'
- **TYPE** (*str*) – 'clip'
- **duration** (*int*) – Duration of the clip in milliseconds.
- **extraType** (*int*) – Unknown.
- **index** (*int*) – Plex index number for the clip.
- **media** (List<*Media*>) – List of media objects.
- **originallyAvailableAt** (*datetime*) – Datetime the clip was released.

- **skipDetails** (*int*) – Unknown.
- **subtype** (*str*) – Type of clip (trailer, behindTheScenes, sceneOrSample, etc.).
- **thumbAspectRatio** (*str*) – Aspect ratio of the thumbnail image.
- **viewOffset** (*int*) – View offset in milliseconds.
- **year** (*int*) – Year clip was released.

property locations

This does not exist in plex xml response but is added to have a common interface to get the locations of the clip.

Returns

List<str> of file paths where the clip is found on disk.

property metadataDirectory

Returns the Plex Media Server data directory where the metadata is stored.

class plexapi.video.**Extra**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *Clip*

Represents a single Extra (trailer, behindTheScenes, etc).

class plexapi.video.**MovieSession**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexSession*, *Movie*

Represents a single Movie session loaded from *sessions()*.

class plexapi.video.**EpisodeSession**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexSession*, *Episode*

Represents a single Episode session loaded from *sessions()*.

class plexapi.video.**ClipSession**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexSession*, *Clip*

Represents a single Clip session loaded from *sessions()*.

class plexapi.video.**MovieHistory**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexHistory*, *Movie*

Represents a single Movie history entry loaded from *history()*.

class plexapi.video.**EpisodeHistory**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexHistory*, *Episode*

Represents a single Episode history entry loaded from *history()*.

class plexapi.video.**ClipHistory**(*server*, *data*, *initpath=None*, *parent=None*)

Bases: *PlexHistory*, *Clip*

Represents a single Clip history entry loaded from *history()*.

CHAPTER

TWENTYFOUR

USAGE & CONTRIBUTIONS

- Source is available on the Github Project Page.
- Contributors to python-plexapi own their own contributions and may distribute that code under the BSD license.

PYTHON MODULE INDEX

p

INDEX

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W