vlivepy

Release 1.0.2

box-archived

Mar 22, 2021

GETTING STARTED

1 Getting started **3**
- 1.1 Installation . . . 3
- 1.2 Custom variables . . . 4
- 1.3 Exceptions . . . 5

2 Base Models & Objects **7**
- 2.1 vlivepy.model.DataModel . . . 7
- 2.2 vlivepy.model.OfficialVideoModel . . . 8
- 2.3 vlivepy.model.PostModel . . . 10
- 2.4 vlivepy.Channel . . . 13
- 2.5 vlivepy.Comment . . . 15
- 2.6 vlivepy.GroupedBoards . . . 16
- 2.7 vlivepy.Post . . . 17
- 2.8 vlivepy.OfficialVideoPost . . . 17
- 2.9 vlivepy.OfficialVideoLive . . . 18
- 2.10 vlivepy.OfficialVideoVOD . . . 19
- 2.11 vlivepy.Schedule . . . 20
- 2.12 vlivepy.Upcoming . . . 22
- 2.13 vlivepy.UserSession . . . 24

3 Module & Functions **25**
- 3.1 Functions . . . 25
- 3.2 board . . . 27
- 3.3 channel . . . 28
- 3.4 comment . . . 29
- 3.5 connections . . . 32
- 3.6 parser . . . 32
- 3.7 post . . . 34
- 3.8 schedule . . . 34
- 3.9 session . . . 35
- 3.10 upcoming . . . 36
- 3.11 video . . . 37

Python Module Index **41**

Index **43**

vlivepy is a VLIVE(vlive.tv) parser for python.

Easily parse and explore VLIVE with vlivepy.

```
import vlivepy

video = vlivepy.OfficialVideoPost(231176)
print(video.title)
#

print(video.channel_name)
# Rocket Punch
```

CHAPTER

ONE

GETTING STARTED

First, you need to install the vlivepy. Check out *how to install vlivepy*. If you want to customize some settings? Read about *custom variables*.

Oops. You've got trouble? Check all *exceptions* or create an issue. You can also ask for help on the discussion

- **Getting started**: *Installation* | *Custom variables* | *exceptions*
- **Supports**: GitHub Issues | GitHub Discussions

1.1 Installation

This page describes how to install vlivepy. vlivepy can be downloaded from pypi and github.

1.1.1 Install from PyPI

vlivepy can be installed via PyPI.

```
$ python -m pip install vlivepy
```

1.1.2 Get source code

vlivepy is developed and maintained on GitHub. Check out box-archived/vlive-py on GitHub.

You can clone our repository with git command.

```
$ git clone https://github.com/box-archived/vlive-py.git
```

Or just download source code as zip file.

```
$ curl -OL https://github.com/box-archived/vlive-py/archive/main.zip
```

1.1.3 Dependencies

vlivepy uses following pacakges

- requests >= 2.*
- reqWrapper >= 0.2
- beautifulsoup4 >= 4.*

1.2 Custom variables

User can customize variables used to connect with VLIVE

1.2.1 Override variable

User can override variables by import and replace variable that starts with `override_*` from vlivepy.variables package. This affects entire vlivepy workflow

This is the example of overriding user-agent.

```
from vlivepy.variables import override_user_agent

override_user_agent = "<enter custom user agent>"
```

All customizable variables are below

1.2.2 override_gcc

gcc is request parameter to specify user country. VLIVE server chooses cdn by this value.

The default value is `KR`

1.2.3 override_locale

locale is request parameter to specify response language. VLIVE server chooses language of data (e.g Video title) by this value

The default value is `ko_KR`

1.2.4 override_user_agent

user-agent is request header to disguise as web browser. You can get it from your own browser by pasting `console.log(navigator.userAgent)` to web browser's javascript console

The default value is `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36`

1.2.5 override_accept_language

accpet_language is request header to set webpage language. This value affects Upcoming object's language

The default value is `ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7`

1.3 Exceptions

This page describes **Exceptions** of vlivepy

exception `vlivepy.exception.`**APIError**
Common API Error

exception `vlivepy.exception.`**APIJSONParesError**
Failed to parse target

exception `vlivepy.exception.`**APINetworkError**
Failed to load API request

exception `vlivepy.exception.`**APIServerResponseError**
Warning if server response only error

exception `vlivepy.exception.`**APIServerResponseWarning**
Warning if server response with error

exception `vlivepy.exception.`**APISignInFailedError**
Failed to Sign in

exception `vlivepy.exception.`**ModelError**
Common Model Error

exception `vlivepy.exception.`**ModelInitError**
Model Initialize Error

exception `vlivepy.exception.`**ModelRefreshWarning**
Model refresh failed

CHAPTER

TWO

BASE MODELS & OBJECTS

This is the base model for grouping common properties of the objects.

Entire inheritance structure is down below.

- *DataModel*
  - *Channel*
  - *Comment*
  - *GroupedBoards*
  - *OfficialVideoModel*
    - *OfficialVideoLive*
    - *OfficialVideoVOD*
  - *PostModel*
    - *Post*
    - *OfficialVideoPost*
  - *Schedule*
- *Upcoming*
- *UserSession*

2.1 vlivepy.model.DataModel

class vlivepy.model.**DataModel**(*method: Callable, target_id: str, session: Optional[vlivepy.session.UserSession] = None, init_data: Optional[dict] = None*)

Bases: `object`

This is the base object for other class objects. It sends request with method from each modules and caching response.

DataModels and its child objects are able to compare equality. Each objects are considered equal, if their `type` and `target_id` is equal.

Note: This is the base object for other object without independent usage.

Parameters

- **method** (`typing.Callable`) – function for loading data.
- **target_id** (`str`) – argument for *method*.
- **session** (`UserSession`, optional) – session for *method*, defaults to None.
- **init_data** (`dict`, optional) – set initial data instead of loading data, defaults to None.

session
session for method

Type `UserSession`

property raw
Get full data as deep-copied dict.

Return type `dict`

refresh() → None
Reload self data.

property target_id
Get internal target id.

Return type `str`

2.2 vlivepy.model.OfficialVideoModel

class vlivepy.model.**OfficialVideoModel**(*video_seq: Union[str, int], session: Optional[vlivepy.session.UserSession] = None*)

Bases: *vlivepy.model.DataModel*

This is the base object for `OfficialVideoLive` and `OfficialVideoVOD` This contains common property of Live and VOD object.

Note: This is the base object for other object without independent usage.

Parameters

- **video_seq** (`Union[str, int]`) – Unique id(seq) of video.
- **session** (`UserSession`, optional) – Session for loading data with permission, defaults to None.

session
Optional. Session for loading data with permission.

Type `UserSession`

property comment_count
Count of comment in video.

Return type `int`

property created_at
Epoch timestamp about Unknown. The nanosecond units are displayed below the decimal point.

Return type `float`

property expose_status
Exposed-on-website status of video.

Return type bool

property has_live_thumb
Boolean value for having live thumbnail or not.

Return type `bool`

property has_mobile_da
Boolean value for Unknown.

Return type `bool`

property has_notice
Boolean value for having notice.

Return type `bool`

property has_post_ad
Boolean value for having post advertise.

Return type `bool`

property has_pre_ad
Boolean value for having pre advertise.

Return type `bool`

property has_upcoming
Boolean value for having upcoming.

Return type `bool`

property like_count
Count of like received in video.

Return type `int`

multinational_title_get(*locale*) → dict
Get multinational title info by locale.

Parameters **locale** (`str`) – locale to load.

Return type `dict`

multinational_title_locales() → list
Get locales from multinational title.

Return type `list`

property multinational_titles
Title translations.

Return type `List[dict]`

property on_air_start_at
Epoch timestamp about reserved/started on air time. The nanosecond units are displayed below the decimal point.

Return type `float`

property play_count
Count of video play time.

Return type `int`

property product_type
Product type about VLIVE+

Returns "NONE" if the video is normal video. "VLIVE_PLUS" if the video is VLIVE+.

Return type `str`

property screen_orientation
Orientation of video.

Returns "VERTICAL" if the video orientation is vertical. "HORIZONTAL" if the video orientation is horizontal.

Return type str

property thumb
Url of thumbnail.

Return type `str`

property title
Title of video.

Return type `str`

property video_seq
Unique id(seq) of video.

Return type `str`

property video_type
Type of video.

Returns "LIVE" if the video is upcoming/on air live. "VOD" if the video is VOD.

Return type `str`

property vr_content_type
String value for vr content type.

Return type `str`

property will_end_at
Epoch timestamp about reserved end time. The nanosecond units are displayed below the decimal point.

Return type `float`

property will_start_at
Epoch timestamp about Unknown. The nanosecond units are displayed below the decimal point.

Return type `float`

2.3 vlivepy.model.PostModel

class `vlivepy.model.`**`PostModel`**(*post_id: str, session: Optional[vlivepy.session.UserSession] = None*)

Bases: *`vlivepy.model.DataModel`*

This is the base object for `Post` and `OfficialVideoPost` This contains common property of each object

`OfficialVideoPost` has *`OfficialVideoModel`* and doesn't have `body` compared with `Post`

Note: This is the base object for other object without independent usage.

Parameters

- **post_id** (str) – Unique id of post to load.
- **session** (UserSession, optional) – Session for loading data with permission, defaults to None.

session
Optional. Session for loading data with permission.

Type UserSession

property attachments
Detailed attachments data of post.

Return type dict

property attachments_photo
Detailed photo attachments data of post.

Return type dict

property attachments_video
Detailed video attachments data of post.

Return type dict

property author
Detailed author info of post.

Return type dict

property author_id
Unique id of author.

Return type str

property author_nickname
Author nickname.

Return type str

property board_id
Unique id of parent board

Return type int

property channel_code
The code of the channel that contains the post

Return type str

property channel_name
The name of the channel that contains the post

Return type str

property comment_count
Count of its comment

Return type int

property content_type
Type of post.

Returns "POST" if the post is normal Post. "VIDEO" if the post is OfficialVideoPost

Return type `str`

property created_at
Epoch timestamp about created time. The nanosecond units are displayed below the decimal point.

Return type `float`

property emotion_count
Count of received emotion.

Return type `int`

getPostCommentsIter() → Generator[vlivepy.model.Comment, None, None]
Get Its comments as iterable

Return type `Generator[Comment, None, None]`

Yields `Comment`

getPostStarCommentsIter() → Generator[vlivepy.model.Comment, None, None]
Get Its star-comments as iterable

Return type `Generator[Comment, None, None]`

Yields `Comment`

property is_comment_enabled
Boolean value for comment-enabled.

Return type `bool`

property is_hidden_from_star
Boolean value for hidden-from-star.

Return type `bool`

property is_viewer_bookmarked
Boolean value for viewer-bookmarked.

Return type `bool`

property post_id
Unique id of the post.

Return type `str`

property title
Title of the post.

Return type `str`

2.4 vlivepy.Channel

class vlivepy.**Channel**(*channel_code: str*, *session: Optional[vlivepy.session.UserSession] = None*)

Bases: *vlivepy.model.DataModel*

This is the object represents a post of VLIVE

Parameters

- **channel_code** (str) – Unique id of channel.
- **session** (*UserSession*, optional) – Session for loading data with permission, defaults to None.

session

Optional. Session for loading data with permission.

Type *UserSession*

property background_color

Background color(hex) of the channel.

Return type str

property channel_code

Unique id of channel.

Return type str

property channel_cover_image

Cover image url of the channel.

Return type str

property channel_description

Description of the channel.

Return type str

property channel_name

Name of the channel.

Return type str

property channel_profile_image

Profile image url of the channel.

Return type str

decode_channel_code() → int

Decode channel code to unique channel seq

Return type int

groupedBoards() → vlivepy.model.GroupedBoards

Load grouped board list of the channel

Return type *GroupedBoards*

property member_count

Count of members in channel.

Return type int

property open_at

Epoch timestamp about channel opened(created) time.

Returns

property post_count
Count of post in channel.

Return type `int`

property prohibited_word_exact_list
Prohibited word (EXACT) in the channel.

Return type `str`

property prohibited_word_like_list
Prohibited word (LIKE) in the channel.

Return type `str`

property qr_code
QR code image url of the channel.

Return type `str`

property representative_color
Representative color(hex) of the channel.

Return type `str`

property show_upcoming
Boolean value for using upcoming in the channel

Return type `bool`

property sns_share_img
SNS Share image url of the channel.

Return type `str`

property use_member_level
Boolean value for using member level in the channel

Return type `bool`

property video_comment_count
Count of video comment in channel.

Return type `int`

property video_count
Count of video in channel.

Return type `int`

property video_like_count
Count of like in channel.

Return type `int`

property video_play_count
Count of video play times in channel.

Return type `int`

2.5 vlivepy.Comment

class vlivepy.**Comment**(*commentId: str, session: Optional[vlivepy.session.UserSession] = None, init_data: Optional[dict] = None*)

Bases: *vlivepy.model.DataModel*

This is the object represents a comment of VLIVE's post

Parameters

- **commentId** (str) – Unique id of comment to load.
- **session** (*UserSession*, optional) – Session for loading data with permission, defaults to None.
- **init_data** (dict, optional) – set initial data instead of loading data, defaults to None.

session

Optional. Session for loading data with permission.

Type *UserSession*

property author

Detailed author info of post.

Return type dict

property author_memberId

Unique id of author.

Return type str

property author_nickname

Author nickname.

Return type str

property body

Content of comment.

Return type str

property commentId

Unique id of comment.

Return type str

property comment_count

Count of its nested comments.

Return type int

property created_at

Epoch timestamp about created time. The nanosecond units are displayed below the decimal point.

Return type float

property emotion_count

Count of received emotion.

Return type int

getNestedCommentsIter() → Generator[*Comment*]

Get nested comments as iterable (generator).

Return type Generator[Comment]

property parent
Detailed information about parent(upper) item.

Return type `dict`

parent_info_tuple() → tuple
Get parent info as tuple (Parent type, Its(parent) id)

Return type `tuple`

property root
Detailed information about root post.

Return type `dict`

root_info_tuple() → tuple
Get root info as tuple (Root type, Its(root) id)

Return type `tuple`

property sticker
Sticker list of comment.

Return type `list`

property written_in
User language setting of comment.

Return type `str`

2.6 vlivepy.GroupedBoards

class vlivepy.**GroupedBoards**(*channel_code: str*, *session: Optional[vlivepy.session.UserSession] = None*)
Bases: *vlivepy.model.DataModel*

This is the object represents board list of channel.

Parameters

- **channel_code** (`str`) – Unique id of channel.
- **session** (*UserSession*, optional) – Session for loading data with permission, defaults to None.

session
Optional. Session for loading data with permission.

Type *UserSession*

board_names() → List[str]
Get name of the boards

Return type `list`

boards() → List[dict]
Get detailed info of boards

Return type `List[dict]`

groups() → List[str]
Get name of board-groups

Return type `list`

2.7 vlivepy.Post

class vlivepy.**Post**(*post_id: str*, *session: Optional[vlivepy.session.UserSession] = None*)

Bases: *vlivepy.model.PostModel*

This is the object represents a post of VLIVE

Parameters

- **post_id** (str) – Unique id of post to load.
- **session** (*UserSession*, optional) – Session for loading data with permission, defaults to None.

session

Optional. Session for loading data with permission.

Type *UserSession*

property body

Contents of post with <v:attachment> tag

Return type str

formatted_body() → str

Get contents of post with formatting attachments and styles as html.

Return type str

property plain_body

Text-only contents of post

Return type str

property written_in

User language setting of post.

Return type str

2.8 vlivepy.OfficialVideoPost

class vlivepy.**OfficialVideoPost**(*init_id: Union[str, int]*, *session: Optional[vlivepy.session.UserSession] = None*)

Bases: *vlivepy.model.PostModel*

This is the object represents a post of VLIVE

Parameters

- **init_id** (Union[str, int]) – Unique id of post to load. Also, the object can be initialized by video_seq.
- **session** (*UserSession*, optional) – Session for loading data with permission, defaults to None.

session

Optional. Session for loading data with permission.

Type *UserSession*

official_video() → Union[vlivepy.model.OfficialVideoVOD, vlivepy.model.OfficialVideoLive]

Generate *OfficialVideoLive* or *OfficialVideoVOD* object that paired to official video posts

Returns `OfficialVideoVOD`, if the video is VOD.

Returns `OfficialVideoLive`, if the video is Live.

property official_video_type
Type of video.

Returns "LIVE" if the video is upcoming/on air live. "VOD" if the video is VOD.

Return type `str`

property video_seq
Unique id of OfficialVideoPost. (video_seq type)

Return type `str`

2.9 vlivepy.OfficialVideoLive

class vlivepy.**OfficialVideoLive**(*video_seq: Union[int, str], session: Optional[vlivepy.session.UserSession] = None*)

Bases: `vlivepy.model.OfficialVideoModel`

This is the object represents a Live-type-OfficialVideo of VLIVE

Parameters

- **video_seq** (`str`) – Unique id of Live to load.
- **session** (`UserSession`, optional) – Session for loading data with permission, defaults to None.

session
Optional. Session for loading data with permission.

Type `UserSession`

getLivePlayInfo(*silent=False*)
Get play info of live

Parameters **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Return type `dict`

getLiveStatus(*silent=False*)
Get detailed status of live.

Parameters **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Return type `dict`

property has_filter_ad
Boolean value for having filter ad

Return type `bool`

property has_special_live
Boolean value for having special live

Return type `bool`

property hevc
Boolean value for broadcasting with hevc codec.

Return type `bool`

property low_latency
Boolean value for broadcasting with low-latency option

Return type `bool`

property momentable
Boolean value for what user can create moment of the video

Return type `bool`

property pp_type
Unknown boolean value

Return type `bool`

property status
Status of the live

Returns "RESERVED" if the live is reserved to broadcast. "ON_AIR" if the live is going. "ENDED" if the live is ended.

Return type `str`

2.10 vlivepy.OfficialVideoVOD

class vlivepy.**OfficialVideoVOD**(*video_seq: Union[str, int], session: Optional[vlivepy.session.UserSession] = None*)
Bases: `vlivepy.model.OfficialVideoModel`

This is the object represents a VOD-type-OfficialVideo of VLIVE

Parameters

- **video_seq** (`str`) – Unique id of VOD to load.
- **session** (`UserSession`, optional) – Session for loading data with permission, defaults to None.

session
Optional. Session for loading data with permission.

Type `UserSession`

property dimension_type
Unknown value. Server commonly respond "NORMAL"

Return type `str`

property encoding_status
VOD encoding status

Returns "CONVERTING" if the video encoding is in progress. "COMPLETE" if the video encoding is done.

Return type `str`

getInkeyData(*silent: bool = False*) → dict
Get InKey data of video

Parameters **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Return type `dict`

getVodPlayInfo(*silent: bool = False*) → dict
Get VOD play info of video

Parameters **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Return type `dict`

property has_moment
Boolean value for having user-created-moments

Return type `bool`

property has_preview
Boolean value for having 30s preview video

Return type `bool`

property play_time
Count of video play

Return type `int`

recommended_videos(*as_object: bool = False*) → list
Get recommended video list

Parameters **as_object** (`bool`, optional) – Init each item to `OfficialVideoPost`, defaults to False.

Return type `list`

property vod_id
Unique id of VOD that paired with videoSeq

Return type `bool`

property vod_secure_status
Status of DRM protection

Returns "READY" if the DRM is ready but not applied to video. "COMPLETE" if the DRM is applied to video.

Return type `str`

2.11 vlivepy.Schedule

class `vlivepy.`**Schedule**(*schedule_id: str, session: vlivepy.session.UserSession*)
Bases: `vlivepy.model.DataModel`

This is the object represents a post of VLIVE.

Parameters

- **schedule_id** (`Union[str, int]`) – Unique id of schedule to load.
- **session** (`UserSession`) – Session for loading data with permission.

session
Session for loading data with permission.

Type `UserSession`

property author
Detailed author info of post.

Return type `dict`

property author_id
Unique id of author.

Return type `str`

property author_nickname
Author nickname.

Return type `str`

property channel_code
The code of the channel that contains the schedule.

Return type `str`

property channel_name
The name of the channel that contains the post.

Return type `str`

property comment_count
Count of comment in video.

Return type `int`

property emotion_count
Count of received emotion in video.

Return type `int`

official_video() → Union[vlivepy.model.OfficialVideoVOD, vlivepy.model.OfficialVideoLive]
Generate *OfficialVideoLive* or *OfficialVideoVOD* object that paired to schedule

Returns *OfficialVideoVOD*, if the video is VOD.

Returns *OfficialVideoLive*, if the video is Live.

property official_video_type
Type of video.

Returns "LIVE" if the video is upcoming/on air live. "VOD" if the video is VOD.

Return type `str`

property post_id
Post id that paired with the schedule.

Return type `str`

property schedule_id
Unique id of schedule.

Return type `str`

property title
Title of the schedule.

Return type `str`

property video_seq
videoSeq id that paired with the schedule.

Return type `str`

2.12 vlivepy.Upcoming

class `vlivepy.`**`Upcoming`**(*refresh_rate: float = 5*, *show_vod: bool = True*, *show_upcoming_vod: bool = True*, *show_upcoming_live: bool = True*, *show_live: bool = True*)

Bases: `object`

This is the object represents a upcoming list of VLIVE.

This object doesn't use endpoint API but use parsing upcoming webpage. This use refresh rate to caching result. Set *`refresh_rate`* to 0 to disable caching

See also:

This object mainly returns list of `vlivepy.parser.UpcomingVideo`. Check docs!

Parameters

- **`refresh_rate`** (`float`, optional) – Unique id of post to load. Also, the object can be initialized by video_seq. Defaults to 5
- **`show_vod`** (`bool`, optional) – Add VOD to upcoming list, defaults to True.
- **`show_upcoming_vod`** (`bool`, optional) – Add reserved VOD to upcoming list, defaults to True.
- **`show_upcoming_live`** (`bool`, optional) – Add reserved Live to upcoming list, defaults to True.
- **`show_live`** (`bool`, optional) – Add on air live to upcoming list, defaults to True.

`refresh_rate`

Optional. Unique id of post to load. Also, the object can be initialized by video_seq. Defaults to 5

Type `float`

`show_vod`

Optional. Add VOD to upcoming list, defaults to True.

Type `bool`

`show_upcoming_vod`

Optional. Add reserved VOD to upcoming list, defaults to True.

Type `bool`

`show_upcoming_live`

Optional. Add reserved Live to upcoming list, defaults to True.

Type `bool`

`show_live`

Optional. Add on air live to upcoming list, defaults to True.

Type `bool`

`load`(*date: Optional[str]*, *show_vod: Optional[bool] = None*, *show_upcoming_vod: Optional[bool] = None*, *show_upcoming_live: Optional[bool] = None*, *show_live: Optional[bool] = None*, *silent: Optional[bool] = False*) → Optional[List[*vlivepy.upcoming.UpcomingVideo*]]

Get upcoming list data with specific date

Note: Use `upcoming()` instead of using this function to load current upcoming.

Parameters

- **date** (bool) – Specify date to load upcoming.
- **show_vod** (bool, optional) – Add VOD to upcoming list, defaults to `self.show_vod`
- **show_upcoming_vod** (bool, optional) – Add reserved VOD to upcoming list, defaults to `self.show_upcoming_vod`
- **show_upcoming_live** (bool, optional) – Add reserved Live to upcoming list, defaults to `self.show_upcoming_live`
- **show_live** (bool, optional) – Add on air live to upcoming list, defaults to `self.show_live`
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns List of `vlivepy.parser.UpcomingVideo`

refresh (*force: bool = False*) → None

Refresh self data

Parameters **force** (bool, optional) – Force refresh with ignoring refresh rate, defaults to False.

upcoming (*force=False*, *show_vod: Optional[bool] = None*, *show_upcoming_vod: Optional[bool] = None*, *show_upcoming_live: Optional[bool] = None*, *show_live: Optional[bool] = None*) → List[*vlivepy.upcoming.UpcomingVideo*]

Upcoming list with cache life check

Parameters

- **force** (bool, optional) – Force refresh with ignoring refresh rate, defaults to False.
- **show_vod** (bool, optional) – Add VOD to upcoming list, defaults to `self.show_vod`
- **show_upcoming_vod** (bool, optional) – Add reserved VOD to upcoming list, defaults to `self.show_upcoming_vod`
- **show_upcoming_live** (bool, optional) – Add reserved Live to upcoming list, defaults to `self.show_upcoming_live`
- **show_live** (bool, optional) – Add on air live to upcoming list, defaults to `self.show_live`

Returns List of `vlivepy.parser.UpcomingVideo`

2.13 vlivepy.UserSession

class vlivepy.**UserSession**(*email: str*, *pwd: str*)

Bases: object

This is the object for using vlivepy with user permission. You need to use UserSession when you load user-only content (e.g VLIVE+, Membership, etc..)

Email-account info(email, pwd) should be used as login info. This is not working with social login info.

> **Caution:** Too frequent login-try will be banned from VLIVE.
>
> Use *vlivepy.dumpSession()* and *vlivepy.loadSession()* to saving UserSession

Parameters

- **email** (str) – Sign-in email
- **pwd** (str) – Sign-in password

refresh() → None

Reload login data

property session

Get logged-in Session

Return type reqWrapper.Session

CHAPTER

THREE

MODULE & FUNCTIONS

Check more *functions!*. Or you just can import modules and use core functions.

- **Functions**: *Functions*
- **Modules**: *vlivepy.board* | *vlivepy.channel* | *vlivepy.comment* | *vlivepy.connections* | *vlivepy.parser* | *vlivepy.post* | *vlivepy.schedule* | *vlivepy.session* | *vlivepy.upcoming* | *vlivepy.video*

3.1 Functions

This page describes vlivepy's functions.

3.1.1 vlivepy.postIdToVideoSeq()

`vlivepy.`**`postIdToVideoSeq`**(*post_id: str*, *silent=False*) → Optional[str]

Convert post id to videoSeq id

Parameters

- **`post_id`** (`str`) – Post id to convert to videoSeq id.
- **`silent`** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `str`. Paired videoSeq id of the post.

3.1.2 vlivepy.videoSeqToPostId()

`vlivepy.`**`videoSeqToPostId`**(*video_seq: Union[str, int]*, *silent=False*) → Optional[str]

Convert videoSeq id to post id

Parameters

- **`video_seq`** (`str`, optional) – VideoSeq to convert to post id.
- **`silent`** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `str`. Paired post id of the videoSeq.

3.1.3 vlivepy.postTypeDetector()

`vlivepy.postTypeDetector`(*post_id*, *silent=False*)

Check type of the post

Parameters

- **post_id** (`str`, optional) – Unique id of the post to check.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `str`. "POST" if the post is normal Post. "VIDEO" if the post is OfficialVideoPost

3.1.4 vlivepy.decode_channel_code()

`vlivepy.decode_channel_code`(*channel_code: str*, *silent: bool = False*) → Optional[int]

Decode channel code to unique channel seq

Parameters

- **channel_code** (`str`, optional) – Unique id of the post to check.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `int`. Decoded channel code as channel seq.

3.1.5 vlivepy.loadSession()

`vlivepy.loadSession`(*fp*) → vlivepy.session.UserSession

Load UserSession

Parameters **fp** (*Any*) – BufferedReader to read file

Returns *UserSession*

3.1.6 vlivepy.dumpSession()

`vlivepy.dumpSession`(*session: vlivepy.session.UserSession*, *fp*) → None

Dump UserSession

> **Danger:** Dumped UserSession file is unencrypted plain binary. Do not upload/commit dumped file to public place.

Parameters

- **session** (*UserSession*) – UserSession object to dump
- **fp** (*Any*) – BufferedWriter to write file

3.2 board

This page describes **board** module which can be imported as `vlivepy.board`

3.2.1 BoardPostItem

class `vlivepy.board.`**`BoardPostItem`**(*post_id: str, official_video: bool, session: vlivepy.session.UserSession*)

Bases: `object`

This is the object for board post list.

Parameters

- **`post_id`** (`str`) – Unique id of post.
- **`official_video`** (`bool`) – Session for loading data with permission, defaults to None.
- **`session`** (`vlivepy.UserSession`, optional) – Session for loading data with permission.

`session`

Optional. Session for loading data with permission.

Type `vlivepy.UserSession`

`property has_official_video`

Boolean value for having official video

Return type `bool`

`property post_id`

Unique id of post.

Return type `str`

`to_object`() → Union[vlivepy.model.Post, vlivepy.model.OfficialVideoPost]

Initialize matched object from post_id

Returns `vlivepy.Post`, if the post is normal post. `vlivepy.OfficialVideoPost`, if the post is official video

3.2.2 getBoardPosts()

`vlivepy.board.`**`getBoardPosts`**(*board_id: Union[str, int], channel_code: str, session: Optional[vlivepy.session.UserSession] = None, after: Optional[str] = None, latest: bool = False, silent: bool = False*) → Optional[dict]

Get board post from page

Parameters

- **`board_id`** (`str`) – Unique id of the board to load.
- **`channel_code`** (`str`) – Unique id of the channel which contains board.
- **`session`** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **`after`** (`str`, optional) – After parameter to load another page, defaults to None.
- **`latest`** (`bool`, optional) – Load latest post first, defaults to False.

- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data.

3.2.3 getBoardPostsIter()

vlivepy.board.**getBoardPostsIter**(*board_id: Union[str, int], channel_code: str, session: Optional[vlivepy.session.UserSession] = None, latest: bool = False*) → Generator[*vlivepy.board.BoardPostItem*, None, None]

Get board post as iterable (generator).

Parameters

- **board_id** (str) – Unique id of the board to load.
- **channel_code** (str) – Unique id of the channel which contains board.
- **session** (*vlivepy.UserSession*, optional) – Session for loading data with permission, defaults to None.
- **latest** (str, optional) – Load latest post first.

Yields *BoardPostItem*

3.3 channel

This page describes **channel** module which can be imported as vlivepy.channel

3.3.1 getChannelInfo()

vlivepy.channel.**getChannelInfo**(*channel_code: str, session: Optional[vlivepy.session.UserSession] = None, silent: bool = False*) → Optional[dict]

Get detailed Channel info.

Parameters

- **channel_code** (str, optional) – Unique id of channel to load.
- **session** (*vlivepy.UserSession*, optional) – Session for loading data with permission, defaults to None.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed channel data

3.3.2 getGroupedBoards()

`vlivepy.channel.getGroupedBoards`(*channel_code: str, session: Optional[vlivepy.session.UserSession] = None, silent: bool = False*) → Optional[dict]

Get grouped boards info.

Parameters

- **channel_code** (`str`, optional) – Unique id of channel to load boards.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data.

3.4 comment

This page describes **comment** module which can be imported as `vlivepy.channel`

3.4.1 comment_parser()

`vlivepy.comment.comment_parser`(*comment_list: list, session: Optional[vlivepy.session.UserSession] = None*) → list

Parse each comment json data to `vlivepy.Comment` object.

Parameters

- **comment_list** (`list`) – Comment list to parse.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.

Returns List of `vlivepy.Comment`

3.4.2 getCommentData()

`vlivepy.comment.getCommentData`(*comment_id: str, session: Optional[vlivepy.session.UserSession] = None, silent: bool = False*) → Optional[dict]

Get detailed comment data.

Parameters

- **comment_id** (`str`) – Unique id of the comment to load data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.4.3 getNestedComments()

`vlivepy.comment.getNestedComments`(*comment_id: str, session: Optional[vlivepy.session.UserSession] = None, after: Optional[str] = None, silent: bool = False*) → Optional[dict]

Get nested comments of the comment.

Parameters

- **comment_id** (str) – Unique id of the comment to load nested comment.
- **session** (*vlivepy.UserSession*, optional) – Session for loading data with permission, defaults to None.
- **after** (str, optional) – After parameter to load another page, defaults to None.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data.

3.4.4 getNestedCommentsIter()

`vlivepy.comment.getNestedCommentsIter`(*comment_id: str, session: Optional[vlivepy.session.UserSession] = None*)

Get nested comments of the comment as iterable (generator).

Parameters

- **comment_id** (str) – Unique id of the comment to load nested comment.
- **session** (*vlivepy.UserSession*, optional) – Session for loading data with permission, defaults to None.

Return type Generator[*vlivepy.Comment*, None, None]

Yields *vlivepy.Comment*

3.4.5 getPostComments()

`vlivepy.comment.getPostComments`(*post_id: str, session: Optional[vlivepy.session.UserSession] = None, after: Optional[str] = None, silent: bool = False*) → Optional[dict]

Get comments of the post.

Parameters

- **post_id** (str) – Unique id of the post to load comment.
- **session** (*vlivepy.UserSession*, optional) – Session for loading data with permission, defaults to None.
- **after** (str, optional) – After parameter to load another page, defaults to None.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data.

3.4.6 getPostCommentsIter()

`vlivepy.comment.getPostCommentsIter`(*post_id: str, session: Optional[vlivepy.session.UserSession] = None*)

Get comments of post as iterable (generator).

Parameters

- **post_id** (`str`) – Unique id of the post to load comment.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.

Return type Generator[*vlivepy.Comment*, None, None]

Yields `vlivepy.Comment`

3.4.7 getPostStarComments()

`vlivepy.comment.getPostStarComments`(*post_id: str, session: Optional[vlivepy.session.UserSession] = None, after: Optional[str] = None, silent: bool = False*) → Optional[dict]

Get star comments of the post.

Parameters

- **post_id** (`str`) – Unique id of the post to load star comment.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **after** (`str`, optional) – After parameter to load another page, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data.

3.4.8 getPostStarCommentsIter()

`vlivepy.comment.getPostStarCommentsIter`(*post_id: str, session: Optional[vlivepy.session.UserSession] = None*)

Get star comments of post as iterable (generator).

Parameters

- **post_id** (`str`) – Unique id of the post to load star comment.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.

Return type Generator[*vlivepy.Comment*, None, None]

Yields `vlivepy.Comment`

3.5 connections

This page describes **connections** module which can be imported as `vlivepy.connections`

3.5.1 getPostInfo()

`vlivepy.connections.`**`getPostInfo`**(*post_id: str*, *session: Optional[vlivepy.session.UserSession] = None*, *silent: bool = False*) → Optional[dict]

Get detailed post data.

Parameters

- **`post_id`** (`str`) – Unique id of the post to load data.
- **`session`** (*`vlivepy.UserSession`*, optional) – Session for loading data with permission, defaults to None.
- **`silent`** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.5.2 decode_channel_code()

This function has alias of *`vlivepy.decode_channel_code()`*

3.5.3 postIdToVideoSeq()

This function has alias of *`vlivepy.postIdToVideoSeq()`*

3.5.4 videoSeqToPostId()

This function has alias of *`vlivepy.videoSeqToPostId()`*

3.5.5 postTypeDetector()

This function has alias of *`vlivepy.postTypeDetector()`*

3.6 parser

This page describes **parser** module which can be imported as `vlivepy.parser`

3.6.1 format_epoch()

`vlivepy.parser.`**`format_epoch`**(*epoch: Union[int, float], fmt: str*)

This is the function for formatting epoch to string easily.

Parameters

- **epoch** (`int`) – Epoch timestamp.
- **fmt** (`str`) – Format-string to format Epoch.

Returns `str`. Formatted epoch time.

3.6.2 max_res_from_play_info()

`vlivepy.parser.`**`max_res_from_play_info`**(*play_info: dict*) → dict

This is the parser for finding maximum resolution from play info (FVideo, VOD).

Parameters **play_info** (`dict`) – Play info dict to find maximum resolution.

Returns `dict`. Video data that has maximum resolution from play_info

3.6.3 next_page_checker()

`vlivepy.parser.`**`next_page_checker`**(*page: dict*) → Optional[str]

This is the parser for checking "nextParams" from react page data for automatic paging in get*Iter function.

Parameters **page** (`dict`) – Page data from get(Comments, Posts, etc..)

Returns `str`. "after" parameter for paging. This returns None if response doesn't have "nextParams"

3.6.4 response_json_stripper()

`vlivepy.parser.`**`response_json_stripper`**(*parsed_json_dict: dict, silent: bool = False*) → Optional[dict]

General parser for normalize response data format of json. This parses result and deletes response code. Also strip "data" field when dealing with membership response.

Parameters

- **parsed_json_dict** (`dict`) – Loaded json data to normalize.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.6.5 v_timestamp_parser()

`vlivepy.parser.`**`v_timestamp_parser`**(*ts: Union[str, int]*) → float

This is the function for parsing VLIVE timeunit(microsecond epoch) to float second.

Parameters **ts** (`Union[str, int]`) – VLIVE epoch time to parse.

Returns `float`. parsed epoch.

3.7 post

This page describes **post** module which can be imported as `vlivepy.post`

3.7.1 getFVideoInkeyData()

`vlivepy.post.`**`getFVideoInkeyData`**(*f_video_id: str, session: Optional[vlivepy.session.UserSession] = None, silent: bool = False*) → Optional[dict]

Get InKey data of File video

Parameters

- **f_video_id** (`str`) – Unique id of the FVideo to load InKey data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.7.2 getFVideoPlayInfo()

`vlivepy.post.`**`getFVideoPlayInfo`**(*f_video_id: str, f_vod_id: str, session: Optional[vlivepy.session.UserSession] = None, silent: bool = False*) → Optional[dict]

Get InKey data of File video

Parameters

- **f_video_id** (`str`) – Unique id of the video-attachment to load data.
- **f_vod_id** (`str`) – Unique id of the video-vod to load data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.8 schedule

This page describes **schedule** module which can be imported as `vlivepy.schdule`

3.8.1 getScheduleData()

`vlivepy.schedule.getScheduleData`(*schedule_id: str*, *session: vlivepy.session.UserSession*, *silent: bool = False*) → Optional[dict]

Get detailed schedule data.

Parameters

- **schedule_id** (`str`) – Unique id of the schedule to load data.
- **session** (*`vlivepy.UserSession`*) – Session for loading data with permission.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.9 session

This page describes **controllers** module which can be imported as `vlivepy.session`

3.9.1 getUserSession()

`vlivepy.session.getUserSession`(*email: str*, *pwd: str*, *silent: bool = False*) → Optional[requests.sessions.Session]

Get logged in `reqWrapper.Session` session

Parameters

- **email** (`str`) – Email of the account to sign-in.
- **pwd** (*`vlivepy.UserSession`*, optional) – Password of the account to sign-in.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `reqWrapper.Session`. Logged in session.

3.9.2 UserSession

This function has alias of *`vlivepy.UserSession`*

3.9.3 dumpSession()

This function has alias of *`vlivepy.dumpSession()`*

3.9.4 loadSession()

This function has alias of *`vlivepy.loadSession()`*

3.10 upcoming

This page describes **upcoming** module which can be imported as `vlivepy.upcoming`

3.10.1 UpcomingVideo

class `vlivepy.upcoming.`**`UpcomingVideo`**(*seq*, *time*, *cseq*, *cname*, *ctype*, *name*, *type*, *product*)

This is the object for upcoming list item

property cname

Origin channel name of item.

Return type `str`

property cseq

Origin channel seq id of item.

Return type `str`

property ctype

Origin channel type of item.

Returns "BASIC" if the channel type is normal. "PREMIUM" if the channel type is membership.

Return type `str`

property name

Title of item.

Return type `str`

property product

Product type of item.

Returns "NONE" if the item is normal live. "PAID" if the item is VLIVE+ product.

Return type `str`

property seq

VideoSeq of item.

Return type `str`

property time

String start time of item.

Return type `str`

property type

Type of item.

Returns "VOD", "UPCOMING_VOD", "UPCOMING_LIVE", "LIVE"

Return type `str`

3.10.2 getUpcomingList()

`vlivepy.upcoming.getUpcomingList`(*date: Optional[Union[str, int]] = None*, *silent: bool = False*) → Optional[List[*vlivepy.upcoming.UpcomingVideo*]]

Load upcoming webpage and parse each item.

Parameters

- **date** (`Union[str, int]`, optional) – The date with yyyyMMdd format to load upcoming, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns List of `UpcomingVideo`

3.11 video

This page describes **video** module which can be imported as `vlivepy.video`

3.11.1 getInkeyData()

`vlivepy.video.getInkeyData`(*video_seq: Union[str, int]*, *session: Optional[vlivepy.session.UserSession] = None*, *silent: bool = False*) → Optional[dict]

Get InKey data of current session and video.

Parameters

- **video_seq** (`str`) – Unique seq id of the video post to load data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.11.2 getLivePlayInfo()

`vlivepy.video.getLivePlayInfo`(*video_seq: Union[str, int]*, *session: Optional[vlivepy.session.UserSession] = None*, *vpdid2: Optional[str] = None*, *silent: bool = False*) → Optional[dict]

Get detailed play info of live.

Parameters

- **video_seq** (`str`) – Unique seq id of the video post to load data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **vpdid2** (`str`, optional) – User vpdid2 data, defaults to None. It can be automatically generated by `session`.
- **silent** (`bool`, optional) – Return None instead of raising exception, defaults to False.

Returns `dict`. Parsed json data

3.11.3 getLiveStatus()

`vlivepy.video.getLiveStatus(video_seq: Union[str, int], silent: bool = False) → Optional[dict]`

Get simplified status of live.

Parameters

- **video_seq** (str) – Unique seq id of the video post to load data.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data

3.11.4 getOfficialVideoData()

`vlivepy.video.getOfficialVideoData(video_seq: Union[str, int], session: Optional[vlivepy.session.UserSession] = None, silent: bool = False) → Optional[dict]`

Video utility for parsing VOD id from official video post data. This internally uses `getOfficialVideoPost()` function and parse VOD id from it.

Parameters

- **video_seq** (str) – Unique seq id of the video post to load data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data, defaults to None.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data

3.11.5 getOfficialVideoPost()

`vlivepy.video.getOfficialVideoPost(video_seq: Union[str, int], session: Optional[vlivepy.session.UserSession] = None, silent: bool = False) → Optional[dict]`

Get detailed official video post data.

Parameters

- **video_seq** (str) – Unique seq id of the video post to load data.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data

3.11.6 getVodId()

`vlivepy.video.getVodId(video_seq: Union[str, int], silent: bool = False)` → Optional[str]

Video utility for parsing VOD id from official video post data. This internally uses `getOfficialVideoPost()` function and parse VOD id from it.

Parameters

- **video_seq** (str) – Unique seq id of the video post to load data.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns str. Parsed VOD id

3.11.7 getVodPlayInfo()

`vlivepy.video.getVodPlayInfo(video_seq: Union[str, int], vod_id: Optional[str] = None, session: Optional[vlivepy.session.UserSession] = None, silent: bool = False)` → Optional[dict]

Get detailed play info of VOD.

Parameters

- **video_seq** (str) – Unique seq id of the video post to load data.
- **vod_id** (str, optional) – Unique id of the VOD to load data, defaults to None. It can be automatically generated with func:*getVodId*.
- **session** (`vlivepy.UserSession`, optional) – Session for loading data with permission, defaults to None.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns dict. Parsed json data

3.11.8 getVpdid2()

`vlivepy.video.getVpdid2(session: vlivepy.session.UserSession, silent: bool = False)` → Optional[str]

Video utility for get user's vpdid2 info. This internally uses `getInkeyData()` function with `video_seq="142851"` param and parse vpdid2 from it.

Parameters

- **session** (`vlivepy.UserSession`) – Session for loading data.
- **silent** (bool, optional) – Return None instead of raising exception, defaults to False.

Returns str. Parsed vpdid2.

PYTHON MODULE INDEX

V

vlivepy.exception, 5

INDEX

A

APIError, 5
APIJSONParesError, 5
APINetworkError, 5
APIServerResponseError, 5
APIServerResponseWarning, 5
APISignInFailedError, 5
attachments() (*vlivepy.model.PostModel property*), 11
attachments_photo() (*vlivepy.model.PostModel property*), 11
attachments_video() (*vlivepy.model.PostModel property*), 11
author() (*vlivepy.Comment property*), 15
author() (*vlivepy.model.PostModel property*), 11
author() (*vlivepy.Schedule property*), 20
author_id() (*vlivepy.model.PostModel property*), 11
author_id() (*vlivepy.Schedule property*), 21
author_memberId() (*vlivepy.Comment property*), 15
author_nickname() (*vlivepy.Comment property*), 15
author_nickname() (*vlivepy.model.PostModel property*), 11
author_nickname() (*vlivepy.Schedule property*), 21

B

background_color() (*vlivepy.Channel property*), 13
board_id() (*vlivepy.model.PostModel property*), 11
board_names() (*vlivepy.GroupedBoards method*), 16
BoardPostItem (*class in vlivepy.board*), 27
boards() (*vlivepy.GroupedBoards method*), 16
body() (*vlivepy.Comment property*), 15
body() (*vlivepy.Post property*), 17

C

Channel (*class in vlivepy*), 13
channel_code() (*vlivepy.Channel property*), 13
channel_code() (*vlivepy.model.PostModel property*), 11
channel_code() (*vlivepy.Schedule property*), 21
channel_cover_image() (*vlivepy.Channel property*), 13
channel_description() (*vlivepy.Channel property*), 13
channel_name() (*vlivepy.Channel property*), 13
channel_name() (*vlivepy.model.PostModel property*), 11
channel_name() (*vlivepy.Schedule property*), 21
channel_profile_image() (*vlivepy.Channel property*), 13
cname() (*vlivepy.upcoming.UpcomingVideo property*), 36
Comment (*class in vlivepy*), 15
comment_count() (*vlivepy.Comment property*), 15
comment_count() (*vlivepy.model.OfficialVideoModel property*), 8
comment_count() (*vlivepy.model.PostModel property*), 11
comment_count() (*vlivepy.Schedule property*), 21
comment_parser() (*in module vlivepy.comment*), 29
commentId() (*vlivepy.Comment property*), 15
content_type() (*vlivepy.model.PostModel property*), 11
created_at() (*vlivepy.Comment property*), 15
created_at() (*vlivepy.model.OfficialVideoModel property*), 8
created_at() (*vlivepy.model.PostModel property*), 12
cseq() (*vlivepy.upcoming.UpcomingVideo property*), 36
ctype() (*vlivepy.upcoming.UpcomingVideo property*), 36

D

DataModel (*class in vlivepy.model*), 7
decode_channel_code() (*in module vlivepy*), 26
decode_channel_code() (*vlivepy.Channel method*), 13
dimension_type() (*vlivepy.OfficialVideoVOD property*), 19
dumpSession() (*in module vlivepy*), 26

E

emotion_count() (*vlivepy.Comment property*), 15

emotion_count() (*vlivepy.model.PostModel property*), 12
emotion_count() (*vlivepy.Schedule property*), 21
encoding_status() (*vlivepy.OfficialVideoVOD property*), 19
expose_status() (*vlivepy.model.OfficialVideoModel property*), 8

F

format_epoch() (*in module vlivepy.parser*), 33
formatted_body() (*vlivepy.Post method*), 17

G

getBoardPosts() (*in module vlivepy.board*), 27
getBoardPostsIter() (*in module vlivepy.board*), 28
getChannelInfo() (*in module vlivepy.channel*), 28
getCommentData() (*in module vlivepy.comment*), 29
getFVideoInkeyData() (*in module vlivepy.post*), 34
getFVideoPlayInfo() (*in module vlivepy.post*), 34
getGroupedBoards() (*in module vlivepy.channel*), 29
getInkeyData() (*in module vlivepy.video*), 37
getInkeyData() (*vlivepy.OfficialVideoVOD method*), 19
getLivePlayInfo() (*in module vlivepy.video*), 37
getLivePlayInfo() (*vlivepy.OfficialVideoLive method*), 18
getLiveStatus() (*in module vlivepy.video*), 38
getLiveStatus() (*vlivepy.OfficialVideoLive method*), 18
getNestedComments() (*in module vlivepy.comment*), 30
getNestedCommentsIter() (*in module vlivepy.comment*), 30
getNestedCommentsIter() (*vlivepy.Comment method*), 15
getOfficialVideoData() (*in module vlivepy.video*), 38
getOfficialVideoPost() (*in module vlivepy.video*), 38
getPostComments() (*in module vlivepy.comment*), 30
getPostCommentsIter() (*in module vlivepy.comment*), 31
getPostCommentsIter() (*vlivepy.model.PostModel method*), 12
getPostInfo() (*in module vlivepy.connections*), 32
getPostStarComments() (*in module vlivepy.comment*), 31
getPostStarCommentsIter() (*in module vlivepy.comment*), 31
getPostStarCommentsIter() (*vlivepy.model.PostModel method*), 12
getScheduleData() (*in module vlivepy.schedule*), 35
getUpcomingList() (*in module vlivepy.upcoming*), 37
getUserSession() (*in module vlivepy.session*), 35
getVodId() (*in module vlivepy.video*), 39
getVodPlayInfo() (*in module vlivepy.video*), 39
getVodPlayInfo() (*vlivepy.OfficialVideoVOD method*), 20
getVpdid2() (*in module vlivepy.video*), 39
GroupedBoards (*class in vlivepy*), 16
groupedBoards() (*vlivepy.Channel method*), 13
groups() (*vlivepy.GroupedBoards method*), 16

H

has_filter_ad() (*vlivepy.OfficialVideoLive property*), 18
has_live_thumb() (*vlivepy.model.OfficialVideoModel property*), 9
has_mobile_da() (*vlivepy.model.OfficialVideoModel property*), 9
has_moment() (*vlivepy.OfficialVideoVOD property*), 20
has_notice() (*vlivepy.model.OfficialVideoModel property*), 9
has_official_video() (*vlivepy.board.BoardPostItem property*), 27
has_post_ad() (*vlivepy.model.OfficialVideoModel property*), 9
has_pre_ad() (*vlivepy.model.OfficialVideoModel property*), 9
has_preview() (*vlivepy.OfficialVideoVOD property*), 20
has_special_live() (*vlivepy.OfficialVideoLive property*), 18
has_upcoming() (*vlivepy.model.OfficialVideoModel property*), 9
hevc() (*vlivepy.OfficialVideoLive property*), 18

I

is_comment_enabled() (*vlivepy.model.PostModel property*), 12
is_hidden_from_star() (*vlivepy.model.PostModel property*), 12
is_viewer_bookmarked() (*vlivepy.model.PostModel property*), 12

L

like_count() (*vlivepy.model.OfficialVideoModel property*), 9
load() (*vlivepy.Upcoming method*), 22

loadSession() (*in module vlivepy*), 26
low_latency() (*vlivepy.OfficialVideoLive property*), 19

M

max_res_from_play_info() (*in module vlivepy.parser*), 33
member_count() (*vlivepy.Channel property*), 13
ModelError, 5
ModelInitError, 5
ModelRefreshWarning, 5
module
vlivepy.exception, 5
momentable() (*vlivepy.OfficialVideoLive property*), 19
multinational_title_get() (*vlivepy.model.OfficialVideoModel method*), 9
multinational_title_locales() (*vlivepy.model.OfficialVideoModel method*), 9
multinational_titles() (*vlivepy.model.OfficialVideoModel property*), 9

N

name() (*vlivepy.upcoming.UpcomingVideo property*), 36
next_page_checker() (*in module vlivepy.parser*), 33

O

official_video() (*vlivepy.OfficialVideoPost method*), 17
official_video() (*vlivepy.Schedule method*), 21
official_video_type() (*vlivepy.OfficialVideoPost property*), 18
official_video_type() (*vlivepy.Schedule property*), 21
OfficialVideoLive (*class in vlivepy*), 18
OfficialVideoModel (*class in vlivepy.model*), 8
OfficialVideoPost (*class in vlivepy*), 17
OfficialVideoVOD (*class in vlivepy*), 19
on_air_start_at() (*vlivepy.model.OfficialVideoModel property*), 9
open_at() (*vlivepy.Channel property*), 13

P

parent() (*vlivepy.Comment property*), 15
parent_info_tuple() (*vlivepy.Comment method*), 16
plain_body() (*vlivepy.Post property*), 17
play_count() (*vlivepy.model.OfficialVideoModel property*), 9
play_time() (*vlivepy.OfficialVideoVOD property*), 20
Post (*class in vlivepy*), 17
post_count() (*vlivepy.Channel property*), 14
post_id() (*vlivepy.board.BoardPostItem property*), 27
post_id() (*vlivepy.model.PostModel property*), 12
post_id() (*vlivepy.Schedule property*), 21
postIdToVideoSeq() (*in module vlivepy*), 25
PostModel (*class in vlivepy.model*), 10
postTypeDetector() (*in module vlivepy*), 26
pp_type() (*vlivepy.OfficialVideoLive property*), 19
product() (*vlivepy.upcoming.UpcomingVideo property*), 36
product_type() (*vlivepy.model.OfficialVideoModel property*), 9
prohibited_word_exact_list() (*vlivepy.Channel property*), 14
prohibited_word_like_list() (*vlivepy.Channel property*), 14

Q

qr_code() (*vlivepy.Channel property*), 14

R

raw() (*vlivepy.model.DataModel property*), 8
recommended_videos() (*vlivepy.OfficialVideoVOD method*), 20
refresh() (*vlivepy.model.DataModel method*), 8
refresh() (*vlivepy.Upcoming method*), 23
refresh() (*vlivepy.UserSession method*), 24
refresh_rate (*vlivepy.Upcoming attribute*), 22
representative_color() (*vlivepy.Channel property*), 14
response_json_stripper() (*in module vlivepy.parser*), 33
root() (*vlivepy.Comment property*), 16
root_info_tuple() (*vlivepy.Comment method*), 16

S

Schedule (*class in vlivepy*), 20
schedule_id() (*vlivepy.Schedule property*), 21
screen_orientation() (*vlivepy.model.OfficialVideoModel property*), 10
seq() (*vlivepy.upcoming.UpcomingVideo property*), 36
session (*vlivepy.board.BoardPostItem attribute*), 27
session (*vlivepy.Channel attribute*), 13
session (*vlivepy.Comment attribute*), 15
session (*vlivepy.GroupedBoards attribute*), 16
session (*vlivepy.model.DataModel attribute*), 8
session (*vlivepy.model.OfficialVideoModel attribute*), 8
session (*vlivepy.model.PostModel attribute*), 11
session (*vlivepy.OfficialVideoLive attribute*), 18
session (*vlivepy.OfficialVideoPost attribute*), 17
session (*vlivepy.OfficialVideoVOD attribute*), 19
session (*vlivepy.Post attribute*), 17
session (*vlivepy.Schedule attribute*), 20

session() (*vlivepy.UserSession property*), 24
show_live (*vlivepy.Upcoming attribute*), 22
show_upcoming() (*vlivepy.Channel property*), 14
show_upcoming_live (*vlivepy.Upcoming attribute*), 22
show_upcoming_vod (*vlivepy.Upcoming attribute*), 22
show_vod (*vlivepy.Upcoming attribute*), 22
sns_share_img() (*vlivepy.Channel property*), 14
status() (*vlivepy.OfficialVideoLive property*), 19
sticker() (*vlivepy.Comment property*), 16

T

target_id() (*vlivepy.model.DataModel property*), 8
thumb() (*vlivepy.model.OfficialVideoModel property*), 10
time() (*vlivepy.upcoming.UpcomingVideo property*), 36
title() (*vlivepy.model.OfficialVideoModel property*), 10
title() (*vlivepy.model.PostModel property*), 12
title() (*vlivepy.Schedule property*), 21
to_object() (*vlivepy.board.BoardPostItem method*), 27
type() (*vlivepy.upcoming.UpcomingVideo property*), 36

U

Upcoming (*class in vlivepy*), 22
upcoming() (*vlivepy.Upcoming method*), 23
UpcomingVideo (*class in vlivepy.upcoming*), 36
use_member_level() (*vlivepy.Channel property*), 14
UserSession (*class in vlivepy*), 24

V

v_timestamp_parser() (*in module vlivepy.parser*), 33
video_comment_count() (*vlivepy.Channel property*), 14
video_count() (*vlivepy.Channel property*), 14
video_like_count() (*vlivepy.Channel property*), 14
video_play_count() (*vlivepy.Channel property*), 14
video_seq() (*vlivepy.model.OfficialVideoModel property*), 10
video_seq() (*vlivepy.OfficialVideoPost property*), 18
video_seq() (*vlivepy.Schedule property*), 21
video_type() (*vlivepy.model.OfficialVideoModel property*), 10
videoSeqToPostId() (*in module vlivepy*), 25
vlivepy.exception
module, 5
vod_id() (*vlivepy.OfficialVideoVOD property*), 20
vod_secure_status() (*vlivepy.OfficialVideoVOD property*), 20
vr_content_type() (*vlivepy.model.OfficialVideoModel property*), 10

W

will_end_at() (*vlivepy.model.OfficialVideoModel property*), 10
will_start_at() (*vlivepy.model.OfficialVideoModel property*), 10
written_in() (*vlivepy.Comment property*), 16
written_in() (*vlivepy.Post property*), 17