

---

# **plexdevices Documentation**

***Release 0.4.1***

**Cory Parsons**

May 02, 2016



<b>1</b>	<b>API Documentation</b>	<b>3</b>
1.1	Main Interface . . . . .	3
1.2	Sessions . . . . .	4
1.3	Devices . . . . .	5
1.3.1	Device . . . . .	5
1.3.2	Server . . . . .	6
1.3.3	Player . . . . .	7
1.4	Containers . . . . .	7
1.4.1	Media Container . . . . .	7
1.4.2	Play Queues . . . . .	8
1.4.3	Hubs . . . . .	9
1.5	Items in Containers . . . . .	9
1.5.1	Directories . . . . .	10
1.5.2	Media Directories . . . . .	10
1.5.3	Media Items . . . . .	12
1.6	Remote . . . . .	16
<b>2</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>



A Python module for working with plex devices.

```
>>> import plexdevices
>>> s = plexdevices.create_session(user=username, password=password)
>>> s.refresh_devices()
>>> s.servers
[<Device:Server1 - Plex Media Server>, <Device:Server2 - Plex Media Server>]
>>> on_deck = s.servers[0].media_container('/library/onDeck')
>>> on_deck
<plexdevices.media.MediaContainer object at 0x03BD32B0>
>>> on_deck.children
[<Movie:Movie>, <Episode:Episode>, <Episode:Episode>, <Episode:Episode>]
>>> on_deck.children[0].media[0].parts[0].resolve_url()
http://server/file.mp4?X-Plex-Token=XXXXXXXXXXXX
```



---

API Documentation

---

## 1.1 Main Interface

`plexdevices.create_session` (*user=None, password=None, token=None*)  
Create a *Session*.

### Parameters

- **user** – (optional) Plex.tv username.
- **password** – (optional) Plex.tv password.
- **token** – (optional) an *X-Plex-Token*.

**Returns** *Session* object.

**Return type** *plexdevices.session.Session*

Plex.TV Usage:

```
>>> import plexdevices
>>> s = plexdevices.create_session(username, password)
>>> s.refresh_devices()
>>> on_deck = s.servers[0].media_container('/library/onDeck')
>>> on_deck.children[0].resolve_url()
http://server/file.mp4?X-Plex-Token=XXXXXXXXXXXX
```

Manual Usage:

```
>>> import plexdevices
>>> s = plexdevices.create_session()
>>> s.manual_add_server('192.168.1.1', 32400)
>>> s.manual_add_server('192.168.1.2', 32400, token='w1zWSdJXzdeQEnpgdmLxB')
>>> s.servers
[<Device:A - Plex Media Server>, <Device:B - Plex Media Server>]
```

**note:** manually connecting to a server requires either an *X-Plex-Token* (see [Plex support article](#)) or an ip address added to the List of networks that are allowed without auth PMS Network setting.

`plexdevices.create_remote` (*player, name, port=8000, callback=None*)  
Create a *Remote*.

### Parameters

- **player** – the *Player* to create the remote for.
- **name** – the name of the remote.

- **port** – (optional) HTTP port to allow the Player to communicate to the remote.
- **callback** – (optional) a function that will be called with a single argument when the player *POSTs* data. (when using the subscribe method).

**Returns** *Remote* object.

**Return type** *plexdevices.remote.Remote*

Setup:

```
>>> import plexdevices
>>> s = plexdevices.create_session(username, password)
>>> s.refresh_devices()
>>> s.players
[<Device:My-PC - Plex Media Player>]
>>> player = s.players[0]
```

Usage (subscribe method):

```
>>> r = plexdevices.create_remote(player, 'myremote')
>>> r.timeline_subscribe()
>>> r.down()
>>> r.select()
>>> r.timeline_unsubscribe()
```

Usage (poll method):

```
>>> r = plexdevices.create_remote(player, 'myremote')
>>> timeline_data = r.timeline_poll()
>>> timeline_data
{
  'video': {'type': 'video', 'time': '0', 'seekRange': '0-0'},
  'photo': {'type': 'photo', 'time': '0', 'seekRange': '0-0',
            'controllable': 'playPause,stop,skipPrevious,skipNext'}
  'music': {'type': 'music', 'time': '0', 'seekRange': '0-0',
            'controllable': 'playPause,stop,skipPrevious,skipNext,seekTo'},
  'location': 'navigation',
  'commandID': '0'
}
>>> r.down()
>>> r.select()
```

---

## 1.2 Sessions

**class** `plexdevices.session.Session`

A Plex session. You can use pickle to save and load existing session objects.

**login** (*password*)

Retrieve the token for the session user from `https://plex.tv/users/sign_in.json`.

**manual\_add\_server** (*address, port=32400, protocol='http', token=''*)

Add a *Server* to the session.

**Parameters**

- **address** – address to the server. e.g. 127.0.0.1.
- **token** – (optional) the X-Plex-Token to use when accessing this server.



**players = None**

List of *Player*'s accessible by the current user.

**refresh\_devices()**

Retrieve the devices for the current user from `https://plex.tv/api/resources`

**refresh\_users()**

Retrieve the Plex Home users from `https://plex.tv/api/home/users`.

**servers = None**

List of *Server*'s accessible by the current user.

**switch\_user(*user\_id*, *pin*=None)**

Switch the current user to the given user id, and refresh the available devices.

**Parameters**

- **user\_id** – the *id* of the user. As given from `https://plex.tv/api/home/users`.
- **pin** – (optional) the 4-digit PIN code of the user.

**users = None**

List of Plex Home users.

---

## 1.3 Devices

### 1.3.1 Device

`class plexdevices.device.Device`

**access\_token**

**client\_identifier**

Unique identifier string.

**created\_at**

**device**

**headers**

**https\_required**

Type bool

**last\_seen\_at**

**name**

**owned**

Type bool

**platform**

Operating system of the device.

**platform\_version**

Operating system version.

**presence**

Type bool

**product**

Plex product name. e.g. Plex Media Server

**product\_version**

Version of the Plex product. e.g. 0.9.16.4.1911-ee6e505

**provides**

Type list

**public\_address\_matches**

Type bool

**request** (*endpoint*, *method*='GET', *data*=None, *params*=None, *headers*={}, *raw*=False, *allow\_redirects*=True)

Make an HTTP request to the device.

Parameters

- **endpoint** – location on server. e.g. /library/onDeck.
- **method** – (optional) request function. Defaults to GET.
- **data** – (optional) data to send with the request. Defaults to None.
- **params** – (optional) params to include in the URL. Defaults to None.
- **headers** – (optional) additional headers. Defaults to {}.
- **raw** – (optional) return raw data. Defaults to False.
- **allow\_redirects** – (optional) follow 302 redirects. Defaults to True.

Returns (HTTP status code, data)

Return type Tuple (int, str)

**synced**

Type bool

### 1.3.2 Server

class `plexdevices.device.Server`

Bases: `plexdevices.device.Device`

A `Device` which provides a server.

**container** (*endpoint*, *size*=None, *page*=None, *params*=None, *usejson*=True)

Parameters

- **endpoint** – destination on the server. e.g. /library/onDeck.
- **size** – (optional) the max number of items to retrieve.
- **page** – (optional) the page number for paging large containers.
- **params** – (optional) Dictionary of parameters to be added to the url in the request.

Returns a Dictionary representing a Plex Media Container.

Return type Dictionary

**hub** (*endpoint, size=None, page=None, params=None*)  
added in 0.4.0

**Parameters**

- **endpoint** – destination on the hubs api. e.g. /hubs/onDeck.
- **params** – (optional) Dictionary of parameters to be added to the url in the request.

**Return type** *HubsContainer*

**image** (*endpoint, w=None, h=None*)

If w and h are set, the server will transcode the image to the given size.

**Parameters**

- **endpoint** – location of the image. This can also be a full URL of an image not on the server (for easy channel support).
- **w** – (optional) width to transcode.
- **h** – (optional) height to transcode.

**Returns** Raw data of an image.

**media\_container** (*endpoint, size=None, page=None, params=None, usejson=True*)

**Parameters**

- **endpoint** – destination on the server. e.g. /library/onDeck.
- **size** – (optional) the max number of items to retrieve.
- **page** – (optional) the page number for paging large containers.
- **params** – (optional) Dictionary of parameters to be added to the url in the request.

**Returns** a *MediaContainer* representing a Plex Media Container.

**Return type** *MediaContainer*

### 1.3.3 Player

**class** plexdevices.device.**Player**

Bases: *plexdevices.device.Device*

A *Device* which provides a player.

---

## 1.4 Containers

### 1.4.1 Media Container

**class** plexdevices.media.**MediaContainer**

An object representing a Plex MediaContainer.

**children** = None

List of items in the container.

**data** = None

Dictionary of the MediaContainer's values.

**server = None**

The *Server* which this container was retrieved from.

## 1.4.2 Play Queues

**class** plexdevices.media.**PlayQueue**

Bases: *plexdevices.media.MediaContainer*

An object representing a Plex PlayQueue. A PlayQueue is a playlist that is maintained by the server.

When you want to play a Media item, use the `create()` method. Then during playback, use `timeline_update()` to let the server know about the players state.

**add\_item** (*item*, *player\_headers*)

Add a *MediaDirectory* or *MediaItems* to the PlayQueue.

**static create** (*item*, *player\_headers*)

Create a PlayQueue on a server and return a PlayQueue object.

### Parameters

- **item** – the *Media* to be the initial item added to the PlayQueue.
- **player\_headers** – Dictionary of headers identifying the player using the PlayQueue. Must include X-Plex-Client-Identifier and X-Plex-Device-Name.

**Returns** *PlayQueue* object

**Return type** plexdevices.PlayQueue

**get\_next** ()

Select and return the next *MediaItem* in the PlayQueue, or *None*.

**get\_prev** ()

Select and return the previous *MediaItem* in the PlayQueue, or *None*.

**remove\_item** (*item*)

Remove a *MediaItem* from the PlayQueue.

**selected\_item**

The selected *MediaItem*

**timeline\_update** (*item*, *time*, *headers*, *state*='playing')

Update the timeline. This should be done frequently during playback. The server will update the view offset of the item and handle the watched/unwatched state.

### Parameters

- **item** – the play queue item.
- **time** – the current playback time in ms.
- **headers** – the players headers. X-Plex-Client-Identifier, X-Plex-Device-Name.
- **state** – playing, stopped, paused

**update** ()

Update the *PlayQueue*'s data from its server.

### 1.4.3 Hubs

*added in 0.4.0*

These are for the Plex Media Server api at /hubs.

```
class plexdevices.hubs.HubsContainer
    Bases: plexdevices.media.MediaContainer

    allow_sync

    children = None
        List of items in the container. BaseObject, Hub

    data = None
        Dictionary of the HubsContainer's values.

    is_library

    server = None
        The Server which this container was retrieved from.

    size
```

```
class plexdevices.hubs.Hub
    A Hub is a container inside a HubsContainer which groups together multiple items.

    children = None
        List of BaseObject's in the Hub.

    container = None
        The HubsContainer which holds this item.

    hub_identifier

    hub_key

    key

    more

    size

    title

    type
```

---

## 1.5 Items in Containers

Deciding what to do with the items in a container should be done by checking the type with `isinstance()`.

```
def item_clicked(item):
    if isinstance(item, (plexdevices.media.Movie, plexdevices.media.Episode)):
        video_player.play(item.resolve_url())
    elif isinstance(item, plexdevices.media.Directory):
        next_container = server.media_container(item.key)
```

## 1.5.1 Directories

**class** `plexdevices.media.Directory`  
Bases: `plexdevices.media.BaseObject`

a directory that is used for navigation.

**key**

The endpoint on the server which this object points to. This can be relative or absolute.

**title**

**class** `plexdevices.media.InputDirectory`  
Bases: `plexdevices.media.Directory`

A special *Directory* where you should get a string from the user and send it to the key as a *query* parameter.

```
if isinstance(item, InputDirectory):
    user_input = get_string()
    next_container = server.media_container(item.key, params={'query': user_input})
```

**class** `plexdevices.media.PreferencesDirectory`  
Bases: `plexdevices.media.Directory`

A special *Directory* used in channels for channel preferences.

## 1.5.2 Media Directories

### MediaDirectory

**class** `plexdevices.media.MediaDirectory`  
Bases: `plexdevices.media.Directory`, `plexdevices.media.Metadata`

A directory that holds *MediaItems*. These directories have metadata, can be added to a *PlayQueue*, and can be marked watched/unwatched. A *MediaContainer* with its key will contain all the *MediaItems*.

**added\_at**

Unix timestamp.

**allow\_sync**

Type bool

**art**

Art key.

**key**

The endpoint on the server which this object points to. This can be relative or absolute.

**mark\_unwatched()**

Mark this item as unwatched on its server.

**mark\_watched()**

Mark this item as watched on its server.

**summary**

Description of the item.

**thumb**

Either the key to the thumb, or None. Use `image()` to get the image data.

**title**

**updated\_at**  
Unix timestamp.

## PhotoAlbum

**class** `plexdevices.media.PhotoAlbum`  
Bases: `plexdevices.media.MediaDirectory`  
`MediaDirectory` with extra metadata for a Photo Album.

## Album

**class** `plexdevices.media.Album`  
Bases: `plexdevices.media.MediaDirectory`  
`MediaDirectory` with extra metadata for a music Album.

**genres**  
Type list

**originally\_available\_at**  
Release date.

**parent\_key**  
Artist key.

**parent\_title**  
Artist name.

**year**

## Artist

**class** `plexdevices.media.Artist`  
Bases: `plexdevices.media.MediaDirectory`  
`MediaDirectory` with extra metadata for a music Artist.

**country**  
Type list

**genres**  
Type list

## Season

**class** `plexdevices.media.Season`  
Bases: `plexdevices.media.MediaDirectory`  
`MediaDirectory` with extra metadata for a TV Season.

**count**  
Number of episodes.

**parent\_key**  
Show key.

**parent\_summary**  
Show summary.

**parent\_thumb**  
Show thumb.

**parent\_title**  
Show name.

**unwatched\_count**

**watched\_count**

## Show

**class** `plexdevices.media.Show`  
Bases: `plexdevices.media.MediaDirectory`  
`MediaDirectory` with extra metadata for a TV Show.

**banner**

**child\_count**  
Number of seasons.

**count**  
Number of episodes.

**duration**  
Length of one episode in ms.

**originally\_available\_at**  
Premiere date.

**studio**

**unwatched\_count**

**watched\_count**

## 1.5.3 Media Items

### MediaItem

**class** `plexdevices.media.MediaItem`  
Bases: `plexdevices.media.BaseObject`, `plexdevices.media.Metadata`  
An object representing a piece of Media.

**added\_at**  
Unix timestamp.

**allow\_sync**  
Type bool

**art**  
Art key.

**in\_progress**  
True if the item is in progress.



**key**

The endpoint on the server which this object points to. This can be relative or absolute.

**last\_viewed\_at**

not always available.

**mark\_unwatched()**

Mark this item as unwatched on its server.

**mark\_watched()**

Mark this item as watched on its server.

**media = None**

List of *Media* objects that hold information about the file.

**resolve\_url()**

Return the url of the first part regardless of how many there are.

**summary**

Description of the item.

**thumb**

Either the key to the thumb, or None. Use `image()` to get the image data.

**title****updated\_at**

Unix timestamp.

**view\_offset**

The resume position in ms.

**watched**

True if the item is watched.

**year**

## Media

**class plexdevices.media.Media**

A Media object represents a single copy of a *MediaItem*. In most cases, a *MediaItem* will have a single Media object. If the server has a 480p and a 1080p copy of a movie, there will be two Media objects.

**aspect\_ratio**

Aspect ratio as floating point number. e.g. 1.78 for 16:9.

**audio\_channels**

Number of audio channels.

**audio\_codec**

Audio codec as a string.

**bitrate****container**

Container file format. e.g. mkv.

**duration**

Duration in ms.

**height**

Vertical resolution.

**id**

**parts = None**

List of *Part* objects which references the actual files. Typically there is only one part.

**video\_codec**

Video codec as a string.

**video\_frame\_rate**

Framerate as a string. e.g. 24p, NTSC.

**video\_profile**

main, high, ...

**video\_resolution**

Vertical resolution as an integer. e.g. 1080

**width**

Horizontal resolution.

## Parts

**class** `plexdevices.media.Part`

A part represents an actual file.

**container**

Container file format. e.g. mkv.

**duration**

Duration in ms.

**file**

The path to the file on the server. use the key to actually access it.

**id**

**key**

The key to playing this item. It either points to the file on the server, or to a function on the server that will resolve it to a playable url.

**resolve\_key()**

Resolve the key into a url which can be given to a media player.

**size**

Size of the file in bytes.

**video\_profile**

main, high, ...

## Episode

**class** `plexdevices.media.Episode`

Bases: `plexdevices.media.MediaItem`

*MediaItem* with extra metadata for a TV Show Episode.

**duration**

Duration in ms.

**grandparent\_key**

Key of the show.

**grandparent\_title**

Title of the show.

**index**

Episode number.

**originally\_available\_at**

Air date.

**parent\_index**

Season number.

**parent\_key**

Key of the season.

## Movie

```
class plexdevices.media.Movie
```

Bases: *plexdevices.media.MediaItem*

*MediaItem* with extra metadata for a Movie.

**duration**

Duration in ms.

**originally\_available\_at**

**Type** datetime.date

**rating**

Movie rating as a floating point number.

**studio****tagline**

## Track

```
class plexdevices.media.Track
```

Bases: *plexdevices.media.MediaItem*

*MediaItem* with extra metadata for a Music Track.

**duration**

Duration in ms.

**grandparent\_key**

Artist key.

**grandparent\_thumb****grandparent\_title**

Artist name.

**index**

Track number.

**parent\_key**

Album key.

**parent\_thumb**

**parent\_title**  
Album name.

## Photo

**class** plexdevices.media.**Photo**  
Bases: *plexdevices.media.MediaItem*  
*MediaItem* with extra metadata for a Photo.

**grandparent\_key**  
Key to the folder which holds the photoalbum.

**originally\_available\_at**

**parent\_key**  
Key to the album which holds this photo.

---

## 1.6 Remote

**class** plexdevices.remote.**Remote**  
A remote control for a Plex device.

**identifier = None**  
The unique identifier string for this device.

**name = None**

**player = None**

**port = None**

**headers**  
Dictionary of the remote's X-Plex-Client-Identifier and X-Plex-Device-Name.

**command** (*command*, *params=None*)  
Send a command to the player with optional parameters.

**timeline()**  
Returns the latest timeline that was POSTed from the player.

**timeline\_subscribe()**  
Subscribe to the timeline.

**timeline\_unsubscribe()**  
Unsubscribe from the timeline.

**timeline\_poll()**  
This is an alternative to the subscribe command for controllers that cannot use persistent connections to receive updates from the player.

**mirror** (*plex\_object*, *\*\*kwargs*)  
Send the player to the preplay screen of the given BaseObject.

**play\_media** (*media\_object*)  
Make the player play the given *MediaItem*.

**up()**  
Navigation: up

**down()**  
Navigation: down

**left()**  
Navigation: left

**right()**  
Navigation: right

**select()**  
Navigation: select

**back()**  
Navigation: back

**home()**  
Navigation: home

**music()**  
Navigation: music  
  
Navigate to the player's music playback view, if music is currently playing.

**pause** (*media\_type=None*)  
Playback control: pause

**Parameters** **media\_type** – (optional) *music*, *photo*, *video* in case there are multiple things happening.

**play** (*media\_type=None*)  
Playback control: play

**skip\_next** (*media\_type=None*)  
Playback control: skip next

**skip\_previous** (*media\_type=None*)  
Playback control: skip previous

**stop** (*media\_type=None*)  
Playback control: stop

**seek** (*media\_type, offset*)  
Playback control: seek to

**Parameters** **offset** – absolute position in milliseconds.

**skip** (*media\_type, key*)  
Playback control: skip to item with matching key

**step\_back** (*media\_type*)  
Playback control: step back

**step\_forward** (*media\_type*)  
Playback control: step forward

**volume** (*media\_type, level*)  
Playback control: set volume.

**Parameters**

- **media\_type** – mandatory. *music*, *photo*, *video*.
- **level** – volume level [0-100].



---

## Indices and tables

---

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





**p**

plexdevices, 3



## A

access\_token (plexdevices.device.Device attribute), 5  
add\_item() (plexdevices.media.PlayQueue method), 8  
added\_at (plexdevices.media.MediaDirectory attribute), 10  
added\_at (plexdevices.media.MediaItem attribute), 12  
Album (class in plexdevices.media), 11  
allow\_sync (plexdevices.hubs.HubsContainer attribute), 9  
allow\_sync (plexdevices.media.MediaDirectory attribute), 10  
allow\_sync (plexdevices.media.MediaItem attribute), 12  
art (plexdevices.media.MediaDirectory attribute), 10  
art (plexdevices.media.MediaItem attribute), 12  
Artist (class in plexdevices.media), 11  
aspect\_ratio (plexdevices.media.Media attribute), 13  
audio\_channels (plexdevices.media.Media attribute), 13  
audio\_codec (plexdevices.media.Media attribute), 13

## B

back() (plexdevices.remote.Remote method), 17  
banner (plexdevices.media.Show attribute), 12  
bitrate (plexdevices.media.Media attribute), 13

## C

child\_count (plexdevices.media.Show attribute), 12  
children (plexdevices.hubs.Hub attribute), 9  
children (plexdevices.hubs.HubsContainer attribute), 9  
children (plexdevices.media.MediaContainer attribute), 7  
client\_identifier (plexdevices.device.Device attribute), 5  
command() (plexdevices.remote.Remote method), 16  
container (plexdevices.hubs.Hub attribute), 9  
container (plexdevices.media.Media attribute), 13  
container (plexdevices.media.Part attribute), 14  
container() (plexdevices.device.Server method), 6  
count (plexdevices.media.Season attribute), 11  
count (plexdevices.media.Show attribute), 12  
country (plexdevices.media.Artist attribute), 11  
create() (plexdevices.media.PlayQueue static method), 8  
create\_remote() (in module plexdevices), 3  
create\_session() (in module plexdevices), 3

created\_at (plexdevices.device.Device attribute), 5

## D

data (plexdevices.hubs.HubsContainer attribute), 9  
data (plexdevices.media.MediaContainer attribute), 7  
Device (class in plexdevices.device), 5  
device (plexdevices.device.Device attribute), 5  
Directory (class in plexdevices.media), 10  
down() (plexdevices.remote.Remote method), 17  
duration (plexdevices.media.Episode attribute), 14  
duration (plexdevices.media.Media attribute), 13  
duration (plexdevices.media.Movie attribute), 15  
duration (plexdevices.media.Part attribute), 14  
duration (plexdevices.media.Show attribute), 12  
duration (plexdevices.media.Track attribute), 15

## E

Episode (class in plexdevices.media), 14

## F

file (plexdevices.media.Part attribute), 14

## G

genres (plexdevices.media.Album attribute), 11  
genres (plexdevices.media.Artist attribute), 11  
get\_next() (plexdevices.media.PlayQueue method), 8  
get\_prev() (plexdevices.media.PlayQueue method), 8  
grandparent\_key (plexdevices.media.Episode attribute), 14  
grandparent\_key (plexdevices.media.Photo attribute), 16  
grandparent\_key (plexdevices.media.Track attribute), 15  
grandparent\_thumb (plexdevices.media.Track attribute), 15  
grandparent\_title (plexdevices.media.Episode attribute), 14  
grandparent\_title (plexdevices.media.Track attribute), 15

## H

headers (plexdevices.device.Device attribute), 5  
headers (plexdevices.remote.Remote attribute), 16

height (plexdevices.media.Media attribute), 13  
home() (plexdevices.remote.Remote method), 17  
https\_required (plexdevices.device.Device attribute), 5  
Hub (class in plexdevices.hubs), 9  
hub() (plexdevices.device.Server method), 6  
hub\_identifier (plexdevices.hubs.Hub attribute), 9  
hub\_key (plexdevices.hubs.Hub attribute), 9  
HubsContainer (class in plexdevices.hubs), 9

## I

id (plexdevices.media.Media attribute), 13  
id (plexdevices.media.Part attribute), 14  
identifier (plexdevices.remote.Remote attribute), 16  
image() (plexdevices.device.Server method), 7  
in\_progress (plexdevices.media.MediaItem attribute), 12  
index (plexdevices.media.Episode attribute), 15  
index (plexdevices.media.Track attribute), 15  
InputDirectory (class in plexdevices.media), 10  
is\_library (plexdevices.hubs.HubsContainer attribute), 9

## K

key (plexdevices.hubs.Hub attribute), 9  
key (plexdevices.media.Directory attribute), 10  
key (plexdevices.media.MediaDirectory attribute), 10  
key (plexdevices.media.MediaItem attribute), 12  
key (plexdevices.media.Part attribute), 14

## L

last\_seen\_at (plexdevices.device.Device attribute), 5  
last\_viewed\_at (plexdevices.media.MediaItem attribute), 13  
left() (plexdevices.remote.Remote method), 17  
login() (plexdevices.session.Session method), 4

## M

manual\_add\_server() (plexdevices.session.Session method), 4  
mark\_unwatched() (plexdevices.media.MediaDirectory method), 10  
mark\_unwatched() (plexdevices.media.MediaItem method), 13  
mark\_watched() (plexdevices.media.MediaDirectory method), 10  
mark\_watched() (plexdevices.media.MediaItem method), 13  
Media (class in plexdevices.media), 13  
media (plexdevices.media.MediaItem attribute), 13  
media\_container() (plexdevices.device.Server method), 7  
MediaContainer (class in plexdevices.media), 7  
MediaDirectory (class in plexdevices.media), 10  
MediaItem (class in plexdevices.media), 12  
mirror() (plexdevices.remote.Remote method), 16  
more (plexdevices.hubs.Hub attribute), 9

Movie (class in plexdevices.media), 15  
music() (plexdevices.remote.Remote method), 17

## N

name (plexdevices.device.Device attribute), 5  
name (plexdevices.remote.Remote attribute), 16

## O

originally\_available\_at (plexdevices.media.Album attribute), 11  
originally\_available\_at (plexdevices.media.Episode attribute), 15  
originally\_available\_at (plexdevices.media.Movie attribute), 15  
originally\_available\_at (plexdevices.media.Photo attribute), 16  
originally\_available\_at (plexdevices.media.Show attribute), 12  
owned (plexdevices.device.Device attribute), 5

## P

parent\_index (plexdevices.media.Episode attribute), 15  
parent\_key (plexdevices.media.Album attribute), 11  
parent\_key (plexdevices.media.Episode attribute), 15  
parent\_key (plexdevices.media.Photo attribute), 16  
parent\_key (plexdevices.media.Season attribute), 11  
parent\_key (plexdevices.media.Track attribute), 15  
parent\_summary (plexdevices.media.Season attribute), 11  
parent\_thumb (plexdevices.media.Season attribute), 12  
parent\_thumb (plexdevices.media.Track attribute), 15  
parent\_title (plexdevices.media.Album attribute), 11  
parent\_title (plexdevices.media.Season attribute), 12  
parent\_title (plexdevices.media.Track attribute), 15  
Part (class in plexdevices.media), 14  
parts (plexdevices.media.Media attribute), 14  
pause() (plexdevices.remote.Remote method), 17  
Photo (class in plexdevices.media), 16  
PhotoAlbum (class in plexdevices.media), 11  
platform (plexdevices.device.Device attribute), 5  
platform\_version (plexdevices.device.Device attribute), 5  
play() (plexdevices.remote.Remote method), 17  
play\_media() (plexdevices.remote.Remote method), 16  
Player (class in plexdevices.device), 7  
player (plexdevices.remote.Remote attribute), 16  
players (plexdevices.session.Session attribute), 5  
PlayQueue (class in plexdevices.media), 8  
plexdevices (module), 3  
port (plexdevices.remote.Remote attribute), 16  
PreferencesDirectory (class in plexdevices.media), 10  
presence (plexdevices.device.Device attribute), 5  
product (plexdevices.device.Device attribute), 6  
product\_version (plexdevices.device.Device attribute), 6  
provides (plexdevices.device.Device attribute), 6

public\_address\_matches (plexdevices.device.Device attribute), 6

## R

rating (plexdevices.media.Movie attribute), 15  
 refresh\_devices() (plexdevices.session.Session method), 5  
 refresh\_users() (plexdevices.session.Session method), 5  
 Remote (class in plexdevices.remote), 16  
 remove\_item() (plexdevices.media.PlayQueue method), 8  
 request() (plexdevices.device.Device method), 6  
 resolve\_key() (plexdevices.media.Part method), 14  
 resolve\_url() (plexdevices.media.MediaItem method), 13  
 right() (plexdevices.remote.Remote method), 17

## S

Season (class in plexdevices.media), 11  
 seek() (plexdevices.remote.Remote method), 17  
 select() (plexdevices.remote.Remote method), 17  
 selected\_item (plexdevices.media.PlayQueue attribute), 8  
 Server (class in plexdevices.device), 6  
 server (plexdevices.hubs.HubsContainer attribute), 9  
 server (plexdevices.media.MediaContainer attribute), 7  
 servers (plexdevices.session.Session attribute), 5  
 Session (class in plexdevices.session), 4  
 Show (class in plexdevices.media), 12  
 size (plexdevices.hubs.Hub attribute), 9  
 size (plexdevices.hubs.HubsContainer attribute), 9  
 size (plexdevices.media.Part attribute), 14  
 skip() (plexdevices.remote.Remote method), 17  
 skip\_next() (plexdevices.remote.Remote method), 17  
 skip\_previous() (plexdevices.remote.Remote method), 17  
 step\_back() (plexdevices.remote.Remote method), 17  
 step\_forward() (plexdevices.remote.Remote method), 17  
 stop() (plexdevices.remote.Remote method), 17  
 studio (plexdevices.media.Movie attribute), 15  
 studio (plexdevices.media.Show attribute), 12  
 summary (plexdevices.media.MediaDirectory attribute), 10  
 summary (plexdevices.media.MediaItem attribute), 13  
 switch\_user() (plexdevices.session.Session method), 5  
 synced (plexdevices.device.Device attribute), 6

## T

tagline (plexdevices.media.Movie attribute), 15  
 thumb (plexdevices.media.MediaDirectory attribute), 10  
 thumb (plexdevices.media.MediaItem attribute), 13  
 timeline() (plexdevices.remote.Remote method), 16  
 timeline\_poll() (plexdevices.remote.Remote method), 16  
 timeline\_subscribe() (plexdevices.remote.Remote method), 16  
 timeline\_unsubscribe() (plexdevices.remote.Remote method), 16

timeline\_update() (plexdevices.media.PlayQueue method), 8  
 title (plexdevices.hubs.Hub attribute), 9  
 title (plexdevices.media.Directory attribute), 10  
 title (plexdevices.media.MediaDirectory attribute), 10  
 title (plexdevices.media.MediaItem attribute), 13  
 Track (class in plexdevices.media), 15  
 type (plexdevices.hubs.Hub attribute), 9

## U

unwatched\_count (plexdevices.media.Season attribute), 12  
 unwatched\_count (plexdevices.media.Show attribute), 12  
 up() (plexdevices.remote.Remote method), 16  
 update() (plexdevices.media.PlayQueue method), 8  
 updated\_at (plexdevices.media.MediaDirectory attribute), 10  
 updated\_at (plexdevices.media.MediaItem attribute), 13  
 users (plexdevices.session.Session attribute), 5

## V

video\_codec (plexdevices.media.Media attribute), 14  
 video\_frame\_rate (plexdevices.media.Media attribute), 14  
 video\_profile (plexdevices.media.Media attribute), 14  
 video\_profile (plexdevices.media.Part attribute), 14  
 video\_resolution (plexdevices.media.Media attribute), 14  
 view\_offset (plexdevices.media.MediaItem attribute), 13  
 volume() (plexdevices.remote.Remote method), 17

## W

watched (plexdevices.media.MediaItem attribute), 13  
 watched\_count (plexdevices.media.Season attribute), 12  
 watched\_count (plexdevices.media.Show attribute), 12  
 width (plexdevices.media.Media attribute), 14

## Y

year (plexdevices.media.Album attribute), 11  
 year (plexdevices.media.MediaItem attribute), 13