dizqueTV

Release VERSIONADDEDBYGITHUB

Nate Harris

Mar 06, 2023

CONTENTS:

CHAPTER

ONE

INSTALLATION

1.1 From GitHub

- Clone the repository with `git clone https://github.com/nwithan8/dizqueTV-python.git`
- Enter project folder with `cd dizqueTV-python`
- Install requirements with `pip install -r requirements.txt`

1.2 From PyPi

- Run `pip install dizqueTV`

CHAPTER

TWO

SETUP

Import the `API` class from the dizqueTV module

```
from dizqueTV import API
dtv = API(url="http://localhost:8000")
```

Enable verbose logging by passing `verbose=True` into the API object declaration

CHAPTER

THREE

DOCUMENTATION

3.1 DizqueTV

class dizqueTV.dizquetv.**API**(*url: str*, *verbose: bool = False*, *allow_analytics: bool = True*, *anonymous_analytics: bool = True*)

Bases: object

add_channel(*programs: Optional[List[Union[Program, Redirect, Video, Movie, Episode, Track]]] = None*, *plex_server: Optional[PlexServer] = None*, *handle_errors: bool = True*, ***kwargs*) → Optional[*Channel*]

Add a channel to dizqueTV

Parameters

- **programs** (*List[Union[Program, Redirect, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]], optional*) – Program, Redirect or PlexAPI Video, Movie, Episode or Track objects to add to the new channel
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer (optional, required if adding PlexAPI Video, Movie, Episode or Track)
- **kwargs** – keyword arguments of setting names and values
- **handle_errors** (*bool, optional*) – Suppress error if they arise (ex. alter invalid channel number, add Flex Time if no program is included)

Returns
new Channel object or None

Return type
Channel

add_custom_show(*name: str*, *content: List[Union[Program, Video, Movie, Episode, Track]]*, *plex_server: Optional[PlexServer] = None*) → Optional[*CustomShow*]

Add a dizqueTV custom show

Parameters

- **name** (*str*) – Name of custom show to add
- **content** (*List[Union[Program, Video, Movie, Episode, Track]]*) – List of content to add to custom show

- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (required if adding PlexAPI Video, Movie, Episode or Track objects)

Returns
New CustomShow object if successful, None if unsuccessful

Return type
CustomShow or None

add_filler_list(*content: List[Union[*Program*, Video, Movie, Episode, Track]]*, *plex_server: Optional[PlexServer] = None*, *handle_errors: bool = False*, ***kwargs*) → Optional[*FillerList*]

Add a filler list to dizqueTV Must include at least one program to create

Parameters

- **content** (*List[Union[*Program*, Video, Movie, Episode, Track]]*) – At least one Program or PlexAPI Video, Movie, Episode or Track to add to the new filler list
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer (optional, required if adding PlexAPI Video, Movie, Episode or Track)
- **kwargs** – keyword arguments of setting names and values
- **handle_errors** (*bool, optional*) – Suppress error if they arise (ex. add redirect if no program is included)

Returns
new FillerList object or None

Return type
FillerList

add_filler_lists_to_channels(*filler_lists: List[*FillerList*]*, *channels: Optional[List[*Channel*]] = None*, *channel_numbers: Optional[List[int]] = None*) → bool

Add multiple filler lists to multiple channels

Parameters

- **filler_lists** (*List[*FillerList*]*) – List of FillerList objects
- **channels** (*List[*Channel*], optional*) – List of Channel objects (optional)
- **channel_numbers** (*List[int], optional*) – List of channel numbers

Returns
True if successful, False if unsuccessful (Channel objects reload in place)

Return type
bool

add_plex_server(***kwargs*) → Optional[*PlexServer*]

Add a Plex Media Server to dizqueTV

Parameters
kwargs – keyword arguments of setting names and values

Returns
PlexServer object or None

Return type
PlexServer

add_plex_server_from_plexapi(*plex_server: PlexServer*) → Optional[*PlexServer*]

Convert and add a plexapi.PlexServer as a Plex Media Server to dizqueTV

Parameters
plex_server (*plexapi.server.PlexServer*) – plexapi.PlexServer object to add to dizqueTV

Returns
PlexServer object or None

Return type
PlexServer

add_programs_to_channels(*programs: List[Union[*Program*,* CustomShow*, Video, Movie, Episode, Track]]*, *channels: Optional[List[*Channel*]] = None*, *channel_numbers: Optional[List[int]] = None*, *plex_server: Optional[PlexServer] = None*) → bool

Add multiple programs to multiple channels

Parameters

- **programs** (*List[Union[*Program*,* CustomShow*, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]]*) – List of Program, CustomShow plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track objects
- **channels** (*List[*Channel*], optional*) – List of Channel objects (optional)
- **channel_numbers** (*List[int], optional*) – List of channel numbers
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (required if adding PlexAPI Video, Movie, Episode or Track objects)

Returns
True if successful, False if unsuccessful (Channel objects reload in place)

Return type
bool

convert_plex_item_to_filler_item(*plex_item: Union[Video, Movie, Episode, Track]*, *plex_server: PlexServer*) → *FillerItem*

Convert a PlexAPI Video, Movie, Episode or Track object into a FillerItem

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]*) – plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track object
- **plex_server** (*plexapi.server.PlexServer*) – plexapi.server.PlexServer object

Returns
Program object

Return type
Program

convert_plex_item_to_program(*plex_item: Union[Video, Movie, Episode, Track]*, *plex_server: PlexServer*) → *Program*

Convert a PlexAPI Video, Movie, Episode or Track object into a Program

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]*) – plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track object
- **plex_server** (*plexapi.server.PlexServer*) – plexapi.server.PlexServer object

Returns
Program object

Return type
Program

convert_program_to_custom_show_item(*program:* Program) → *CustomShowItem*

Convert a dizqueTV Program to a dizqueTV CustomShowItem (add durationStr and commercials)

Parameters
program (Program) – Program to convert

Returns
CustomShowItem

Return type
CustomShowItem

create_custom_show_with_programs(*custom_show_programs: list*) → *CustomShow*

delete_channel(*channel_number: int*) → bool

Delete a dizqueTV channel

Parameters
channel_number (*int*) – Number of channel to delete

Returns
True if successful, False if unsuccessful

Return type
bool

delete_custom_show(*custom_show_id: str*) → bool

Delete a dizqueTV custom show

Parameters
custom_show_id (*str*) – ID of CustomShow to delete

Returns
True if successful, False if unsuccessful

Return type
bool

delete_filler_list(*filler_list_id: str*) → bool

Delete a dizqueTV filler list

Parameters
filler_list_id (*str*) – ID of FillerList to delete

Returns
True if successful, False if unsuccessful

Return type
bool

delete_plex_server(*server_name: str*) → bool

Remove a Plex Media Server from dizqueTV

Parameters
server_name (*str*) – Name of Plex Server

Returns
True if successful, False if unsuccessful

Return type
bool

expand_custom_show_items(*programs: List[Union[*Program*,* FillerItem*,* CustomShow*, Video, Movie, Episode, Track]]*) → List[Union[*Program*, *Redirect*, *FillerItem*, Video, Movie, Episode, Track]]

Expand all custom shows in a list out to their individual programs

Parameters
programs (*list*) – List of programs (i.e. Program, Movie, Video, Track, CustomShow)

Returns
list of all programs (including custom show programs)

Return type
list

get_channel(*channel_number: Optional[int] = None*, *channel_name: Optional[str] = None*) → Optional[*Channel*]

Get a specific dizqueTV channel by number or name

Parameters

- **channel_number** (*int, optional*) – Number of channel
- **channel_name** (*str, optional*) – Name of channel

Returns
Channel object or None

Return type
Channel

get_channel_info(*channel_number: int*) → <module 'json' from '/home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/__init__.py'>

Get the name, number and icon for a dizqueTV channel

Parameters
channel_number (*int*) – Number of channel

Returns
JSON data with channel name, number and icon path

Return type
dict

get_channel_m3u(*channel_number: int*) → M3U8

Get a channel-specific m3u playlist

Parameters
channel_number (*int*) – Number of channel to get M3U playlist

Returns
m3u8 object

Return type
m3u8.model.M3U8

`get_channel_programs`(*channel_number: int*) → List[Union[*Program*, *CustomShow*]]

`get_channel_without_programs`(*channel_number: int*) → Optional[*Channel*]

`get_custom_show`(*custom_show_id: str*) → Optional[*CustomShow*]

Get a CustomShow object by its ID

Parameters
custom_show_id (*str*) – ID of custom show

Returns
CustomShow object or None

Return type
CustomShow

`get_custom_show_details`(*custom_show_id: str*) → Optional[*CustomShowDetails*]

Get the details of a custom show

Parameters
custom_show_id (*str*) – ID of custom show

Returns
CustomShowDetails object or None

Return type
CustomShowDetails

`get_filler_list`(*filler_list_id: str*) → Optional[*FillerList*]

Get a specific dizqueTV filler list

Parameters
filler_list_id (*str*) – id of filler list

Returns
FillerList object

Return type
FillerList

`get_filler_list_by_name`(*filler_list_name: str*) → Optional[*FillerList*]

Get a specific dizqueTV filler list

Parameters
filler_list_name (*str*) – name of filler list

Returns
FillerList object

Return type
FillerList

`get_filler_list_channels`(*filler_list_id: str*) → List[*Channel*]

Get the channels that a dizqueTV filler list belongs to

Parameters
filler_list_id (*str*) – ID of filler list

Returns
List of Channel objects

Return type
List[*Channel*]

get_filler_list_info(*filler_list_id: str*) → <module 'json' from '/home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/__init__.py'>

Get the name, content and id for a dizqueTV filler list

Parameters
filler_list_id (*str*) – id of filler list

Returns
JSON data with filler list name, content and id

Return type
dict

get_plex_server(*server_name: str*) → Optional[*PlexServer*]

Get a specific Plex Media Server

Parameters
server_name (*str*) – Name of Plex Server

Returns
PlexServer object or None

Return type
PlexServer

get_radio_url(*channel_number: int*) → str

Get URL for only audio (to use for network stream in players like VLC)

Parameters
channel_number (*int*) – Number of channel to stream

Returns
Audio-only URL for channel

Return type
str

get_stream_url(*channel_number: int*, *audio_only: bool = False*) → str

Get URL for stream (to use for network stream in players like VLC)

Parameters

- **channel_number** (*int*) – Number of channel to stream
- **audio_only** (*bool*) – Stream only the audio

Returns
Stream URL for channel

Return type
str

get_video_url(*channel_number: int*) → str

Get URL for video (to use for network stream in players like VLC)

Parameters
channel_number (*int*) – Number of channel to stream

Returns
Video URL for channel

Return type
str

parse_custom_shows_and_non_custom_shows(*items: List*, *non_custom_show_type*, ***kwargs*) → List[Union[*Program*, *CustomShow*]]

plex_server_foreign_status(*server_name: str*) → bool

Parameters
server_name (*str*) – Name of Plex Server

Returns
True if active, False if not active

Return type
bool

plex_server_status(*server_name: str*) → bool

Check if a Plex Media Server is accessible

Parameters
server_name (*str*) – Name of Plex Server

Returns
True if active, False if not active

Return type
bool

refresh_xml() → bool

Force the server to update the xmltv.xml file

Returns
True if successful, False if unsuccessful

Return type
bool

reset_ffmpeg_settings() → bool

Reset dizqueTV's FFMPEG settings to default

Returns
True if successful, False if unsuccessful

Return type
bool

reset_hdhr_settings() → bool

Reset dizqueTV's HDHomeRun settings to default

Returns
True if successful, False if unsuccessful

Return type
bool

reset_plex_settings() → bool

Reset dizqueTV's Plex settings to default

Returns
True if successful, False if unsuccessful

Return type
bool

`reset_xmltv_settings`() → bool

Reset dizqueTV's XMLTV settings to default

Returns
True if successful, False if unsuccessful

Return type
bool

`update_channel`(*channel_number: int*, ***kwargs*) → bool

Edit a dizqueTV channel

Parameters

- **channel_number** (*int*) – Number of channel to update
- **kwargs** – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_custom_show`(*custom_show_id: str*, ***kwargs*) → bool

Edit a dizqueTV custom show

Parameters

- **custom_show_id** (*str*) – ID of CustomShow to update
- **kwargs** – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_ffmpeg_settings`(***kwargs*) → bool

Edit dizqueTV's FFMPEG settings

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_filler_list`(*filler_list_id: str*, ***kwargs*) → bool

Edit a dizqueTV filler list

Parameters

- **filler_list_id** (*str*) – ID of FillerList to update
- **kwargs** – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_hdhr_settings`(***kwargs*) → bool

Edit dizqueTV's HDHomeRun settings

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_plex_server`(*server_name: str*, ***kwargs*) → bool

Edit a Plex Media Server on dizqueTV

Parameters

- **server_name** (*str*) – name of Plex Media Server to update
- **kwargs** – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_plex_settings`(***kwargs*) → bool

Edit dizqueTV's Plex settings

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`update_xmltv_settings`(***kwargs*) → bool

Edit dizqueTV's XMLTV settings

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

`upload_image`(*image_file_path: str*) → Optional[UploadImageResponse]

Upload an image to dizqueTV

Parameters
image_file_path (*str*) – path of image to upload

Returns
UploadImageResponse object or None

Return type
UploadImageResponse

property channel_count: int

Get the number of dizqueTV channels

Returns
Int number of channels

Return type
int

property channel_numbers: List[int]

Get all dizqueTV channel numbers

Returns
List of channel numbers

Return type
List[int]

property channels: List[*Channel*]

Get all dizqueTV channels

Returns
List of Channel objects

Return type
List[*Channel*]

property custom_shows: List[*CustomShow*]

Get a list of all custom shows

Returns
List of CustomShow objects

Return type
List[*CustomShow*]

property dizquetv_server_details: *ServerDetails*

Get dizqueTV server details

Returns
ServerDetails object

Return type
ServerDetails

property dizquetv_version: str

Get dizqueTV version number

Returns
dizqueTV version number

Return type
str

property ffmpeg_settings: Optional[*FFMPEGSettings*]

Get dizqueTV's FFMPEG settings

Returns
FFMPEGSettings object or None

Return type
FFMPEGSettings

property ffmpeg_version: str

Get FFmpeg version number

Returns
FFmpeg version number

Return type
str

property filler_lists: List[*FillerList*]

Get all dizqueTV filler lists

Returns
List of FillerList objects

Return type
List[*FillerList*]

property guide: *Guide*

Get the dizqueTV guide

Returns
dizqueTV.Guide object

Return type
Guide

property guide_channel_numbers: List[str]

Get the list of channel numbers from the guide

Returns
List of strings (not ints)

Return type
List[str]

property guide_lineup_json: <module 'json' from '/home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/__init__.py'>

Get the raw guide JSON data

Returns
JSON data

Return type
dict

property hdhr_settings: Optional[*HDHomeRunSettings*]

Get dizqueTV's HDHomeRun settings

Returns
HDHomeRunSettings object or None

Return type
HDHomeRunSettings

property highest_channel_number: int

Get the highest active channel number

Returns
Int number of the highest active channel

Return type
int

`property hls_m3u: M3U8`

Get dizqueTV's hls.m3u playlist Without m3u8, this method currently produces an error.

Returns
m3u8 object

Return type
m3u8.model.M3U8

`property last_guide_update: Optional[datetime]`

Get the last update time for the guide

Returns
datetime.datetime object

Return type
datetime

`property last_xmltv_refresh: str`

Get the last time the XMLTV file was refreshed

Returns
Timestamp of last refresh

Return type
str

`property lowest_available_channel_number: int`

Get the lowest channel number that doesn't currently exist

Returns
Int number of the lowest available channel

Return type
int

`property lowest_channel_number: int`

Get the lowest active channel number

Returns
Int number of the lowest active channel

Return type
int

`property m3u: M3U8`

Get dizqueTV's m3u playlist Without m3u8, this method currently produces an error.

Returns
m3u8 object

Return type
m3u8.model.M3U8

property nodejs_version: str

Get Node.js version number

Returns
Node.js version number

Return type
str

property plex_servers: List[*PlexServer*]

Get the Plex Media Servers connected to dizqueTV

Returns
List of PlexServer objects

Return type
List[*PlexServer*]

property plex_settings: Optional[*PlexSettings*]

Get dizqueTV's Plex settings

Returns
PlexSettings object or None

Return type
PlexSettings

property xmltv_settings: Optional[*XMLTVSettings*]

Get dizqueTV's XMLTV settings

Returns
XMLTVSettings object or None

Return type
XMLTVSettings

property xmltv_xml: Optional[Element]

Get dizqueTV's XMLTV data

Returns
xml.etree.ElementTree.Element object or None

Return type
ElementTree.Element

dizqueTV.dizquetv.**convert_custom_show_to_programs**(*custom_show:* CustomShow, *dizque_instance*) → List[*Program*]

Convert a CustomShow into a list of Program objects

Parameters

- **custom_show** (CustomShow) – CustomShow object to convert
- **dizque_instance** (API) – dizqueTV API instance

Returns
List of Program objects

Return type
list

dizqueTV.dizquetv.**convert_plex_item_to_filler_item**(*plex_item: Union[Video, Movie, Episode, Track]*, *plex_server: PlexServer*) → *FillerItem*

Convert a PlexAPI Video, Movie, Episode or Track object into a FillerItem

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]*) – plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track object
- **plex_server** (*plexapi.server.PlexServer*) – plexapi.server.PlexServer object

Returns
Program object

Return type
Program

dizqueTV.dizquetv.**convert_plex_item_to_program**(*plex_item: Union[Video, Movie, Episode, Track]*, *plex_server: PlexServer*) → *Program*

Convert a PlexAPI Video, Movie, Episode or Track object into a Program

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]*) – plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track object
- **plex_server** (*plexapi.server.PlexServer*) – plexapi.server.PlexServer object

Returns
Program object

Return type
Program

dizqueTV.dizquetv.**convert_plex_server_to_dizque_plex_server**(*plex_server: PlexServer*) → *PlexServer*

Convert a plexapi.PlexServer object to a dizqueTV PlexServer object

Parameters
plex_server (*plexapi.server.PlexServer*) – plexapi.PlexServer object to convert

Returns
PlexServer object

Return type
PlexServer

dizqueTV.dizquetv.**convert_program_to_custom_show_item**(*program:* Program, *dizque_instance*) → *CustomShowItem*

Convert a dizqueTV Program to a dizqueTV CustomShowItem (add durationStr and commercials)

Parameters

- **program** (Program) – Program to convert
- **dizque_instance** (API) – dizqueTV API instance

Returns
CustomShowItem

Return type
CustomShowItem

dizqueTV.dizquetv.**expand_custom_show_items**(*programs: List[Union[*Program*, *Redirect*, *FillerItem*, *CustomShow*, Video, Movie, Episode, Track]]*, *dizque_instance*) → List[Union[*Program*, *Redirect*, *FillerItem*, Video, Movie, Episode, Track]]

Expand all custom shows in a list out to their individual programs

Parameters

- **programs** (*list*) – List of programs (i.e. Program, Movie, Video, Track, CustomShow)
- **dizque_instance** (API) – DizqueTV API instance

Returns
list of all programs (including custom show programs)

Return type
list

dizqueTV.dizquetv.**fill_in_watermark_settings**(*handle_errors: bool = True*, ***kwargs*) → dict

Create complete watermark settings

Parameters

- **handle_errors** (*bool*) – whether to handle errors or not
- **kwargs** – All kwargs, including some related to watermark

Returns
A complete and valid watermark dict

Return type
dict

dizqueTV.dizquetv.**make_time_slot_from_dizque_program**(*program: Union[*Program*, *Redirect*]*, *time: str*, *order: str*) → Optional[*TimeSlot*]

Convert a DizqueTV Program or Redirect into a TimeSlot object for use in scheduling

Parameters

- **program** (*Union[Program, Redirect]*) – Program or Redirect object
- **time** (*str*) – time for time slot
- **order** (*str*) – order ('shuffle' or 'next') for time slot

Returns
TimeSlot object

Return type
TimeSlot

dizqueTV.dizquetv.**repeat_and_shuffle_list**(*items: List*, *how_many_times: int*) → List

Repeat items in a list, shuffled, x number of times. Items will be shuffled in each repeat group. Ex. [A, B, C] x3 -> [A, B, C, B, A, C, C, A, B]

Parameters

- **items** (*list*) – list of items to repeat
- **how_many_times** (*int*) – how many times the list should repeat

Returns
repeated list

Return type
list

dizqueTV.dizquetv.**repeat_list**(*items: List*, *how_many_times: int*) → List

Repeat items in a list x number of times. Items will remain in the same order. Ex. [A, B, C] x3 -> [A, B, C, A, B, C, A, B, C]

Parameters

- **items** (*list*) – list of items to repeat
- **how_many_times** (*int*) – how many times the list should repeat

Returns
repeated list

Return type
list

3.2 Channels

class dizqueTV.models.channels.**Channel**(*data: dict*, *dizque_instance*, *plex_server: Optional[PlexServer] = None*)

Bases: BaseAPIObject

add_channel_at_night(*night_channel_number: int*, *start_hour: int*, *end_hour: int*) → bool

Add a Channel at Night to a dizqueTV channel

Parameters

- **night_channel_number** (*int*) – number of the channel to redirect to
- **start_hour** (*int*) – hour (in 24-hour time) to start the redirect
- **end_hour** (*int*) – hour (in 24-hour time) to end the redirect

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

add_channel_at_night_alt(*night_channel_number: int*, *start_hour: int*, *end_hour: int*) → bool

Add a Channel at Night to a dizqueTV channel

Parameters

- **night_channel_number** (*int*) – number of the channel to redirect to
- **start_hour** (*int*) – hour (in 24-hour time) to start the redirect
- **end_hour** (*int*) – hour (in 24-hour time) to end the redirect

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

add_collection(*collection: Collection*, *plex_server: PlexServer*) → bool

Add a collection to this channel

Parameters

- **collection** (*plexapi.collection.Collection*) – PlexAPI Collection to add to this channel
- **plex_server** (*plexapi.server.PlexServer*) – plexapi.server.PlexServer object

Returns
True if successful, False if unsuccessful (Channel reloads in place)

Return type
bool

add_filler_list(*filler_list:* FillerList = *None*, *filler_list_id: str = None*, *weight: int = 300*, *cooldown: int = 0*) → bool

Add a filler list to this channel

Parameters

- **filler_list** (FillerList, *optional*) – FillerList object (optional)
- **filler_list_id** (*str, optional*) – ID of FillerList (optional)
- **weight** (*int, optional*) – weight to assign list in channel (default: 300)
- **cooldown** (*int, optional*) – cooldown to assign list in channel (default: 0)

Returns
True if successful, False if unsuccessful (Channel reloads in place)

Return type
bool

add_playlist(*playlist: Playlist*, *plex_server: PlexServer*) → bool

Add a playlist to this channel

Parameters

- **playlist** (*plexapi.playlist.Playlist*) – PlexAPI Playlist to add to this channel
- **plex_server** (*plexapi.server.PlexServer*) – plexapi.server.PlexServer object

Returns
True if successful, False if unsuccessful (Channel reloads in place)

Return type
bool

add_program(*plex_item: Union[Video, Movie, Episode, Track] = None*, *plex_server: PlexServer = None*, *program: Union[*Program, CustomShow*] = None*, ***kwargs*) → bool

Add a program to this channel

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track], optional*) – plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track object (optional)
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (optional)
- **program** (Program, *optional*) – Program object (optional)

- **kwargs** – keyword arguments of Program settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in place)

Return type
bool

add_programs(*programs: List[Union[*Program*,* Redirect*,* FillerItem*,* CustomShow*, Video, Movie, Episode, Track]], plex_server: PlexServer = None*) → bool

Add multiple programs to this channel

Parameters

- **programs** (*List[Union[*Program*,* CustomShow*, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]]*) – List of Program, CustomShow plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track objects
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (required if adding PlexAPI Video, Movie, Episode or Track objects)

Returns
True if successful, False if unsuccessful (Channel reloads in place)

Return type
bool

add_random_schedule(*time_slots: List[*TimeSlot*]*, ***kwargs*) → bool

Add a random schedule to this channel

Parameters

- **time_slots** (*List[*TimeSlot*]*) – List of TimeSlot objects
- **kwargs** – keyword arguments for schedule settings

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

add_reruns(*start_time: datetime*, *length_hours: int*, *times_to_repeat: int*) → bool

Add a block of reruns to a dizqueTV channel

Parameters

- **start_time** (*datetime.datetime*) – datetime.datetime object, what time the reruns start
- **length_hours** (*int*) – how long the block of reruns should be
- **times_to_repeat** (*int*) – how many times to repeat the block of reruns

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

add_schedule(*time_slots: List[TimeSlot]*, ***kwargs*) → bool

Add a schedule to this channel

Parameters

- **time_slots** (*List[TimeSlot]*) – List of TimeSlot objects
- **kwargs** – keyword arguments for schedule settings

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

add_x_duration_of_show_episodes(*duration_in_milliseconds: int*, *list_of_episodes: List[Union[Program, Episode]]*, *plex_server: PlexServer = None*, *allow_overtime: bool = False*) → bool

Add an X duration of items from a list of programs to a dizqueTV channel

Parameters

- **duration_in_milliseconds** (*int*) – length of time to add
- **list_of_episodes** (*List[Union[Program, plexapi.video.Episode]]*) – list of Program or plexapi.media.Episode objects
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer, needed if adding plexapi.media.Episode objects
- **allow_overtime** – Allow adding one more episode, even if total time would go over.

Otherwise, don't add any more if total time would exceed duration_in_milliseconds (default: False) :type allow_overtime: bool, optional :return: True if successful, False if unsuccessful (Channel reloads in-place) :rtype: bool

add_x_number_of_show_episodes(*number_of_episodes: int*, *list_of_episodes: List[Union[Program, Episode]]*, *plex_server: PlexServer = None*) → bool

Add the first X number of items from a list of programs to a dizqueTV channel

Parameters

- **number_of_episodes** (*int*) – number of items to add from the list
- **list_of_episodes** (*List[Union[Program, plexapi.video.Episode]]*) – list of Program or plexapi.media.Episode objects
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer, needed if adding plexapi.media.Episode objects

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

balance_programs(*margin_of_error: float = 0.1*) → bool

Balance shows to the shortest show length. Movies unaffected.

Parameters
margin_of_error (*float, optional*) – (Optional) Specify margin of error when deciding whether to add another episode. Ex. margin_of_error = 0.1 -> If adding a new episode would eclipse the shortest show length by 10% or less, add the episode.

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

`block_shuffle`(*block_length: int*, *randomize: bool = False*) → bool

Sort TV shows on this channel cyclically

Parameters

- **block_length** (*int*) – Length of each block
- **randomize** (*bool, optional*) – Random block lengths between 1 and block_length

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

`cyclical_shuffle`() → bool

Sort TV shows on this channel cyclically

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

`delete`() → bool

Delete this channel

Returns
True if successful, False if unsuccessful

Return type
bool

`delete_all_filler_lists`() → bool

Delete all filler lists from this channel

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

`delete_all_programs`() → bool

Delete all programs from this channel

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

`delete_filler_list`(*filler_list:* FillerList *= None*, *filler_list_id: str = None*) → bool

Delete a program from this channel

Parameters

- **filler_list** (FillerList, *optional*) – FillerList object to delete

- **filler_list_id** (*str, optional*) – ID of filler list to delete

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

delete_program(*program:* Program) → bool

Delete a program from this channel

Parameters
program (Program) – Program object to delete

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

delete_schedule() → bool

Delete this channel's Schedule Removes all offline times, removes duplicate programs (and all redirects), random shuffles remaining items

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

delete_show(*show_name: str*, *season_number: int = None*) → bool

Delete all episodes of a specific show

Parameters

- **show_name** (*str*) – Name of show to delete
- **season_number** (*int, optional*) – (Optional) Number of season to delete

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

edit(***kwargs*) → bool

Alias for channels.update()

Parameters
kwargs – keyword arguments of Channel settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

fast_forward(*seconds: int = 0*, *minutes: int = 0*, *hours: int = 0*, *days: int = 0*, *months: int = 0*, *years: int = 0*) → bool

Fast forward the channel start time by an amount of time

Parameters

- **seconds** (*int, optional*) – how many seconds

- **minutes** (*int, optional*) – how many minutes
- **hours** (*int, optional*) – how many hours
- **days** (*int, optional*) – how many days
- **months** (*int, optional*) – how many months (assume 30 days in month)
- **years** (*int, optional*) – how many years (assume 365 days in year)

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

get_filler_list(*filler_list_title: str*) → Optional[*FillerList*]

Get a specific filler list on this channel

Parameters
filler_list_title (*str*) – Title of filler list

Returns
FillerList object or None

Return type
FillerList

get_program(*program_title: str = None*, *redirect_channel_number: int = None*) → Optional[*Program*]

Get a specific program on this channel

Parameters

- **program_title** (*str, optional*) – Title of program
- **redirect_channel_number** (*int, optional*) – Channel number for Redirect object (use if getting Redirect instead of Program)

Returns
Program object or None

Return type
Program

pad_times(*start_every_x_minutes: int*) → bool

Add padding between programs on a channel, so programs start at specific intervals

Parameters
start_every_x_minutes – Programs start every X minutes past the hour

(ex. 10 for :00, :10, :20, :30, :40 & :50; 15 for :00, :15, :30 & :45; 20 for :00, :20 & :40; 30 for :00 & :30; 60 or 0 for :00) :type start_every_x_minutes: int :return: True if successful, False if unsuccessful (Channel reloads in-place) :rtype: bool

refresh()

Reload current Channel object Use to update program and filler data

Returns
None

remove_duplicate_programs() → bool

Delete duplicate programs on this channel NOTE: Removes all redirects

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

remove_duplicate_redirects() → bool

Delete duplicate redirects on this channel

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

remove_redirects() → bool

Delete all redirects from a channel, preserving offline times, programs and filler items

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

remove_specials() → bool

Delete all specials from this channel Note: Removes all redirects

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

replicate(*how_many_times: int*) → bool

Replicate/repeat the channel lineup x number of times Items will remain in the same order. Ex. [A, B, C] x3 -> [A, B, C, A, B, C, A, B, C]

Parameters
how_many_times (*int*) – how many times to repeat the lineup

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

replicate_and_shuffle(*how_many_times: int*) → bool

Replicate/repeat the channel lineup, shuffled, x number of times Items will be shuffled in each repeat group. Ex. [A, B, C] x3 -> [A, B, C, B, A, C, C, A, B]

Parameters
how_many_times (*int*) – how many times to repeat the lineup

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

rewind(*seconds: int = 0*, *minutes: int = 0*, *hours: int = 0*, *days: int = 0*, *months: int = 0*, *years: int = 0*) → bool

Fast forward the channel start time by an amount of time

Parameters

- **seconds** (*int, optional*) – how many seconds
- **minutes** (*int, optional*) – how many minutes
- **hours** (*int, optional*) – how many hours
- **days** (*int, optional*) – how many days
- **months** (*int, optional*) – how many months (assume 30 days in month)
- **years** (*int, optional*) – how many years (assume 365 days in year)

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

sort_programs_alphabetically() → bool

Sort all programs on this channel in alphabetical order

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

sort_programs_by_duration() → bool

Sort all programs on this channel by duration

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

sort_programs_by_release_date() → bool

Sort all programs on this channel by release date

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

sort_programs_by_season_order() → bool

Sort all programs on this channel by season order Movies are added at the end of the list

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

sort_programs_randomly() → bool

Sort all programs on this channel randomly

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

update(***kwargs*) → bool

Edit this Channel on dizqueTV Automatically refreshes current Channel object

Parameters
kwargs – keyword arguments of Channel settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

update_program(*program:* Program, ***kwargs*) → bool

Update a program from this channel

Parameters

- **program** (Program) – Program object to update
- **kwargs** – Keyword arguments of new Program settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

update_schedule(***kwargs*) → bool

Update the schedule for this channel

Parameters
kwargs – keyword arguments for schedule settings (slots data included if needed)

(include random=true to use random time slots) :return: True if successful, False if unsuccessful (Channel reloads in-place) :rtype: bool

property filler_lists: List[*FillerList*]

Get all filler lists on this channel

Returns
List of FillerList objects

Return type
List[*FillerList*]

property programs: List[Union[*Program*, *CustomShow*]]

Get all programs on this channel

Returns
List of Program and CustomShow objects

Return type
List[Union[*Program*, *CustomShow*]]

property startTime_datetime: datetime

class dizqueTV.models.channels.**ChannelFFMPEGSettings**(*data: dict*, *dizque_instance*, *channel_instance*)

Bases: BaseAPIObject

update(*use_global_settings: bool = False*, ***kwargs*) → bool

Edit this channel's FFMPEG settings on dizqueTV Automatically refreshes associated Channel object

Parameters

- **use_global_settings** (*bool, optional*) – Use global dizqueTV FFMPEG settings (default: False)
- **kwargs** – keyword arguments of Channel FFMPEG settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place, ChannelFFMPEGSettings object is destroyed)

Return type
bool

class dizqueTV.models.channels.**ChannelOnDemandSettings**(*data: dict*, *dizque_instance*, *channel_instance*)

Bases: BaseAPIObject

update(***kwargs*) → bool

Edit this channel's OnDemand settings on dizqueTV Automatically refreshes associated Channel object

Parameters
kwargs – keyword arguments of Channel FFMPEG settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place, ChannelFFMPEGSettings object is destroyed)

Return type
bool

class dizqueTV.models.channels.**Schedule**(*data: dict*, *dizque_instance*, *channel_instance*)

Bases: BaseAPIObject

add_time_slot(*time_slot: Optional[TimeSlot] = None*, *time_string: Optional[str] = None*, ***kwargs*) → bool

Add a time slot to this Schedule

Parameters

- **time_slot** (TimeSlot, *optional*) – TimeSlot object to add (Optional)
- **time_string** (*str, optional*) – time in readable 24-hour format (ex. 00:00:00 = 12:00:00 A.M., 05:15:00 = 5:15 A.M., 20:08:12 = 8:08:12 P.M.) (Optional if time=<milliseconds_since_midnight> not included in kwargs)
- **kwargs** – keyword arguments for a new time slot (time, showId and order)

Returns
True if successful, False if unsuccessful (Channel reloads in-place, this Schedule object is destroyed)

Return type
bool

delete() → bool

Delete this channel's Schedule Removes all duplicate programs, adds random shuffle

Returns
True if successful, False if unsuccessful (Channel reloads in-place, this Schedule object is destroyed)

Return type
bool

delete_time_slot(*time_slot:* TimeSlot) → bool

Delete a time slot from this Schedule

Parameters
time_slot (TimeSlot) – TimeSlot object to remove

Returns
True if successful, False if unsuccessful (Channel reloads in-place, this Schedule object is destroyed)

Return type
bool

edit_time_slot(*time_slot:* TimeSlot, *time_string: Optional[str] = None*, ***kwargs*) → bool

Edit a time slot from this Schedule

Parameters

- **time_slot** (TimeSlot) – TimeSlot object to edit
- **time_string** (*str optional*) – time in readable 24-hour format (ex. 00:00:00 = 12:00:00 A.M., 05:15:00 = 5:15 A.M., 20:08:12 = 8:08:12 P.M.) (Optional if time=<milliseconds_since_midnight> not included in kwargs)
- **kwargs** – Keyword arguments for the edited time slot (time, showId and order)

Returns
True if successful, False if unsuccessful (Channel reloads in-place, this Schedule object is destroyed)

Return type
bool

update(***kwargs*)

Edit this Schedule on dizqueTV Automatically refreshes associated Channel object

Parameters
kwargs – keyword arguments of Schedule settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place, this Schedule object is destroyed)

Return type
bool

class dizqueTV.models.channels.**TimeSlot**(*data: dict*, *program: Optional[TimeSlotItem] = None*, *schedule_instance=None*)

Bases: BaseObject

delete() → bool

Delete this TimeSlot object from the schedule

Returns
True if successful, False if unsuccessful (Channel reloads in-place, this TimeSlot and its parent Schedule object are destroyed)

Return type
bool

edit(*time_string: Optional[str] = None*, ***kwargs*) → bool

Edit this TimeSlot object

Parameters
time_string – time in readable 24-hour format

(ex. 00:00:00 = 12:00:00 A.M., 05:15:00 = 5:15 A.M., 20:08:12 = 8:08:12 P.M.) (Optional if time=<milliseconds_since_midnight> not included in kwargs) :type time_string: str, optional :param kwargs: Keyword arguments for the edited time slot (time, showId and order) :return: True if successful, False if unsuccessful (Channel reloads in-place, this TimeSlot and its parent Schedule object are destroyed) :rtype: bool

class dizqueTV.models.channels.**TimeSlotItem**(*item_type: str*, *item_value: str = ''*)

Bases: object

class dizqueTV.models.channels.**Watermark**(*data: dict*, *dizque_instance*, *channel_instance*)

Bases: BaseAPIObject

update(***kwargs*) → bool

Edit this Watermark on dizqueTV Automatically refreshes associated Channel object

Parameters
kwargs – keyword arguments of Watermark settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place, Watermark object is destroyed)

Return type
bool

3.3 Fillers

class dizqueTV.models.fillers.**FillerList**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

add_filler(*plex_item: Union[Video, Movie, Episode, Track] = None*, *plex_server: PlexServer = None*, *filler:* FillerItem *= None*, ***kwargs*) → bool

Add a filler item to this filler list

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track], optional*) – plexapi.video.Video, plexapi.video.Moviem plexapi.video.Episode or plexapi.audio.Track object (optional)
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (optional)
- **filler** (FillerItem, *optional*) – FillerItem item (optional)
- **kwargs** – keyword arguments of FillerItem settings names and values

Returns
True if successful, False if unsuccessful (FillerList reloads in place)

Return type
bool

add_fillers(*fillers: List[Union[*FillerItem*, *CustomShow*, Video, Movie, Episode, Track]]*, *plex_server: PlexServer = None*) → bool

Add multiple programs to this channel

Parameters

- **fillers** (*List[Union[*FillerItem*, *CustomShow*, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]]*) – List of FillerItem, CustomShow, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track objects
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (required if adding PlexAPI Video, Movie, Episode or Track objects)

Returns
True if successful, False if unsuccessful (Channel reloads in place)

Return type
bool

delete() → bool

Delete this filler list

Returns
True if successful, False if unsuccessful

Return type
bool

delete_all_fillers() → bool

Delete all filler items from this filler list

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

delete_filler(*filler:* FillerItem) → bool

Delete a filler item from this filler list

Parameters
filler (FillerItem) – FillerItem object to delete

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

get_filler_item(*filler_item_title: str*) → Optional[*FillerItem*]

Get a specific program on this channel

Parameters
filler_item_title (*str, optional*) – Title of filler item

Returns
FillerItem object or None

Return type
FillerItem

`refresh()`

Reload current FillerList object Use to update filler data

Returns
None

Return type
None

`remove_duplicate_fillers()` → bool

Delete duplicate filler items on this filler list

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

`sort_filler_by_duration()` → bool

Sort all filler items on this filler list by duration

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

`sort_filler_randomly()` → bool

Sort all filler items on this filler list randomly

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

`update`(***kwargs*) → bool

Edit this FillerList on dizqueTV Automatically refreshes current FillerList object

Parameters
kwargs – keyword arguments of FillerList settings names and values

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

`update_filler`(*filler:* FillerItem, ***kwargs*) → bool

Update a filler item on this filler list

Parameters

- **filler** (FillerItem) – FillerItem object to update
- **kwargs** – Keyword arguments of new FillerItem settings names and values

Returns
True if successful, False if unsuccessful (FillerList reloads in-place)

Return type
bool

property channels: List

Get all channels this filler list is used on

Returns
List of Channel objects

Return type
List[*Channel*]

property content: List[Union[*FillerItem*, *CustomShow*]]

Get all filler items on this list

Returns
List of FillerItem and CustomShow objects

Return type
List[Union[*FillerItem*, *CustomShow*]]

property details: dict

Get FillerList JSON

Returns
JSON data for FillerList object

Return type
dict

3.4 Media

class dizqueTV.models.media.**BaseMediaItem**(*data: dict*, *dizque_instance*, *channel_instance=None*)

Bases: BaseAPIObject

class dizqueTV.models.media.**FillerItem**(*data: dict*, *dizque_instance*, *filler_list_instance*)

Bases: *MediaItem*

delete() → bool

Delete this filler

Returns
True if successful, False if unsuccessful

Return type
bool

refresh(*data: dict = None*, *filler_item_title: str = None*)

Reload current FillerItem object Use to update data

Returns
None

update(***kwargs*) → bool

Update this filler

Returns
True if successful, False if unsuccessful

Return type
bool

class dizqueTV.models.media.**MediaItem**(*data: dict*, *dizque_instance*, *channel_instance=None*)

Bases: *BaseMediaItem*

property full_name

class dizqueTV.models.media.**Program**(*data: dict*, *dizque_instance*, *channel_instance*)

Bases: *MediaItem*, *Redirect*

delete() → bool

Delete this program

Returns
True if successful, False if unsuccessful

Return type
bool

refresh(*data: dict = None*, *program_title: str = None*, *redirect_channel_number: int = None*)

Reload current Program object Use to update data

Returns
None

update(***kwargs*) → bool

Update this program

Returns
True if successful, False if unsuccessful

Return type
bool

class dizqueTV.models.media.**Redirect**(*data: dict*, *dizque_instance*, *channel_instance*)

Bases: *BaseMediaItem*

3.5 Custom Show

class dizqueTV.models.custom_show.**CustomShow**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

add_program(*plex_item: Union[Video, Movie, Episode, Track] = None*, *plex_server: PlexServer = None*, *program: Union[Program, CustomShowItem] = None*)

Add a program to this custom show

Parameters

- **plex_item** (*Union[plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track], optional*) – plexapi.video.Video, plexapi.video.Moviem plexapi.video.Episode or plexapi.audio.Track object (optional)
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (optional)
- **program** (*Union[Program, CustomShowItem], optional*) – Program or CustomShowItem object (optional)

Returns
True if successful, False if unsuccessful (CustomShow reloads in place)

Return type
bool

add_programs(*programs: List[Union[*Program*,* CustomShowItem*, Video, Movie, Episode, Track]] = None*, *plex_server: PlexServer = None*)

Add multiple programs to this custom show

Parameters

- **programs** (*List[Union[*Program*,* CustomShowItem*, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode, plexapi.audio.Track]]*) – List of Program, CustomShowItem, plexapi.video.Video, plexapi.video.Movie, plexapi.video.Episode or plexapi.audio.Track objects
- **plex_server** (*plexapi.server.PlexServer, optional*) – plexapi.server.PlexServer object (required if adding PlexAPI Video, Movie, Episode or Track objects)

Returns
True if successful, False if unsuccessful (CustomShow reloads in place)

Return type
bool

delete() → bool

Delete this custom show

Returns
True if successful, False if unsuccessful

Return type
bool

delete_all_programs() → bool

Delete all custom show items from this custom show

Returns
True if successful, False if unsuccessful (CustomShow reloads in-place)

Return type
bool

delete_program(*program: Union[*Program*,* CustomShowItem*]*) → bool

Delete a custom show item from this custom show

Parameters
program (CustomShowItem *or* Program) – CustomShowItem or Program object to delete

Returns
True if successful, False if unsuccessful (CustomShow reloads in-place)

Return type
bool

edit(***kwargs*) → bool

Alias for custom_show.update()

Parameters
kwargs – keyword arguments of CustomShow settings names and values

Returns
True if successful, False if unsuccessful (Channel reloads in-place)

Return type
bool

`refresh()`

Reload current CustomShow object Use to update program data

Returns
None

`remove_duplicate_fillers()` → bool

Delete duplicate custom show items on this custom show

Returns
True if successful, False if unsuccessful (CustomShow reloads in-place)

Return type
bool

`sort_filler_by_duration()` → bool

Sort all custom show items on this custom show by duration

Returns
True if successful, False if unsuccessful (CustomShow reloads in-place)

Return type
bool

`sort_filler_randomly()` → bool

Sort all custom show items on this custom show randomly

Returns
True if successful, False if unsuccessful (CustomShow reloads in-place)

Return type
bool

`update`(***kwargs*) → bool

Edit this CustomShow on dizqueTV Automatically refreshes current CustomShow object

Parameters
kwargs – keyword arguments of CustomShow settings names and values

Returns
True if successful, False if unsuccessful (CustomShow reloads in-place)

Return type
bool

`property content: List[`*CustomShowItem*`]`

Get the custom show's content (the actual programs)

Returns
list of CustomShowItem objects

Return type
list

`property details: Optional[`*CustomShowDetails*`]`

Get the custom show's details

Returns
CustomShowDetails object

Return type
CustomShowDetails

class dizqueTV.models.custom_show.**CustomShowDetails**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

property content: List[*CustomShowItem*]

Get the custom show's content (the actual programs)

Returns
list of CustomShowItem objects

Return type
list

class dizqueTV.models.custom_show.**CustomShowItem**(*data: dict*, *dizque_instance*, *order: int*)

Bases: *Program*

property commercials: List

Get the show's commercials

Returns
List of commercials

Return type
list

3.6 Guide

class dizqueTV.models.guide.**Guide**(*data*, *dizque_instance*)

Bases: BaseAPIObject

property last_update: Optional[datetime]

Get the last update time for the guide

Returns
datetime.datetime object

Return type
datetime.datetime

class dizqueTV.models.guide.**GuideChannel**(*data*, *programs*, *dizque_instance*)

Bases: BaseAPIObject

get_lineup(*from_date: datetime*, *to_date: datetime*) → List[*GuideProgram*]

Get guide channel lineup for a certain time range

Parameters

- **from_date** (*datetime.datetime*) – datetime.datetime object to start time frame
- **to_date** (*datetime.datetime*) – datetime.datetime object to end time frame

Returns
list of GuideProgram objects

Return type
list[*GuideProgram*]

class dizqueTV.models.guide.**GuideProgram**(*data*)

Bases: BaseObject

3.7 Plex Server

class dizqueTV.models.plex_server.**PlexServer**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

delete() → bool

Remove this Plex Media Server from dizqueTV

Returns
True if successful, False if unsuccessful

Return type
bool

refresh()

Reload this Plex Media Server

Returns
None

Return type
None

update(***kwargs*) → bool

Edit this Plex Media Server on dizqueTV Automatically refreshes current PlexServer object

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

property foreign_status: bool

Returns
True if active, False if not active

Return type
bool

property status: bool

Check if this Plex Media Server is accessible

Returns
True if active, False if not active

Return type
bool

3.8 Settings

class dizqueTV.models.settings.**FFMPEGSettings**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

refresh()

Reload current FFMPEGSettings object

Returns
None

Return type
None

reset() → bool

Reset these FFMPEG settings Automatically refreshes current FFMPEGSettings object

Returns
True if successful, False if unsuccessful

Return type
bool

update(****kwargs*) → bool

Edit these FFMPEG settings Automatically refreshes current FFMPEGSettings object

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

class dizqueTV.models.settings.**HDHomeRunSettings**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

refresh()

Reload current HDHomeRunSettings object

Returns
None

Return type
None

reset() → bool

Reset these HDHomeRun settings Automatically refreshes current HDHomeRunSettings object

Returns
True if successful, False if unsuccessful

Return type
bool

update(****kwargs*) → bool

Edit these HDHomeRun settings Automatically refreshes current HDHomeRunSettings object

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

class dizqueTV.models.settings.**PlexSettings**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

refresh()

Reload current PlexSettings object

Returns
None

Return type
None

reset() → bool

Reset these Plex settings Automatically refreshes current PlexSettings object

Returns
True if successful, False if unsuccessful

Return type
bool

update(***kwargs*) → bool

Edit these Plex settings Automatically refreshes current PlexSettings object

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

class dizqueTV.models.settings.**ServerDetails**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

reload()

Reload current ServerDetails object

Returns
None

Return type
None

class dizqueTV.models.settings.**XMLTVSettings**(*data: dict*, *dizque_instance*)

Bases: BaseAPIObject

reload()

Reload current XMLTVSettings object

Returns
None

Return type
None

reset() → bool

Reset these XMLTV settings Automatically refreshes current XMLTVSettings object

Returns
True if successful, False if unsuccessful

Return type
bool

update(***kwargs*) → bool

Edit these XMLTV settings Automatically refreshes current XMLTVSettings object

Parameters
kwargs – keyword arguments of setting names and values

Returns
True if successful, False if unsuccessful

Return type
bool

3.9 Exceptions

exception dizqueTV.exceptions.**ChannelCreationError**(*message: str*)

Bases: *IncludeFunctionName*

exception dizqueTV.exceptions.**GeneralException**(*message: str*)

Bases: *IncludeFunctionName*

exception dizqueTV.exceptions.**IncludeFunctionName**(*message: str*, *errored_function: Optional[str] = None*)

Bases: Exception

exception dizqueTV.exceptions.**ItemCreationError**(*message: str*)

Bases: *IncludeFunctionName*

exception dizqueTV.exceptions.**MissingParametersError**(*message: str*)

Bases: *IncludeFunctionName*

exception dizqueTV.exceptions.**MissingSettingsError**(*message: str*)

Bases: *IncludeFunctionName*

exception dizqueTV.exceptions.**NotRemoteObjectError**(*object_type: str*)

Bases: *IncludeFunctionName*

exception dizqueTV.exceptions.**ReportedException**(*message: str*, *send_analytics: bool = True*, *dtv_api_object=None*, *analytics: Optional[GoogleAnalytics] = None*)

Bases: *IncludeFunctionName*

3.10 Plex Utilities

class dizqueTV.plex_utils.**PlexSearch**(*url: Optional[str] = None*, *token: Optional[str] = None*, *plex_server: Optional[PlexServer] = None*)

Bases: *PlexUtils*

search_by_any_keyword_in_title(*keywords: List[str]*, *section_names: Optional[List[str]] = None*) → List[Media]

Find Plex media items with any of the keywords in their title

Parameters

- **keywords** (*List[str]*) – List of keywords to search for. If any of these keywords are found in the title, the item is considered a match.
- **section_names** (*List[str]*) – Names of Plex library sections to search for item, optional

Returns
Matching Plex media item or None

Return type
plexapi.media.Media | None

search_by_any_keywords_in_summary(*keywords: List[str]*, *section_names: Optional[List[str]] = None*) → List[Media]

Find Plex media items with any of the keywords in their summary

Parameters

- **keywords** (*List[str]*) – List of keywords to search for. If any of these keywords are found in the summary, the item is considered a match.
- **section_names** (*List[str]*) – Names of Plex library sections to search for item, optional

Returns
Matching Plex media item or None

Return type
plexapi.media.Media | None

search_by_genre(*genres: List[str]*, *section_names: Optional[List[str]] = None*) → List[Media]

Find Plex media items with any of the keywords in their title

Parameters

- **genres** (*List[str]*) – List of genres to search for. If any of these genres are found on the item, the item is considered a match.
- **section_names** (*List[str]*) – Names of Plex library sections to search for item, optional

Returns
Matching Plex media item or None

Return type
plexapi.media.Media | None

class dizqueTV.plex_utils.**PlexUtils**(*url: str*, *token: str*)

Bases: `object`

create_new_playlist(*playlist_name: str*, *items: List[Media]*) → bool

Create a new Plex playlist

Parameters

- **playlist_name** (*str*) – Name of the new Plex playlist
- **items** (*list[plexapi.media.Media]*) – List of items to add to the playlist

Returns
True if successful, False if unsuccessful

Return type
bool

get_all_section_items(*section: LibrarySection*) → List[Media]

Get all Plex media items in a specific library section NOTE: May be slow on large library sections

Parameters
section (*plexapi.library.LibrarySection*) – Plex library section to load items from

Returns
List of Plex media items

Return type
list[plexapi.media.Media]

get_dizque_item_on_plex(*dizque_item*, *section_name: Optional[str] = None*) → Optional[Media]

Locate a dizqueTV item on Plex

Parameters

- **dizque_item** – dizqueTV item to find on Plex
- **section_name** (*str*) – Name of Plex library section to search for item, optional

Returns
Matching Plex media item or None

Return type
plexapi.media.Media | None

get_playlist(*playlist_name: str*) → Optional[Playlist]

Get a specific Plex playlist

Parameters
playlist_name (*str*) – Name of the Plex playlist

Returns
Plex playlist or None

Return type
plexapi.playlist.Playlist | None

reset_playlist(*playlist_name: str*, *items: List[Media]*) → bool

Reset a Plex playlist (recreate it with new content)

Parameters

- **playlist_name** (*str*) – Name of the Plex playlist
- **items** (*list[plexapi.media.Media]*) – List of items to add to the playlist

Returns
True if successful, False if unsuccessful

Return type
bool

search_for_plex_items(*section_name: Optional[str] = None*, *result_class: Optional[type] = None*, ***search_terms*) → List[Media]

Search for Plex items

Parameters

- **section_name** (*str*) – Name of section to search in, optional
- **result_class** (*Any*) – Type of item to search for, optional
- **search_terms** – keyword arguments of search parameters

Returns
List of matching Plex media items

Return type
List[plexapi.media.Media]

search_for_plex_items_with_advanced_filters(*section_name: Optional[str] = None*, *result_class: Optional[type] = None*, ***search_terms*) → List[Media]

Search for Plex items

Parameters

- **section_name** (*str*) – Name of section to search in, optional
- **result_class** (*Any*) – Type of item to search for, optional
- **search_terms** – keyword arguments of search parameters

Returns
List of matching Plex media items

Return type
List[plexapi.media.Media]

user_has_server_access(*user: MyPlexUser*) → bool

Check if a user has access to a Plex Media Server

Parameters
user (*plexapi.myplex.MyPlexUser*) – User to check access for

Returns
True if user has access, False if user does not have access

Return type
bool

property as_dizquetv_plex_server: dizqueTV.PlexServer

Get this Plex Media Server to a dizqueTV Plex server :return: dizqueTV Plex server :rtype: dizqueTV.PlexServer

property library_sections: List[LibrarySection]

Get all library sections on a Plex Media Server

Returns
list of Plex library sections

Return type
list[plexapi.library.LibrarySection]

`property playlists: List[Playlist]`

Get all playlists on a Plex Media Server

Returns
List of Plex playlists

Return type
list[plexapi.playlist.Playlist]

`property users: List[MyPlexUser]`

Get all users on a Plex Media Server

Returns
List of Plex users

Return type
list[plexapi.myplex.MyPlexUser]

CHAPTER

FOUR

EXAMPLE USES

This library is used in the following projects:

- `dizqueTV-scripts`: https://github.com/nwithan8/dizqueTV-scripts

Documentation of the REST API is available on SwaggerHub: https://app.swaggerhub.com/apis/nwithan8/DizqueTV

CHAPTER

FIVE

CONTRIBUTE

Submit an issue, contribute or just see source code on GitHub: https://github.com/nwithan8/dizqueTV-python

Main dizqueTV project on GitHub: https://github.com/vexorian/dizquetv Wiki: https://github.com/vexorian/dizquetv/wiki

Maybe even buy me a slice of pizza (I don't drink coffee, lol). Your support is greatly appreciated! https://www.buymeacoffee.com/nwithan8

CHAPTER

SIX

CONTACT

Join the dizqueTV Discord server (https://discord.gg/U64P9MR) where I and the other devs tend to hang out.

My Discord username is nwithan8#8438

Follow me on Twitter: @nwithan8

Also feel free to check out my other projects on GitHub or join the `#developer` channel in my own Discord server (https://discord.gg/ygRDVE9)

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX

D

S

T

U

W

X