Python PlexAPI Documentation

M.Shepanski

May 01, 2020

Overview

CHAPTER 1

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.
- Listen in on all Plex Server notifications.

1.2 Installation & Documentation

```
pip install plexapi
```

Documentation can be found at Read the Docs.

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, but 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 watched.
plex.library.section('TV Shows').get('Game of Thrones').markWatched()
```

```
# 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('%s (%s)' % (video.title, video.TYPE))
```

```
# Example 6: List all movies directed by the same person as Elephants Dream.
movies = plex.library.section('Movies')
die_hard = movies.get('Elephants Dream')
director = die_hard.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 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 veriable *SHARED_USERNAME*. It uses official docker image to create a proper instance.

For skipping the docker and reuse a existing server use

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

1.6 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 offerered 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 2

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

[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
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).

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.

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())*).

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).

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 3

Alert plexapi.alert

class plexapi.alert.**AlertListener**(*server*, *callback=None*)

Bases: threading.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): ...`

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 *plexapi.server.PlexServer.startAlertListener()* from a PlexServer instance.

CHAPTER 4

Audio plexapi.audio

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

Bases: *plexapi.base.PlexPartialObject*

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

Variables

- **addedAt** (*datetime*) – Datetime this item was added to the library.
- **index** (*sting*) – Index Number (often the track number).
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **lastViewedAt** (*datetime*) – Datetime item was last accessed.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **listType** (*str*) – Hardcoded as 'audio' (useful for search filters).
- **ratingKey** (*int*) – Unique key identifying this item.
- **summary** (*str*) – Summary of the artist, track, or album.
- **thumb** (*str*) – URL to thumbnail image.
- **title** (*str*) – Artist, Album or Track title. (Jason Mraz, We Sing, Lucky, etc.)
- **titleSort** (*str*) – Title to use when sorting (defaults to title).
- **type** (*str*) – 'artist', 'album', or 'track'.
- **updatedAt** (*datatime*) – Datetime this item was updated.
- **viewCount** (*int*) – Count of times this item was accessed.

thumbUrl

Return url to for the thumbnail image.

artUrl

Return the first art url starting on the most specific for that item.

url (*part*)
Returns the full URL for this audio item. Typically used for getting a specific track.

sync (*bitrate*, *client=None*, *clientId=None*, *limit=None*, *title=None*)
Add current audio (artist, album or track) as sync item for specified device. See *plexapi.myplex.MyPlexAccount.sync()* for possible exceptions.

Parameters

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

Returns an instance of created syncItem.

Return type *plexapi.sync.SyncItem*

class plexapi.audio.**Artist** (*server*, *data*, *initpath=None*)
Bases: *plexapi.audio.Audio*

Represents a single audio artist.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'artist'
- **art** (*str*) – Artist artwork (/library/metadata/<ratingkey>/art/<artid>)
- **countries** (*list*) – List of *Country* objects this artist respresents.
- **genres** (*list*) – List of *Genre* objects this artist respresents.
- **guid** (*str*) – Unknown (unique ID; com.plexapp.agents.plexmusic://gracenote/artist/05517B8701668D28?lang=e
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **location** (*str*) – Filepath this artist is found on disk.
- **similar** (*list*) – List of *Similar* artists.

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 this artist.

track (*title*)
Returns the *Track* that matches the specified title.

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

tracks (***kwargs*)
Returns a list of *Track* objects by this artist.

get(*title*)
Alias of *track()*.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)
Downloads all tracks for this artist to the specified location.

Parameters

- **savepath** (*str*) – Title of the track to return.
- **keep_original_name** (*bool*) – Set True to keep the original filename as stored in the Plex server. False will create a new filename with the format "<Atrist> - <Album> <Track>".
- **kwargs** (*dict*) – If specified, a `getStreamURL()` will be returned and the additional arguments passed in will be sent to that function. If kwargs is not specified, the media items will be downloaded and saved to disk.

class plexapi.audio.**Album**(*server*, *data*, *initpath=None*)
Bases: *plexapi.audio.Audio*

Represents a single audio album.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'album'
- **art** (*str*) – Album artwork (/library/metadata/<ratingkey>/art/<artid>)
- **genres** (*list*) – List of *Genre* objects this album respresents.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **originallyAvailableAt** (*datetime*) – Datetime this album was released.
- **parentKey** (*str*) – API URL of this artist.
- **parentRatingKey** (*int*) – Unique key identifying artist.
- **parentThumb** (*str*) – URL to artist thumbnail image.
- **parentTitle** (*str*) – Name of the artist for this album.
- **studio** (*str*) – Studio that released this album.
- **year** (*int*) – Year this album was released.

track(*title*)
Returns the *Track* that matches the specified title.

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

tracks(***kwargs*)
Returns a list of *Track* objects in this album.

get(*title*)
Alias of *track()*.

artist()
Return *Artist()* of this album.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)
Downloads all tracks for this artist to the specified location.

Parameters

- **savepath** (*str*) – Title of the track to return.
- **keep_original_name** (*bool*) – Set True to keep the original filename as stored in the Plex server. False will create a new filename with the format "<Atrist> - <Album> <Track>".
- **kwargs** (*dict*) – If specified, a `getStreamURL()` will be returned and the additional arguments passed in will be sent to that function. If kwargs is not specified, the media items will be downloaded and saved to disk.

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

Bases: *plexapi.audio.Audio*, *plexapi.base.Playable*

Represents a single audio track.

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'track'
- **art** (*str*) – Track artwork (/library/metadata/<ratingkey>/art/<artid>)
- **chapterSource** (*TYPE*) – Unknown
- **duration** (*int*) – Length of this album in seconds.
- **grandparentArt** (*str*) – Album artist artwork.
- **grandparentKey** (*str*) – Album artist API URL.
- **grandparentRatingKey** (*str*) – Unique key identifying album artist.
- **grandparentThumb** (*str*) – URL to album artist thumbnail image.
- **grandparentTitle** (*str*) – Name of the album artist for this track.
- **guid** (*str*) – Unknown (unique ID).
- **media** (*list*) – List of *Media* objects for this track.
- **moods** (*list*) – List of *Mood* objects for this track.
- **originalTitle** (*str*) – Track artist.
- **parentIndex** (*int*) – Album index.
- **parentKey** (*str*) – Album API URL.
- **parentRatingKey** (*int*) – Unique key identifying album.
- **parentThumb** (*str*) – URL to album thumbnail image.
- **parentTitle** (*str*) – Name of the album for this track.
- **primaryExtraKey** (*str*) – Unknown
- **ratingCount** (*int*) – Unknown
- **userRating** (*float*) – Rating of this track (0.0 - 10.0) equaling (0 stars - 5 stars)
- **viewOffset** (*int*) – Unknown
- **year** (*int*) – Year this track was released.
- **sessionKey** (*int*) – Session Key (active sessions only).
- **usernames** (*str*) – Username of person playing this track (active sessions only).
- **player** (*str*) – *PlexClient* for playing track (active sessions only).

- **transcodeSessions** (*None*) – *TranscodeSession* for playing track (active sessions only).

album()
Return this track's *Album*.

artist()
Return this track's *Artist*.

CHAPTER 5

Base plexapi.base

class plexapi.base.**PlexObject**(*server*, *data*, *initpath=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).

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 attribute filters on the items to fetch. For example, passing in viewCount=0 will only return matching items. Filtering is done before the Python objects are built to help keep things speedy. Note: Because some attribute names are already used as arguments to this function, such as 'tag', you may still reference the attr tag byappending an underscore. For example, passing in _tag='foobar' will return all items where tag='foobar'. Also Note: Case very much matters when specifying kwargs – Optionally, operators can be specified by append it to the end of the attribute name for more complex lookups. For example, passing in viewCount__gte=0 will return all items where viewCount >= 0. Available operations include:

- __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 insensative value contains specified arg.
- __iendswith: Case insensative value ends with specified arg.
- __iexact: Case insensative value matches specified arg.
- __in: Value is in a specified list or tuple.
- __iregex: Case insensative value matches the specified regular expression.
- __istartswith: Case insensative 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.

fetchItems(*ekey*, *cls=None*, *container_start=None*, *container_size=None*, ***kwargs*)
Load the specified key to find and build all items with the specified tag and attrs. See *fetchItem()* for more details on how this is used.

Parameters

- **container_start** (*None, int*) – offset to get a subset of the data
- **container_size** (*None, int*) – How many items in data

findItems(*data*, *cls=None*, *initpath=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.

firstAttr(**attrs*)
Return the first attribute in attrs that is not None.

reload(*key=None*)
Reload the data for this object from self.key.

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

isFullObject()
Retruns 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 contain.

isPartialObject()
Returns True if this is not a full object.

edit(***kwargs*)
Edit an object.

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

Example

{'type': 1, 'id': movie.ratingKey, 'collection[0].tag.tag': 'Super', 'collection.locked': 0}

addCollection(*collections*)
Add a collection(s).

Parameters **collections** (*list*) – list of strings

removeCollection(*collections*)
Remove a collection(s).

addLabel(*labels*)
Add a label(s).

removeLabel(*labels*)
Remove a label(s).

addGenre(*genres*)
Add a genre(s).

removeGenre(*genres*)
Remove a genre(s).

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=9999999*, *mindate=None*)
Get Play History for a media item. :param maxresults: Only return the specified number of results (optional). :type maxresults: int :param mindate: Min datetime to return results from. :type mindate: datetime

posters()
Returns list of available poster objects. *Poster*.

uploadPoster(*url=None*, *filepath=None*)
Upload poster from url or filepath. *Poster* to *Video*.

setPoster(*poster*)
Set . *Poster* to *Video*

arts()
Returns list of available art objects. *Poster*.

uploadArt(*url=None*, *filepath=None*)
Upload art from url or filepath. *Poster* to *Video*.

setArt(*art*)
Set *Poster* to *Video*

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*)
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()

class plexapi.base.**Playable**
Bases: object

This is a general place 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

- **sessionKey** (*int*) – Active session key.
- **usernames** (*str*) – Username of the person playing this item (for active sessions).
- **players** (*PlexClient*) – Client objects playing this item (for active sessions).
- **session** (*Session*) – Session object, for a playing media file.
- **transcodeSessions** (*TranscodeSession*) – Transcode Session object if item is being transcoded (None otherwise).
- **viewedAt** (*datetime*) – Datetime item was last viewed (history).
- **playlistItemID** (*int*) – Playlist item ID (only populated for *Playlist* items).

isFullObject()
Retruns 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 contain.

getStreamURL(***params*)
Returns a stream url that may be used by external applications such as VLC.

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

Raises ***plexapi.exceptions.Unsupported*** – When the item doesn't support fetching a stream URL.

iterParts()
Iterates over the parts of this media item.

split()
Split a duplicate.

merge(*ratingKeys*)
Merge duplicate items.

unmatch()
Unmatch a media file.

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

Parameters

- **savepath** (*str*) – Title of the track to return.
- **keep_original_name** (*bool*) – Set True to keep the original filename as stored in the Plex server. False will create a new filename with the format "<Artist> - <Album> <Track>".
- **kwargs** (*dict*) – If specified, a `getStreamURL()` will be returned and the additional arguments passed in will be sent to that function. If kwargs is not specified, the media items will be downloaded and saved to disk.

stop(*reason=''*)
Stop playback for a media item.

updateProgress(*time*, *state='stopped'*)
Set the watched progress for this video.

Note that setting the time to 0 will not work. Use *markWatched* or *markUnwatched* 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.**Release**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

CHAPTER 6

Client plexapi.client

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

Bases: *plexapi.base.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 dirrectly to this client.
- **token** (*str*) – X-Plex-Token used for authenication (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 authenication
- **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 ***plexapi.exceptions.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 *plexapi.exceptions.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.

Raises *plexapi.exceptions.Unsupported* – When no PlexServer specified in this object.

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

Raises ***plexapi.exceptions.Unsupported*** – When no PlexServer specified in this object.

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).

timeline(*wait=1*)
Poll the current timeline and return the XML response.

isPlayingMedia(*includePaused=False*)
Returns True if any media is currently playing.

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

CHAPTER 7

Config plexapi.config

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

Bases: backports.configparser.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 8

Exceptions plexapi.exceptions

exception plexapi.exceptions.**PlexApiException**
Bases: exceptions.Exception

Base class for all PlexAPI exceptions.

exception plexapi.exceptions.**BadRequest**
Bases: *plexapi.exceptions.PlexApiException*

An invalid request, generally a user error.

exception plexapi.exceptions.**NotFound**
Bases: *plexapi.exceptions.PlexApiException*

Request media item or device is not found.

exception plexapi.exceptions.**UnknownType**
Bases: *plexapi.exceptions.PlexApiException*

Unknown library type.

exception plexapi.exceptions.**Unsupported**
Bases: *plexapi.exceptions.PlexApiException*

Unsupported client request.

exception plexapi.exceptions.**Unauthorized**
Bases: *plexapi.exceptions.BadRequest*

Invalid username/password or token.

CHAPTER 9

Config 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**
Base class to discover GDM services.

scan(*scan_for_clients=False*)
Scan the network.

all()
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 10

Library plexapi.library

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

Bases: *plexapi.base.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=None*)

Returns the *LibrarySection* that matches the specified title.

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

sectionByID(*sectionID*)

Returns the *LibrarySection* that matches the specified sectionID.

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

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 havent 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'*, **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*) – /path/to/files
- **language** (*str*) – Two letter language fx en
- **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, 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 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
- **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 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=9999999*, *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

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

Bases: *plexapi.base.PlexObject*

Base class for a single library section.

Variables

- **ALLOWED_FILTERS** (*tuple*) – ()
- **ALLOWED_SORT** (*tuple*) – ()
- **BOOLEAN_FILTERS** (*tuple<str>*) – ('unwatched', 'duplicate')
- **server** (*PlexServer*) – Server this client is connected to.
- **initpath** (*str*) – Path requested when building this object.
- **agent** (*str*) – Unknown (com.plexapp.agents.imdb, etc)

- **allowSync** (*bool*) – True if you allow syncing content from this section.
- **art** (*str*) – Wallpaper artwork used to respresent this section.
- **composite** (*str*) – Composit image used to represent this section.
- **createdAt** (*datetime*) – Datetime this library section was created.
- **filters** (*str*) – Unknown
- **key** (*str*) – Key (or ID) of this library section.
- **language** (*str*) – Language represented in this section (en, xn, etc).
- **locations** (*str*) – Paths on disk where section content is stored.
- **refreshing** (*str*) – 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 this section.
- **title** (*str*) – Title of this section.
- **type** (*str*) – Type of content section represents (movie, artist, photo, show).
- **updatedAt** (*datetime*) – Datetime this library section was last updated.
- **uuid** (*str*) – Unique id for this section (32258d7c-3e6c-4ac5-98ad-bad7a3b78c63)
- **totalSize** (*int*) – Total number of item in the library

fetchItems(*ekey*, *cls=None*, *container_start=None*, *container_size=None*, ***kwargs*)
Load the specified key to find and build all items with the specified tag and attrs. See *fetchItem()* for more details on how this is used.

Parameters

- **container_start** (*None, int*) – offset to get a subset of the data
- **container_size** (*None, int*) – How many items in data

delete()
Delete a library section.

reload(*key=None*)
Reload the data for this object from self.key.

edit(*agent=None*, ***kwargs*)
Edit a library (Note: agent is required). See *Library* for example usage.

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

get(*title*)
Returns the media item with the specified title.

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

all(*sort=None*, ***kwargs*)
Returns a list of media from this library section.

Parameters **sort** (*string*) – The sort string

agents()
Returns a list of available *:class:'~plexapi.media.Agent* for this library section.

settings()
Returns a list of all library settings.

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

recentlyAdded(*maxresults=50*)
Returns a list of media items recently added from this library section.

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

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()
Scan this section for new media.

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.

listChoices(*category*, *libtype=None*, ***kwargs*)
Returns a list of *FilterChoice* objects for the specified category and libtype. kwargs can be any of the same kwargs in `plexapi.library.LibrarySection.search()` to help narrow down the choices to only those that matter in your current context.

Parameters

- **category** (*str*) – Category to list choices for (genre, contentRating, etc).
- **libtype** (*int*) – Library type of item filter.
- ****kwargs** (*dict*) – Additional kwargs to narrow down the choices.

Raises ***plexapi.exceptions.BadRequest*** – Cannot include kwarg equal to specified category.

search(*title=None*, *sort=None*, *maxresults=None*, *libtype=None*, *container_start=0*, *container_size=100*, ***kwargs*)
Search the library. The http requests will be batched in container_size. 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

- **title** (*str*) – General string query to search for (optional).
- **sort** (*str*) – column:dir; column can be any of {addedAt, originallyAvailableAt, lastViewedAt, titleSort, rating, mediaHeight, duration}. dir can be asc or desc (optional).
- **maxresults** (*int*) – Only return the specified number of results (optional).
- **libtype** (*str*) – Filter results to a spcifiec libtype (movie, show, episode, artist, album, track; optional).

- **container_start** (*int*) – default 0
- **container_size** (*int*) – default X_PLEX_CONTAINER_SIZE in your config file.
- ****kwargs** (*dict*) – Any of the available filters for the current library section. Partial string matches allowed. Multiple matches OR together. Negative filtering also possible, just add an exclamation mark to the end of filter name, e.g. *resolution!=1x1*.
  - unwatched: Display or hide unwatched content (True, False). [all]
  - duplicate: Display or hide duplicate items (True, False). [movie]
  - actor: List of actors to search ([actor_or_id, ...]). [movie]
  - collection: List of collections to search within ([collection_or_id, ...]). [all]
  - contentRating: List of content ratings to search within ([rating_or_key, ...]). [movie,tv]
  - country: List of countries to search within ([country_or_key, ...]). [movie,music]
  - decade: List of decades to search within ([yyy0, ...]). [movie]
  - director: List of directors to search ([director_or_id, ...]). [movie]
  - genre: List Genres to search within ([genere_or_id, ...]). [all]
  - network: List of TV networks to search within ([resolution_or_key, ...]). [tv]
  - resolution: List of video resolutions to search within ([resolution_or_key, ...]). [movie]
  - studio: List of studios to search within ([studio_or_key, ...]). [music]
  - year: List of years to search within ([yyyy, ...]). [all]

Raises ***plexapi.exceptions.BadRequest*** – when applying unknown filter

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 *plexapi.myplex.MyPlexAccount.sync()* for possible exceptions.

Parameters

- **policy** (*plexapi.sync.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** (*plexapi.sync.MediaSettings*) – Transcoding settings used for the media, generated automatically when method called on specific LibrarySection object.
- **client** (*plexapi.myplex.MyPlexDevice*) – sync destination, see *plexapi.myplex.MyPlexAccount.sync()*.
- **clientId** (*str*) – sync destination, see *plexapi.myplex.MyPlexAccount.sync()*.
- **title** (*str*) – descriptive title for the new *plexapi.sync.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 *plexapi.sync.SyncItem*

Raises ***plexapi.exceptions.BadRequest*** – when the library is not allowed to sync

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=9999999*, *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

class plexapi.library.**MovieSection**(*server*, *data*, *initpath=None*)
Bases: *plexapi.library.LibrarySection*

Represents a *LibrarySection* section containing movies.

Variables

- **ALLOWED_FILTERS** (*list<str>*) – List of allowed search filters. ('unwatched', 'duplicate', 'year', 'decade', 'genre', 'contentRating', 'collection', 'director', 'actor', 'country', 'studio', 'resolution', 'guid', 'label')
- **ALLOWED_SORT** (*list<str>*) – List of allowed sorting keys. ('addedAt', 'originallyAvailableAt', 'lastViewedAt', 'titleSort', 'rating', 'mediaHeight', 'duration')
- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'movie'

collection(***kwargs*)
Returns a list of collections from this library section.

sync(*videoQuality*, *limit=None*, *unwatched=False*, ***kwargs*)
Add current Movie library section as sync item for specified device. See description of *plexapi.library.LibrarySection.search()* for details about filtering / sorting and *plexapi.library.LibrarySection.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 *plexapi.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 *plexapi.sync.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*)

Bases: *plexapi.library.LibrarySection*

Represents a *LibrarySection* section containing tv shows.

Variables

- **ALLOWED_FILTERS** (*list<str>*) – List of allowed search filters. ('unwatched', 'year', 'genre', 'contentRating', 'network', 'collection', 'guid', 'label')
- **ALLOWED_SORT** (*list<str>*) – List of allowed sorting keys. ('addedAt', 'lastViewedAt', 'originallyAvailableAt', 'titleSort', 'rating', 'unwatched')
- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'show'

searchShows(***kwargs*)

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

searchEpisodes(***kwargs*)

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

recentlyAdded(*libtype='episode'*, *maxresults=50*)

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

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

collection(***kwargs*)

Returns a list of collections from this library section.

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

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

Bases: *plexapi.library.LibrarySection*

Represents a *LibrarySection* section containing music artists.

Variables

- **ALLOWED_FILTERS** (*list<str>*) – List of allowed search filters. ('genre', 'country', 'collection')
- **ALLOWED_SORT** (*list<str>*) – List of allowed sorting keys. ('addedAt', 'lastViewedAt', 'viewCount', 'titleSort')
- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'artist'

albums()

Returns a list of *Album* objects 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.

collection(***kwargs*)

Returns a list of collections from this library section.

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

Add current Music library section as sync item for specified device. See description of *plexapi.library.LibrarySection.search()* for details about filtering / sorting and *plexapi.library.LibrarySection.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 *plexapi.sync*.
- **limit** (*int*) – maximum count of tracks to sync, unlimited if *None*.

Returns an instance of created syncItem.

Return type *plexapi.sync.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')
```

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

Bases: *plexapi.library.LibrarySection*

Represents a *LibrarySection* section containing photos.

Variables

- **ALLOWED_FILTERS** (*list<str>*) – List of allowed search filters. ('all', 'iso', 'make', 'lens', 'aperture', 'exposure', 'device', 'resolution')
- **ALLOWED_SORT** (*list<str>*) – List of allowed sorting keys. ('addedAt')
- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'photo'

searchAlbums(*title*, ***kwargs*)

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

searchPhotos(*title*, ***kwargs*)

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

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

Add current Music library section as sync item for specified device. See description of *plexapi.library.LibrarySection.search()* for details about filtering / sorting and *plexapi.library.LibrarySection.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 *plexapi.sync*.
- **limit** (*int*) – maximum count of tracks to sync, unlimited if *None*.

Returns an instance of created syncItem.

Return type *plexapi.sync.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)
```

(continues on next page)

(continued from previous page)

```
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.**FilterChoice**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

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

Variables

- **TAG** (*str*) – 'Directory'
- **server** (*PlexServer*) – PlexServer this client is connected to.
- **initpath** (*str*) – Relative path requested when retrieving specified *data* (optional).
- **fastKey** (*str*) – API path to quickly list all items in this filter (/library/sections/<section>/all?genre=<key>)
- **key** (*str*) – Short key (id) of this filter option (used ad <key> in fastKey above).
- **thumb** (*str*) – Thumbnail used to represent this filter option.
- **title** (*str*) – Human readable name for this filter option.
- **type** (*str*) – Filter type (genre, contentRating, etc).

class plexapi.library.**Hub**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

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

Variables

- **TAG** (*str*) – 'Hub'
- **hubIdentifier** (*str*) – Unknown.
- **size** (*int*) – Number of items found.
- **title** (*str*) – Title of this Hub.
- **type** (*str*) – Type of items in the Hub.
- **items** (*str*) – List of items in the Hub.

class plexapi.library.**Collections**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

modeUpdate(*mode=None*)
Update Collection Mode

Parameters **mode** – default (Library default) hide (Hide Collection) hideItems (Hide Items in this Collection) showItems (Show this Collection and its Items)

Example

collection = 'plexapi.library.Collections' collection.updateMode(mode="hide")

sortUpdate(*sort=None*)
Update Collection Sorting

Parameters **sort** – realease (Order Collection by realease dates) alpha (Order Collection Alphabetically)

Example

colleciton = 'plexapi.library.Collections' collection.updateSort(mode="alpha")

posters()
Returns list of available poster objects. *Poster*.

uploadPoster(*url=None*, *filepath=None*)
Upload poster from url or filepath. *Poster* to *Video*.

setPoster(*poster*)
Set . *Poster* to *Video*

arts()
Returns list of available art objects. *Poster*.

uploadArt(*url=None*, *filepath=None*)
Upload art from url or filepath. *Poster* to *Video*.

setArt(*art*)
Set *Poster* to *Video*

CHAPTER 11

Media plexapi.media

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

Bases: *plexapi.base.PlexObject*

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

Variables

- **TAG** (*str*) – 'Media'
- **server** (*PlexServer*) – PlexServer object this is from.
- **initpath** (*str*) – Relative path requested when retrieving specified data.
- **video** (*str*) – Video this media belongs to.
- **aspectRatio** (*float*) – Aspect ratio of the video (ex: 2.35).
- **audioChannels** (*int*) – Number of audio channels for this video (ex: 6).
- **audioCodec** (*str*) – Audio codec used within the video (ex: ac3).
- **bitrate** (*int*) – Bitrate of the video (ex: 1624)
- **container** (*str*) – Container this video is in (ex: avi).
- **duration** (*int*) – Length of the video in milliseconds (ex: 6990483).
- **height** (*int*) – Height of the video in pixels (ex: 256).
- **id** (*int*) – Plex ID of this media item (ex: 46184).
- **has64bitOffsets** (*bool*) – True if video has 64 bit offsets (?).
- **optimizedForStreaming** (*bool*) – True if video is optimized for streaming.
- **target** (*str*) – Media version target name.
- **title** (*str*) – Media version title.
- **videoCodec** (*str*) – Video codec used within the video (ex: ac3).

- **videoFrameRate** (*str*) – Video frame rate (ex: 24p).
- **videoResolution** (*str*) – Video resolution (ex: sd).
- **videoProfile** (*str*) – Video profile (ex: high).
- **width** (*int*) – Width of the video in pixels (ex: 608).
- **parts** (list<*MediaPart*>) – List of MediaParts in this video.

class plexapi.media.**MediaPart**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

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

Variables

- **TAG** (*str*) – 'Part'
- **server** (*PlexServer*) – PlexServer object this is from.
- **initpath** (*str*) – Relative path requested when retrieving specified data.
- **media** (*Media*) – Media object this part belongs to.
- **container** (*str*) – Container type of this media part (ex: avi).
- **duration** (*int*) – Length of this media part in milliseconds.
- **file** (*str*) – Path to this file on disk (ex: /media/Movies/Cars.(2006)/Cars.cd2.avi)
- **id** (*int*) – Unique ID of this media part.
- **indexes** (*str, None*) – None or SD.
- **key** (*str*) – Key used to access this media part (ex: /library/parts/46618/1389985872/file.avi).
- **size** (*int*) – Size of this file in bytes (ex: 733884416).
- **streams** (list<*MediaPartStream*>) – List of streams in this media part.
- **exists** (*bool*) – Determine if file exists
- **accessible** (*bool*) – Determine if file is accessible

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.

setDefaultAudioStream(*stream*)
Set the default *AudioStream* for this MediaPart.

Parameters **stream** (*AudioStream*) – AudioStream to set as default

setDefaultSubtitleStream(*stream*)
Set the default *SubtitleStream* for this MediaPart.

Parameters **stream** (*SubtitleStream*) – SubtitleStream to set as default.

resetDefaultSubtitleStream()
Set default subtitle of this MediaPart to 'none'.

class plexapi.media.**MediaPartStream**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

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

Variables

- **server** (*PlexServer*) – PlexServer object this is from.
- **initpath** (*str*) – Relative path requested when retrieving specified data.
- **part** (*MediaPart*) – Media part this stream belongs to.
- **codec** (*str*) – Codec of this stream (ex: srt, ac3, mpeg4).
- **codecID** (*str*) – Codec ID (ex: XVID).
- **id** (*int*) – Unique stream ID on this server.
- **index** (*int*) – Unknown
- **language** (*str*) – Stream language (ex: English,).
- **languageCode** (*str*) – Ascii code for language (ex: eng, tha).
- **selected** (*bool*) – True if this stream is selected.
- **streamType** (*int*) – Stream type (1=:class:~*plexapi.media.VideoStream*, 2=:class:~*plexapi.media.AudioStream*, 3=:class:~*plexapi.media.SubtitleStream*).
- **type** (*int*) – Alias for streamType.

static parse(*server*, *data*, *initpath*)
Factory method returns a new MediaPartStream from xml data.

class plexapi.media.**VideoStream**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaPartStream*

Respresents a video stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 1
- **bitDepth** (*int*) – Bit depth (ex: 8).
- **bitrate** (*int*) – Bitrate (ex: 1169)
- **cabac** (*int*) – Unknown
- **chromaSubsampling** (*str*) – Chroma Subsampling (ex: 4:2:0).
- **colorSpace** (*str*) – Unknown
- **duration** (*int*) – Duration of video stream in milliseconds.
- **frameRate** (*float*) – Frame rate (ex: 23.976)
- **frameRateMode** (*str*) – Unknown
- **hasScallingMatrix** (*bool*) – True if video stream has a scaling matrix.
- **height** (*int*) – Height of video stream.
- **level** (*int*) – Videl stream level (?).
- **profile** (*str*) – Video stream profile (ex: asp).

- **refFrames** (*int*) – Unknown
- **scanType** (*str*) – Video stream scan type (ex: progressive).
- **title** (*str*) – Title of this video stream.
- **width** (*int*) – Width of video stream.

class plexapi.media.**AudioStream**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaPartStream*

Respresents a audio stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 2
- **audioChannelLayout** (*str*) – Audio channel layout (ex: 5.1(side)).
- **bitDepth** (*int*) – Bit depth (ex: 16).
- **bitrate** (*int*) – Audio bitrate (ex: 448).
- **bitrateMode** (*str*) – Bitrate mode (ex: cbr).
- **channels** (*int*) – number of channels in this stream (ex: 6).
- **dialogNorm** (*int*) – Unknown (ex: -27).
- **duration** (*int*) – Duration of audio stream in milliseconds.
- **samplingRate** (*int*) – Sampling rate (ex: xxx)
- **title** (*str*) – Title of this audio stream.

class plexapi.media.**SubtitleStream**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaPartStream*

Respresents a audio stream within a *MediaPart*.

Variables

- **TAG** (*str*) – 'Stream'
- **STREAMTYPE** (*int*) – 3
- **forced** (*bool*) – True if this is a forced subtitle
- **format** (*str*) – Subtitle format (ex: srt).
- **key** (*str*) – Key of this subtitle stream (ex: /library/streams/212284).
- **title** (*str*) – Title of this subtitle stream.

class plexapi.media.**Session**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a current session.

class plexapi.media.**TranscodeSession**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a current transcode session.

Variables

- **TAG** (*str*) – 'TranscodeSession'

- **TODO** – Document this.

class plexapi.media.**TranscodeJob**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.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*)
Bases: *plexapi.base.PlexObject*

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

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*)
Bases: *plexapi.base.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*)
Bases: *plexapi.base.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

- **server** (*PlexServer*) – Server this client is connected to.
- **id** (*id*) – Tag ID (This seems meaningless except to use it as a unique id).
- **role** (*str*) – Unknown
- **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).
- **<Hub_Search_Attributes>** – Attributes only applicable in search results from PlexServer *search()*. They provide details of which library section the tag was found as well as the url to dig deeper into the results.
  - key (str): API URL to dig deeper into this tag (ex: /library/sections/1/all?actor=9081).
  - librarySectionID (int): Section ID this tag was generated from.
  - librarySectionTitle (str): Library section title this tag was found.

- librarySectionType (str): Media type of the library section this tag was found.
- tagType (int): Tag type ID.
- thumb (str): URL to thumbnail image.

items(**args*, ***kwargs*)
Return the list of items within this tag. This function is only applicable in search results from PlexServer *search()*.

class plexapi.media.**Collection**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Collection media tag.

Variables

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

class plexapi.media.**Label**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single label media tag.

Variables

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

class plexapi.media.**Tag**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single tag media tag.

Variables

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

class plexapi.media.**Country**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Country media tag.

Variables

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

class plexapi.media.**Director**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Director media tag.

Variables

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

class plexapi.media.**Genre**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Genre media tag.

Variables

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

class plexapi.media.**Mood**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Mood media tag.

Variables

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

class plexapi.media.**Poster**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a Poster.

Variables **TAG** (*str*) – 'Photo'

class plexapi.media.**Producer**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Producer media tag.

Variables

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

class plexapi.media.**Role**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

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

Variables

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

class plexapi.media.**Similar**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Similar media tag.

Variables

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

class plexapi.media.**Writer**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.MediaTag*

Represents a single Writer media tag.

Variables

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

class plexapi.media.**Chapter**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a single Writer media tag.

Variables **TAG** (*str*) – 'Chapter'

class plexapi.media.**Field**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a single Field.

Variables **TAG** (*str*) – 'Field'

class plexapi.media.**SearchResult**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a single SearchResult.

Variables **TAG** (*str*) – 'SearchResult'

class plexapi.media.**Agent**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexObject*

Represents a single Agent.

Variables **TAG** (*str*) – 'Agent'

class plexapi.media.**AgentMediaType**(*server*, *data*, *initpath=None*)
Bases: *plexapi.media.Agent*

CHAPTER 12

MyPlex plexapi.myplex

class plexapi.myplex.**MyPlexAccount**(*username=None*, *password=None*, *token=None*, *session=None*, *timeout=None*)

Bases: *plexapi.base.PlexObject*

MyPlex account and profile information. This object represents the data found Account on the myplex.tv servers at the url https://plex.tv/users/account. 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*) – Your MyPlex username.
- **password** (*str*) – Your MyPlex 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).

Variables

- **SIGNIN** (*str*) – 'https://plex.tv/users/sign_in.xml'
- **key** (*str*) – 'https://plex.tv/users/account'
- **authenticationToken** (*str*) – Unknown.
- **certificateVersion** (*str*) – Unknown.
- **cloudSyncDevice** (*str*) – Unknown.
- **email** (*str*) – Your current Plex email address.
- **entitlements** (*List<str>*) – List of devices your allowed to use with this account.
- **guest** (*bool*) – Unknown.
- **home** (*bool*) – Unknown.

- **homeSize** (*int*) – Unknown.
- **id** (*str*) – Your Plex account ID.
- **locale** (*str*) – Your Plex locale
- **mailing_list_status** (*str*) – Your current mailing list status.
- **maxHomeSize** (*int*) – Unknown.
- **queueEmail** (*str*) – Email address to add items to your *Watch Later* queue.
- **queueUid** (*str*) – Unknown.
- **restricted** (*bool*) – Unknown.
- **roles** – (List<str>) Lit of account roles. Plexpass membership listed here.
- **scrobbleTypes** (*str*) – Description
- **secure** (*bool*) – Description
- **subscriptionActive** (*bool*) – True if your subsctiption is active.
- **subscriptionFeatures** – (List<str>) List of features allowed on your subscription.
- **subscriptionPlan** (*str*) – Name of subscription plan.
- **subscriptionStatus** (*str*) – String representation of *subscriptionActive*.
- **thumb** (*str*) – URL of your account thumbnail.
- **title** (*str*) – Unknown. - Looks like an alias for *username*.
- **username** (*str*) – Your account username.
- **uuid** (*str*) – Unknown.
- **_token** (*str*) – Token used to access this client.
- **_session** (*obj*) – Requests session object used to access this client.

device(*name*)
Returns the *MyPlexDevice* that matches the name specified.

Parameters **name** (*str*) – Name 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 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** (*str*) – MyPlexUser, username, email of the user to be added.
- **server** (*PlexServer*) – PlexServer object or machineIdentifier containing the library sections to share.

- **sections** (*[Section]*) – Library sections, names or ids to be shared (default None). [Section] must be defined in order to update shared 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']}

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** (*str*) – MyPlexUser, username, email of the user to be added.
- **server** (*PlexServer*) – PlexServer object or machineIdentifier containing the library sections to share.
- **sections** (*[Section]*) – Library sections, names or ids to be shared (default None shares all 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']}

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** (*str*) – MyPlexUser, username, email of the user to be added.
- **server** (*PlexServer*) – PlexServer object or machineIdentifier containing the library sections to share.
- **sections** (*[Section]*) – Library sections, names or ids to be shared (default None shares all 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']}

removeFriend(*user*)
Remove the specified user from all sharing.

Parameters **user** (*str*) – MyPlexUser, username, email of the user to be added.

removeHomeUser(*user*)
Remove the specified managed user from home.

Parameters **user** (*str*) – MyPlexUser, username, email of the user to be removed from home.

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** (*str*) – MyPlexUser, username, email of the user to be added.
- **server** (*PlexServer*) – PlexServer object or machineIdentifier containing the library sections to share.
- **sections** – ([Section]): Library sections, names or ids to be shared (default None). [Section] must be defined in order to update shared sections.
- **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 email or username specified.

Parameters **username** (*str*) – Username, email or id of the user to return.

users()
Returns a list of all *MyPlexUser* objects connected to your account. This includes both friends and pending invites. You can reference the user.friend to distinguish between the two.

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 *plexapi.sync.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. *plexapi.video.Video.sync()*, *plexapi.audio.Audio.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** (*plexapi.sync.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 *plexapi.sync.SyncItem*

Raises

- ***plexapi.exceptions.BadRequest*** – when client with provided clientId wasn't found.
- ***plexapi.exceptions.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=9999999*, *mindate=None*)
Get Play History for all library sections on all servers 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

videoOnDemand()
Returns a list of VOD Hub items *Hub*

webShows()
Returns a list of Webshow Hub items *Hub*

news()
Returns a list of News Hub items *Hub*

podcasts()
Returns a list of Podcasts Hub items *Hub*

tidal()
Returns a list of tidal Hub items *Hub*

class plexapi.myplex.**MyPlexUser**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.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.
- **thumb** (*str*) – Link to the users avatar.
- **title** (*str*) – Seems to be an aliad for username.
- **username** (*str*) – User's username.
- **servers** – Servers shared between user and friend

server(*name*)
Returns the *MyPlexServerShare* that matches the name specified.

Parameters **name** (*str*) – Name of the server to return.

history(*maxresults=9999999*, *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.**Section**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.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/{serverId}

Variables

- **TAG** (*str*) – section

- **id** (*int*) – shared section id
- **sectionKey** (*str*) – what key we use for this section
- **title** (*str*) – Title of the section
- **sectionId** (*str*) – shared section id
- **type** (*str*) – movie, tvshow, artist
- **shared** (*bool*) – If this section is shared with the user

history(*maxresults=9999999*, *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*)
Bases: *plexapi.base.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*)
Bases: *plexapi.base.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/resources?includeHttps=1&includeRelay=1

Variables

- **TAG** (*str*) – 'Device'
- **key** (*str*) – 'https://plex.tv/api/resources?includeHttps=1&includeRelay=1'

- **accessToken** (*str*) – This resources accesstoken.
- **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).
- **home** (*bool*) – Unknown
- **lastSeenAt** (*datetime*) – Timestamp this resource last connected.
- **name** (*str*) – Descriptive name of this resource.
- **owned** (*bool*) – True if this resource is one of your own (you logged into it).
- **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.)
- **synced** (*bool*) – Unknown (possibly True if the resource has synced content?)

connect(*ssl=None*, *timeout=None*)

Returns a new `PlexServer` or `PlexClient` object. Often times there is more than one address specified for a server or client. This function will prioritize local connections before remote and HTTPS before HTTP. 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** (*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.

Raises ***plexapi.exceptions.NotFound*** – When unable to connect to any addresses for this resource.

class plexapi.myplex.**ResourceConnection**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.PlexObject*

Represents a Resource Connection object found within the `MyPlexResource` objects.

Variables

- **TAG** (*str*) – 'Connection'
- **address** (*str*) – Local IP address
- **httpuri** (*str*) – Full local address
- **local** (*bool*) – True if local
- **port** (*int*) – 32400
- **protocol** (*str*) – HTTP or HTTPS
- **uri** (*str*) – External address

class plexapi.myplex.**MyPlexDevice**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.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 ***plexapi.exceptions.NotFound*** – When unable to connect to any addresses for this device.

delete()

Remove this device from your account.

syncItems()

Returns an instance of *plexapi.sync.SyncList* for current device.

Raises ***plexapi.exceptions.BadRequest*** – when the device doesn't provides *sync-target*.

class plexapi.myplex.**MyPlexPinLogin**(*session=None*, *requestTimeout=None*)

Bases: object

MyPlex PIN login class which supports getting the four character PIN which the user must enter on https://plex.tv/link to authenticate the client and provide an access token to create a *MyPlexAccount* instance. 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 *plexapi.myplex.MyPlexPinLogin.checkLogin()*.
- The threaded approach expects the developer to call *plexapi.myplex.MyPlexPinLogin.run()* and then at a later time call *plexapi.myplex.MyPlexPinLogin.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 *plexapi.myplex.MyPlexPinLogin.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 PMS
- **requestTimeout** (*int*) – timeout in seconds on initial connect to plex.tv (default config.TIMEOUT).

Variables

- **PINS** (*str*) – 'https://plex.tv/pins.xml'
- **CHECKPINS** (*str*) – 'https://plex.tv/pins/{pinid}.xml'
- **POLLINTERVAL** (*int*) – 1
- **finished** (*bool*) – Whether the pin login has finished or not.
- **expired** (*bool*) – Whether the pin login has expired or not.
- **token** (*str*) – Token retrieved through the pin login.
- **pin** (*str*) – Pin to use for the login on https://plex.tv/link.

run(*callback=None*, *timeout=None*)

Starts the thread which monitors the PIN login state. :param callback: Callback called with the received authentication token (optional). :type callback: Callable[str] :param timeout: Timeout in seconds waiting for the PIN login to succeed (optional). :type timeout: int

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. :param callback: Callback called with the received authentication token (optional). :type callback: Callable[str] :param timeout: Timeout in seconds waiting for the PIN login to succeed (optional). :type timeout: int

Returns *True* if the PIN login succeeded or *False* otherwise.

stop()

Stops the thread monitoring the PIN login state.

checkLogin()
Returns *True* if the PIN login has succeeded.

CHAPTER 13

Photo plexapi.photo

class plexapi.photo.**Photoalbum**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.PlexPartialObject*

Represents a photoalbum (collection of photos).

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'photo'
- **addedAt** (*datetime*) – Datetime this item was added to the library.
- **art** (*str*) – Photo art (/library/metadata/<ratingkey>/art/<artid>)
- **composite** (*str*) – Unknown
- **guid** (*str*) – Unknown (unique ID)
- **index** (*sting*) – Index number of this album.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **listType** (*str*) – Hardcoded as 'photo' (useful for search filters).
- **ratingKey** (*int*) – Unique key identifying this item.
- **summary** (*str*) – Summary of the photoalbum.
- **thumb** (*str*) – URL to thumbnail image.
- **title** (*str*) – Photoalbum title. (Trip to Disney World)
- **type** (*str*) – Unknown
- **updatedAt** (*datatime*) – Datetime this item was updated.

albums(***kwargs*)

Returns a list of *Photoalbum* objects in this album.

album(*title*)
Returns the *Photoalbum* that matches the specified title.

photos(***kwargs*)
Returns a list of *Photo* objects in this album.

photo(*title*)
Returns the *Photo* that matches the specified title.

class plexapi.photo.**Photo**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.PlexPartialObject*

Represents a single photo.

Variables

- **TAG** (*str*) – 'Photo'
- **TYPE** (*str*) – 'photo'
- **addedAt** (*datetime*) – Datetime this item was added to the library.
- **index** (*sting*) – Index number of this photo.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **listType** (*str*) – Hardcoded as 'photo' (useful for search filters).
- **media** (*TYPE*) – Unknown
- **originallyAvailableAt** (*datetime*) – Datetime this photo was added to Plex.
- **parentKey** (*str*) – Photoalbum API URL.
- **parentRatingKey** (*int*) – Unique key identifying the photoalbum.
- **ratingKey** (*int*) – Unique key identifying this item.
- **summary** (*str*) – Summary of the photo.
- **thumb** (*str*) – URL to thumbnail image.
- **title** (*str*) – Photo title.
- **type** (*str*) – Unknown
- **updatedAt** (*datatime*) – Datetime this item was updated.
- **year** (*int*) – Year this photo was taken.

photoalbum()
Return this photo's *Photoalbum*.

section()
Returns the *LibrarySection* this item belongs to.

sync(*resolution*, *client=None*, *clientId=None*, *limit=None*, *title=None*)
Add current photo as sync item for specified device. See *plexapi.myplex.MyPlexAccount.sync()* for possible exceptions.

Parameters

- **resolution** (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module *plexapi.sync*.
- **client** (*plexapi.myplex.MyPlexDevice*) – sync destination, see *plexapi.myplex.MyPlexAccount.sync()*.

- **clientId** (*str*) – sync destination, see *plexapi.myplex.MyPlexAccount.sync()*.
- **limit** (*int*) – maximum count of items to sync, unlimited if *None*.
- **title** (*str*) – descriptive title for the new *plexapi.sync.SyncItem*, if empty the value would be generated from metadata of current photo.

Returns an instance of created syncItem.

Return type *plexapi.sync.SyncItem*

CHAPTER 14

Playlist plexapi.playlist

class plexapi.playlist.**Playlist**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.PlexPartialObject*, *plexapi.base.Playable*

Represents a single Playlist object. # TODO: Document attributes

items()

Returns a list of all items in the playlist.

addItems(*items*)

Add items to a playlist.

removeItem(*item*)

Remove a file from a playlist.

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

Move a to a new position in playlist.

edit(*title=None*, *summary=None*)

Edit playlist.

delete()

Delete playlist.

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

Create a playqueue from this playlist.

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

Create a playlist.

Parameters

- **server** (*PlexServer*) – Server your connected to.
- **title** (*str*) – Title of the playlist.
- **items** (*Iterable*) – Iterable of objects that should be in the playlist.
- **section** (*LibrarySection*, str) –

- **limit** (*int*) – default None.
- **smart** (*bool*) – default False.
- ****kwargs** (*dict*) – is passed to the filters. For a example see the search method.

Returns an instance of created Playlist.

Return type *plexapi.playlist.Playlist*

copyToUser(*user*)
Copy playlist to another user account.

sync(*videoQuality=None*, *photoResolution=None*, *audioBitrate=None*, *client=None*, *clientId=None*, *limit=None*, *unwatched=False*, *title=None*)
Add current playlist as sync item for specified device. See *plexapi.myplex.MyPlexAccount.sync()* for possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *plexapi.sync* module. Used only when playlist contains video.
- **photoResolution** (*str*) – maximum allowed resolution for synchronized photos, see PHOTO_QUALITY_* values in the module *plexapi.sync*. Used only when playlist contains photos.
- **audioBitrate** (*int*) – maximum bitrate for synchronized music, better use one of MUSIC_BITRATE_* values from the module *plexapi.sync*. Used only when playlist contains audio.
- **client** (*plexapi.myplex.MyPlexDevice*) – sync destination, see *plexapi.myplex.MyPlexAccount.sync()*.
- **clientId** (*str*) – sync destination, see *plexapi.myplex.MyPlexAccount.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 *plexapi.sync.SyncItem*, if empty the value would be generated from metadata of current photo.

Raises

- ***plexapi.exceptions.BadRequest*** – when playlist is not allowed to sync.
- ***plexapi.exceptions.Unsupported*** – when playlist content is unsupported.

Returns an instance of created syncItem.

Return type *plexapi.sync.SyncItem*

posters()
Returns list of available poster objects. *Poster*.

uploadPoster(*url=None*, *filepath=None*)
Upload poster from url or filepath. *Poster* to *Video*.

setPoster(*poster*)
Set . *Poster* to *Video*

arts()
Returns list of available art objects. *Poster*.

uploadArt(*url=None*, *filepath=None*)
Upload art from url or filepath. *Poster* to *Video*.

setArt(*art*)
Set *Poster* to *Video*

CHAPTER 15

Playqueue plexapi.playqueue

class plexapi.playqueue.**PlayQueue**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.PlexObject*

Control a PlayQueue.

Variables

- **key** (*str*) – This is only added to support playMedia
- **identifier** (*str*) – com.plexapp.plugins.library
- **initpath** (*str*) – Relative url where data was grabbed from.
- **items** (*list*) – List of *Media* or class:*~plexapi.playlist.Playlist*
- **mediaTagPrefix** (*str*) – Fx /system/bundle/media/flags/
- **mediaTagVersion** (*str*) – Fx 1485957738
- **playQueueID** (*str*) – a id for the playqueue
- **playQueueSelectedItemID** (*str*) – playQueueSelectedItemID
- **playQueueSelectedItemOffset** (*str*) – playQueueSelectedItemOffset
- **playQueueSelectedMetadataItemID** (*<type 'str'>*) – 7
- **playQueueShuffled** (*bool*) – True if shuffled
- **playQueueSourceURI** (*str*) – Fx library://150425c9-0d99-4242-821e-e5ab81cd2221/item//library/metadata/7
- **playQueueTotalCount** (*str*) – How many items in the play queue.
- **playQueueVersion** (*str*) – What version the playqueue is.
- **server** (*PlexServer*) – Server you are connected to.
- **size** (*str*) – Seems to be a alias for playQueueTotalCount.

classmethod create(*server*, *item*, *shuffle=0*, *repeat=0*, *includeChapters=1*, *includeRelated=1*)

Create and returns a new *PlayQueue*.

Paramaters: server (*PlexServer*): Server you are connected to. item (*Media* or class:~*plexapi.playlist.Playlist*): A media or Playlist. shuffle (int, optional): Start the playqueue shuffled. repeat (int, optional): Start the playqueue shuffled. includeChapters (int, optional): include Chapters. includeRelated (int, optional): include Related.

CHAPTER 16

Server plexapi.server

class plexapi.server.**PlexServer**(*baseurl=None*, *token=None*, *session=None*, *timeout=None*)

Bases: *plexapi.base.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 PMS
- **timeout** (*int*) – timeout in seconds on initial connect to 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.

library
Library to browse or search your media.

settings
Returns a list of all server settings.

account()
Returns the *Account* object this server belongs to.

agents(*mediaType=None*)
Returns the *:class:'~plexapi.media.Agent* objects this server has available.

createToken(*type='delegation'*, *scope='all'*)
Create a temp access token for the server.

systemAccounts()
Returns the `SystemAccounts` objects this server contains.

myPlexAccount()
Returns a *MyPlexAccount* object using the same token to access this server. If you are not the owner of this PlexServer you're likley to recieve an authentication error calling this.

clients()
Returns list of all *PlexClient* objects connected to server.

client(*name*)
Returns the *PlexClient* that matches the specified name.

Parameters **name** (*str*) – Name of the client to return.

Raises ***plexapi.exceptions.NotFound*** – Unknown client name

createPlaylist(*title*, *items=None*, *section=None*, *limit=None*, *smart=None*, ***kwargs*)
Creates and returns a new *Playlist*.

Parameters

- **title** (*str*) – Title of the playlist to be created.
- **items** (*list<Media>*) – List of media items to include in the playlist.

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.playerque.PlayQueue.create*.

downloadDatabases(*savepath=None*, *unpack=False*)
Download databases.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **unpack** (*bool*) – Unpack the zip file.

downloadLogs(*savepath=None*, *unpack=False*)
Download server logs.

Parameters

- **savepath** (*str*) – Defaults to current working dir.
- **unpack** (*bool*) – Unpack the zip file.

check_for_update(*force=True*, *download=False*)
Returns a *Release* object containing release info.

Parameters

- **force** (*bool*) – Force server to check for new releases
- **download** (*bool*) – Download if a update is available.

isLatest()
Check if the installed version of PMS is the latest.

installUpdate()
Install the newest version of Plex Media Server.

history(*maxresults=9999999*, *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()
Returns a list of all *Playlist* objects saved on the server.

playlist(*title*)
Returns the Playlist that matches the specified title.

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

Raises ***plexapi.exceptions.NotFound*** – Invalid playlist title

optimizedItems(*removeAll=None*)
Returns list of all *Optimized* objects connected to server.

optimizedItem(*optimizedID*)
Returns single queued optimized item Video object. Allows for using optimized item ID to connect back to source item.

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*, *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*)
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*) – Optionally limit your search to the specified media type.
- **limit** (*int*) – Optionally limit to the specified number of results per Hub.

sessions()
Returns a list of all active session (currently playing) media objects.

startAlertListener(*callback=None*)
Creates a websocket connection to the Plex Server to optionally recieve notifications. These often include messages from Plex about media scans as well as updates to currently running Transcode Sessions.

NOTE: You need websocket-client installed in order to use this feature. >> pip install websocket-client

Parameters **callback** (*func*) – Callback function to call on recieved messages.

Raises **plexapi.exception.Unsupported** – Websocket-client not installed.

transcodeImage(*media*, *height*, *width*, *opacity=100*, *saturation=100*)
Returns the URL for a transcoded image from the specified media object. Returns None if no media specified (needed if user tries to pass thumb or art directly).

Parameters

- **height** (*int*) – Height to transcode the image to.
- **width** (*int*) – Width to transcode the image to.
- **opacity** (*int*) – Opacity of the resulting image (possibly deprecated).
- **saturation** (*int*) – Saturating of the resulting image.

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'.

class plexapi.server.**Account**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.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

- **authToken** (*str*) – Plex authentication token to access the server.
- **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.**SystemAccount**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.PlexObject*

Minimal api to list system accounts.

CHAPTER 17

Settings plexapi.settings

class plexapi.settings.**Settings**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.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 settnig changes to the *PlexServer*. This performs a full reload() of Settings after complete.

class plexapi.settings.**Setting**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.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 valis values for this setting.

set(*value*)
Set a new value for this setitng. NOTE: You must call plex.settings.save() for before any changes to setting values are persisted to the *PlexServer*.

toUrl()
Helper for urls

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)

butlerTaskGenerateAutoTags (bool) Analyze and tag photos. (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)**
- **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 18

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: *plexapi.base.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 *plexapi.myplex.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*)
Bases: *plexapi.base.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=''*)

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* | *str*) – unknown, usually equals to 0, may be empty string.
- **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 ***plexapi.exceptions.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 ***plexapi.exceptions.BadRequest*** –

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 19

Utils plexapi.utils

class plexapi.utils.**SecretsFilter**(*secrets=None*)
Bases: logging.Filter

Logging filter to hide secrets.

filter(*record*)
Determine if the specified record is to be logged.

Is the specified record to be logged? Returns 0 for no, nonzero for yes. 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 convery the XML into objects. See buildItem() below for an example.

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*) – Calback 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 nexted tree of dicts, lists, tuples, functions, classes, etc. The lookup is done recursivley 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 (movie, show, season, episode, artist, album, track, collection)

Raises ***plexapi.exceptions.NotFound*** – Unknown libtype

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.**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.**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=4024*, *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 evertything except write the file.
- **unpack** (*bool*) – Unpack the zip file.
- **showstatus** – Display a progressbar.

plexapi.utils.**tag_helper**(*tag*, *items*, *locked=True*, *remove=False*)
Simple tag helper for editing a object.

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.**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.

CHAPTER 20

Video plexapi.video

class plexapi.video.**Video**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.PlexPartialObject*

Base class for all video objects including *Movie*, *Show*, *Season*, *Episode*.

Variables

- **addedAt** (*datetime*) – Datetime this item was added to the library.
- **key** (*str*) – API URL (/library/metadata/<ratingkey>).
- **lastViewedAt** (*datetime*) – Datetime item was last accessed.
- **librarySectionID** (*int*) – *LibrarySection* ID.
- **listType** (*str*) – Hardcoded as 'audio' (useful for search filters).
- **ratingKey** (*int*) – Unique key identifying this item.
- **summary** (*str*) – Summary of the artist, track, or album.
- **thumb** (*str*) – URL to thumbnail image.
- **title** (*str*) – Artist, Album or Track title. (Jason Mraz, We Sing, Lucky, etc.)
- **titleSort** (*str*) – Title to use when sorting (defaults to title).
- **type** (*str*) – 'artist', 'album', or 'track'.
- **updatedAt** (*datatime*) – Datetime this item was updated.
- **viewCount** (*int*) – Count of times this item was accessed.

isWatched

Returns True if this video is watched.

thumbUrl

Return the first first thumbnail url starting on the most specific thumbnail for that item.

artUrl

Return the first first art url starting on the most specific for that item.

url(*part*)
Returns the full url for something. Typically used for getting a specific image.

markWatched()
Mark video as watched.

markUnwatched()
Mark video unwatched.

rate(*rate*)
Rate video.

subtitleStreams()
Returns a list of *SubtitleStream* objects for all MediaParts.

uploadSubtitles(*filepath*)
Upload Subtitle file for video.

removeSubtitles(*streamID=None*, *streamTitle=None*)
Remove Subtitle from movie's subtitles listing.

Note: If subtitle file is located inside video directory it will bbe deleted. Files outside of video directory are not effected.

optimize(*title=None*, *target=''*, *targetTagID=None*, *locationID=-1*, *policyScope='all'*, *policyValue=''*, *policyUnwatched=0*, *videoQuality=None*, *deviceProfile=None*)
Optimize item

locationID (int): -1 in folder with orginal items 2 library path

target (str): custom quality name. if none provided use "Custom: {deviceProfile}"

targetTagID (int): Default quality settings 1 Mobile 2 TV 3 Original Quality

deviceProfile (str): Android, IOS, Universal TV, Universal Mobile, Windows Phone, Windows, Xbox One

Example

Optimize for Mobile item.optimize(targetTagID="Mobile") or item.optimize(targetTagID=1")

Optimize for Android 10 MBPS 1080p item.optimize(deviceProfile="Android", videoQuality=10)

Optimize for IOS Original Quality item.optimize(deviceProfile="IOS", videoQuality=-1)

- see sync.py VIDEO_QUALITIES for additional information for using videoQuality

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 *plexapi.myplex.MyPlexAccount.sync()* for possible exceptions.

Parameters

- **videoQuality** (*int*) – idx of quality of the video, one of VIDEO_QUALITY_* values defined in *plexapi.sync* module.
- **client** (*plexapi.myplex.MyPlexDevice*) – sync destination, see *plexapi.myplex.MyPlexAccount.sync()*.
- **clientId** (*str*) – sync destination, see *plexapi.myplex.MyPlexAccount.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 *plexapi.sync.SyncItem*, if empty the value would be generated from metadata of current media.

Returns an instance of created syncItem.

Return type *plexapi.sync.SyncItem*

class plexapi.video.**Movie**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.Playable*, *plexapi.video.Video*

Represents a single Movie.

Variables

- **TAG** (*str*) – 'Video'
- **TYPE** (*str*) – 'movie'
- **art** (*str*) – Key to movie artwork (/library/metadata/<ratingkey>/art/<artid>)
- **audienceRating** (*float*) – Audience rating (usually from Rotten Tomatoes).
- **audienceRatingImage** (*str*) – Key to audience rating image (rottentomatoes://image.rating.spilled)
- **chapterSource** (*str*) – Chapter source (agent; media; mixed).
- **contentRating** (*str) Content rating (PG-13; NR; TV-G*) –
- **duration** (*int*) – Duration of movie in milliseconds.
- **guid** – Plex GUID (com.plexapp.agents.imdb://tt4302938?lang=en).
- **originalTitle** (*str*) – Original title, often the foreign title (;).
- **originallyAvailableAt** (*datetime*) – Datetime movie was released.
- **primaryExtraKey** (*str) Primary extra key (/library/metadata/66351*) –
- **rating** (*float*) – Movie rating (7.9; 9.8; 8.1).
- **ratingImage** (*str*) – Key to rating image (rottentomatoes://image.rating.rotten).
- **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?).
- **userRating** (*float*) – User rating (2.0; 8.0).
- **viewOffset** (*int*) – View offset in milliseconds.
- **year** (*int*) – Year movie was released.
- **collections** (List<*Collection*>) – List of collections this media belongs.
- **countries** (List<*Country*>) – List of countries objects.
- **directors** (List<*Director*>) – List of director objects.
- **fields** (List<*Field*>) – List of field objects.
- **genres** (List<*Genre*>) – List of genre objects.
- **media** (List<*Media*>) – List of media objects.
- **producers** (List<*Producer*>) – List of producers objects.

- **roles** (List<*Role*>) – List of role objects.
- **writers** (List<*Writer*>) – List of writers objects.
- **chapters** (List<*Chapter*>) – List of Chapter objects.
- **similar** (List<*Similar*>) – List of Similar objects.

actors
Alias to self.roles.

locations
This does not exist in plex xml response but is added to have a common interface to get the location of the Movie/Show/Episode

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)
Download video files to specified directory.

Parameters

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

class plexapi.video.**Show**(*server*, *data*, *initpath=None*)
Bases: *plexapi.video.Video*

Represents a single Show (including all seasons and episodes).

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'show'
- **art** (*str*) – Key to show artwork (/library/metadata/<ratingkey>/art/<artid>)
- **banner** (*str*) – Key to banner artwork (/library/metadata/<ratingkey>/art/<artid>)
- **childCount** (*int*) – Unknown.
- **contentRating**(*str*) *Content rating (PG-13; NR; TV-G)* –
- **collections** (List<*Collection*>) – List of collections this media belongs.
- **duration** (*int*) – Duration of show in milliseconds.
- **guid** (*str*) – Plex GUID (com.plexapp.agents.imdb://tt4302938?lang=en).
- **index** (*int*) – Plex index (?)
- **leafCount** (*int*) – Unknown.
- **locations** (*list<str>*) – List of locations paths.
- **originallyAvailableAt** (*datetime*) – Datetime show was released.
- **rating** (*float*) – Show rating (7.9; 9.8; 8.1).
- **studio** (*str*) – Studio that created show (Di Bonaventura Pictures; 21 Laps Entertainment).
- **theme** (*str*) – Key to theme resource (/library/metadata/<ratingkey>/theme/<themeid>)
- **viewedLeafCount** (*int*) – Unknown.

- **year** (*int*) – Year the show was released.
- **genres** (List<*Genre*>) – List of genre objects.
- **roles** (List<*Role*>) – List of role objects.
- **similar** (List<*Similar*>) – List of Similar objects.

actors
Alias to self.roles.

isWatched
Returns True if this show is fully watched.

seasons(***kwargs*)
Returns a list of *Season* objects.

season(*title=None*)
Returns the season with the specified title or number.

Parameters **title** (*str or int*) – Title or Number of the season to return.

episodes(***kwargs*)
Returns a list of *Episode* objects.

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

- ***plexapi.exceptions.BadRequest*** – If season and episode is missing.
- ***plexapi.exceptions.NotFound*** – If the episode is missing.

watched()
Returns list of watched *Episode* objects.

unwatched()
Returns list of unwatched *Episode* objects.

get(*title=None*, *season=None*, *episode=None*)
Alias to *episode()*.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)
Download video files to specified directory.

Parameters

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

class plexapi.video.**Season**(*server*, *data*, *initpath=None*)
Bases: *plexapi.video.Video*

Represents a single Show Season (including all episodes).

Variables

- **TAG** (*str*) – 'Directory'
- **TYPE** (*str*) – 'season'
- **leafCount** (*int*) – Number of episodes in season.
- **index** (*int*) – Season number.
- **parentKey** (*str*) – Key to this seasons *Show*.
- **parentRatingKey** (*int*) – Unique key for this seasons *Show*.
- **parentTitle** (*str*) – Title of this seasons *Show*.
- **viewedLeafCount** (*int*) – Number of watched episodes in season.

isWatched
Returns True if this season is fully watched.

seasonNumber
Returns season number.

episodes(***kwargs*)
Returns a list of *Episode* objects.

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).

get(*title=None*, *episode=None*)
Alias to *episode()*.

show()
Return this seasons *Show()*..

watched()
Returns list of watched *Episode* objects.

unwatched()
Returns list of unwatched *Episode* objects.

download(*savepath=None*, *keep_original_name=False*, ***kwargs*)
Download video files to specified directory.

Parameters

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

class plexapi.video.**Episode**(*server*, *data*, *initpath=None*)
Bases: *plexapi.base.Playable*, *plexapi.video.Video*

Represents a single Shows Episode.

Variables

- **TAG** (*str*) – 'Video'

- **TYPE** (*str*) – 'episode'
- **art** (*str*) – Key to episode artwork (/library/metadata/<ratingkey>/art/<artid>)
- **chapterSource** (*str*) – Unknown (media).
- **contentRating** (*str) Content rating (PG-13; NR; TV-G*) –
- **duration** (*int*) – Duration of episode in milliseconds.
- **grandparentArt** (*str*) – Key to this episodes *Show* artwork.
- **grandparentKey** (*str*) – Key to this episodes *Show*.
- **grandparentRatingKey** (*str*) – Unique key for this episodes *Show*.
- **grandparentTheme** (*str*) – Key to this episodes *Show* theme.
- **grandparentThumb** (*str*) – Key to this episodes *Show* thumb.
- **grandparentTitle** (*str*) – Title of this episodes *Show*.
- **guid** (*str*) – Plex GUID (com.plexapp.agents.imdb://tt4302938?lang=en).
- **index** (*int*) – Episode number.
- **originallyAvailableAt** (*datetime*) – Datetime episode was released.
- **parentIndex** (*str*) – Season number of episode.
- **parentKey** (*str*) – Key to this episodes *Season*.
- **parentRatingKey** (*int*) – Unique key for this episodes *Season*.
- **parentThumb** (*str*) – Key to this episodes thumbnail.
- **parentTitle** (*str*) – Name of this episode's season
- **title** (*str*) – Name of this Episode
- **rating** (*float*) – Movie rating (7.9; 9.8; 8.1).
- **viewOffset** (*int*) – View offset in milliseconds.
- **year** (*int*) – Year episode was released.
- **directors** (List<*Director*>) – List of director objects.
- **media** (List<*Media*>) – List of media objects.
- **writers** (List<*Writer*>) – List of writers objects.

locations
This does not exist in plex xml response but is added to have a common interface to get the location of the Movie/Show

seasonNumber
Returns this episodes season number.

seasonEpisode
Returns the s00e00 string containing the season and episode.

season()
" Return this episodes *Season()*..

show()
" Return this episodes *Show()*..

class plexapi.video.**Clip**(*server*, *data*, *initpath=None*)

Bases: *plexapi.base.Playable*, *plexapi.video.Video*

Represents a single Clip.

CHAPTER 21

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

m

p

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W