braviaproapi

Jun 19, 2022

Contents

This library provides an easy-to-use Python interface for controlling Sony Bravia televisions. It implements the BRAVIA Professional Display API, which is present on recent consumer hardware. For more information, take a look at Sony's API documentation.

It supports the following features:

- Control and launch applications, including text entry into form fields.
- Configuration of display and audio options
- Control over various system functions (sleep/wake, LED configuration, power saving, etc.)
- Direct control of external inputs and media sources
- Emulated remote control input via IRCC commands

Take a look at the Getting Started page to learn how to use the library.

CHAPTER 1

Compatibility

This library is intended for use on newer, Android-based televisions. A list of devices and software versions known to be compatible is available on the GitHub wiki.

It has come to my attention that some newer Bravia models have received software updates bumping their API version to higher than 3.x. These devices are not supported by braviaproapi at this time as I do not have a compatible device to test with. Contributions to the library (and the above linked wiki page) are encouraged if you have a supported device!

CHAPTER 2

Contributing

See something that could be improved? Pull requests and issues are accepted at the project's GitHub repository.

CHAPTER 3

Table of Contents

3.1 Getting Started

3.1.1 Configuring Your Television

Danger: You should **NEVER** expose your television's API to the Internet directly as this poses a significant security risk to your network and television. If you insist on controlling your TV from outside your home network you should set up separate, more secure, software that only exposes the functionality you need.

These instructions are based on a 2015 Bravia TV running Android 7.0 (Nougat). The steps may differ on newer devices.

To make your television's API accessible to this library:

1. Open Settings
2. Select Network > Home network > IP control
3. Set Authentication to Normal and Pre-Shared Key
4. Select Pre-Shared Key and specify your passcode of choice.
5. Make note of your television's IP address. You may want to make it static to avoid connection loss.

That's it! Now you can begin controlling your television.

3.1.2 Installing The Library

This library is published to PyPI and can be easily installed by running the below command (preferably in a virtualenv). Python 3.7 or higher is required to use this library.

```
pip install braviaproapi
```

3.1.3 Sending Commands

Tip: Full documentation of available commands and the BraviaClient is available on the braviaproapi page.

Now that you have the library available, let's set up the client, change the volume, change inputs, and open an app.

```
from braviaproapi import BraviaClient

television = BraviaClient(host="192.168.1.200", passcode="0000")

# Wake up the TV if it's asleep
is_powered_on = television.system.get_power_status()
if not is_powered_on:
    television.system.power_on()

# Change input to HDMI 2
television.avcontent.set_play_content("extInput:hdmi?port=2")

# Set the volume to 20%
television.audio.set_volume_level(20)

# Play roulette? Open the first app the TV returns.
apps = television.appcontrol.get_application_list(exclude_builtin=True)
television.appcontrol.set_active_app(apps[0].get("uri"))
```

Feel like going retro? You can send raw remote control commands as well. A list of remote codes is available at braviaproapi.bravia.remote.

```
from braviaproapi import BraviaClient
from braviaproapi.bravia import ButtonCode

television = BraviaClient(host="192.168.1.200", passcode="0000")

television.remote.send_button(ButtonCode.POWER)
television.remote.send_button(ButtonCode.HDMI_1)
```

3.1.4 Handling Errors

Since Sony's API documentation is a little sketchy, you should be prepared to handle errors raised by the library. See braviaproapi.bravia.errors for a list of possible errors that may arise due to user error, problems with the library, or device-specific issues. The function documentation also indicates which errors may be raised by each function.

3.2 braviaproapi package

3.2.1 braviaproapi.BraviaClient

class braviaproapi.**BraviaClient**(*host*, *passcode*)

Bases: object

Provides the client for interacting with the Bravia API.

appcontrol

Provides app control and information.

Type *AppControl*

audio
Provides audio control and information.

Type *Audio*

avcontent
Provides control for content displayed by the device.

Type *AvContent*

encryption
Provides access to device encryption.

Type *Encryption*

http_client
HTTP client for direct API communication with the device.

Type *Http*

remote
Provides remote control input and information relating to it.

Type *Remote*

system
Provides system information and configuration functionality.

Type *System*

videoscreen
Provides control of the device's display.

Type *VideoScreen*

Parameters

- **host** (*str*) – The IP address or domain name belonging to the target device.
- **passcode** (*str*) – The pre-shared key configured on the target device.

3.2.2 braviaproapi.bravia package

braviaproapi.bravia.AppControl

class braviaproapi.bravia.**AppControl**(*bravia_client*, *http_client*)
Bases: object

Provides functionality for interacting with applications on the target device.

Parameters

- **bravia_client** – The parent BraviaClient instance.
- **http_client** – The *Http* instance associated with the parent client.

get_application_feature_status()
Determines which features are supported by the currently running application on the target device.

Raises ApiError – The request to the target device failed.

Returns

A dict with the following keys with boolean values:

- textInput (*bool*): True if the application currently has a text input focused.
- cursorDisplay (*bool*): True if the application currently has an interactive cursor.
- webBrowse (*bool*): True if the application currently has a web browser displayed.

Return type dict

get_application_list(*exclude_builtin=False*)

Retrieves a list of applications installed on the target device.

Parameters **exclude_builtin** (*bool*) – If True, excludes built-in Sony applications which are not exposed on the home screen.

Raises

- TypeError – One or more arguments is the incorrect type.
- ApiError – The request to the target device failed.

Returns

A list of dicts containing the following properties:

- name (*str or None*): The display name of the application.
- uri (*str or None*): The internal URI at which the application can be accessed, used when referring to the app from other functions.
- icon (*str or None*): A network URL pointing to the application's icon image.

Return type list(dict)

get_text_form()

Decrypts and returns the contents of the text field focused on the target device.

Raises

- InternalError – The target device was unable to encrypt the text.
- ApiError – The request to the target device failed.
- EncryptionError – The target device could not provide a valid encryption key.

Returns The text, or *None* if no text field is currently focused.

Return type str or None

get_web_app_status()

Returns information about the web application currently in use on the target device.

Raises ApiError – The request to the target device failed.

Returns

A dict containing the following keys:

- active (*bool*): True if there is currently a web application running on the target device.
- url (*str or None*): The URL of the application currently running, None if no such app is running.

Return type dict

set_active_app(*uri*)
Opens the specified app on the target device.

Parameters **uri** (*str*) – The URI of the application to open (acquired using *get_application_list()*)

Raises

- TypeError – One or more arguments is the incorrect type.
- AppLaunchError – The application could not be opened.
- ApiError – The request to the target device failed.

set_text_form(*text*)
Enters the specified text in the focused text field on the target device. Text is encrypted before being sent to the device.

Parameters **text** (*str*) – The text to input.

Raises

- TypeError – One or more arguments is the incorrect type.
- ApiError – The request to the device failed.
- EncryptionError – The target device could not provide a valid encryption key.
- NoFocusedTextFieldError – There is no text field to input text to on the target device.
- InternalError – The target device failed to decrypt the text.

terminate_all_apps()
Instructs the target device to terminate all running applications.

Raises ApiError – The request to the target device failed.

class braviaproapi.bravia.**AppFeature**
Bases: enum.Enum

Describes which features are supported by the current app.

UNKNOWN
The app feature was not recognized.

TEXT_INPUT
The app has a text field focused.

CURSOR_DISPLAY
The app has a cursor displayed.

WEB_BROWSE
The app is using an embedded web browser.

braviaproapi.bravia.Audio

class braviaproapi.bravia.**Audio**(*bravia_client*, *http_client*)
Bases: object

Provides functionality for controlling audio on the target device.

Parameters

- **bravia_client** – The parent BraviaClient instance.

- **http_client** – The *Http* instance associated with the parent client.

decrease_volume(*decrease_by=1*, *show_ui=True*, *device=None*)
Decreases volume level of the specified audio output device on the target device.

Parameters

- **decrease_by** (*int, optional*) – Defaults to 1. How many units to decrease the volume on the target device.
- **show_ui** (*bool, optional*) – Defaults to True. Whether to display the volume UI on the target device when changing volume.
- **device** (*VolumeDevice, optional*) – Defaults to *None*. Specifies which audio device to change the volume of. If not specified, affects all audio devices. May not be *VolumeDevice.UNKNOWN*.

Raises

- `TypeError` – One or more arguments is the incorrect type.
- `ValueError` – One or more arguments is invalid.
- `ApiError` – The request to the target device failed.
- `VolumeOutOfRangeError` – The specified volume is out of range for the target device.
- `TargetNotSupportedError` – The specified audio device is not supported by the target device.
- `InternalError` – An internal error occurred.

get_output_device()
Returns the current audio output device on the target device.

Raises `ApiError` – The request to the target device failed.

Returns The current output device.

Return type *AudioOutput*

get_speaker_settings()
Returns the current audio settings for the target device.

Raises `ApiError` – The request to the target device failed.

Returns

A dict with the following *SpeakerSetting* keys. Each key's value may be *None* if the target device does not provide that setting.

- *SpeakerSetting.TV_POSITION* (*TvPosition*): The physical location of the device.
- *SpeakerSetting.SUBWOOFER_LEVEL* (*int*): The configured volume of the subwoofer.
- *SpeakerSetting.SUBWOOFER_PHASE* (*SubwooferPhase*): The phase setting of the subwoofer.
- *SpeakerSetting.SUBWOOFER_FREQUENCY* (*int*): The confiugred frequency at which the subwoofer activates.
- *SpeakerSetting.SUBWOOFER_POWER* (*bool*): whether the subwoofer is powered on or not.

Return type dict

get_volume_information()
Returns the current volume information of each audio output device on the target device.

Raises ApiError – The request to the target device failed.

Returns

A list of dicts containing the following properties:

- min_volume (*int*): The minimum volume setting for the audio device.
- max_volume (*int*): The maximum volume setting for the audio device.
- muted (*bool*): whether the audio device is muted.
- type (*VolumeDevice*): The audio device represented by this entry.
- volume (*int*): The current volume of the audio device.

Return type list(dict)

increase_volume(*increase_by=1*, *show_ui=True*, *device=None*)
Increases volume level of the specified audio output device on the target device.

Parameters

- **increase_by** (*int, optional*) – Defaults to 1. How many units to increase the volume on the target device.
- **show_ui** (*bool, optional*) – Defaults to True. Whether to display the volume UI on the target device when changing volume.
- **device** (VolumeDevice, *optional*) – Defaults to *None*. Specifies which audio device to change the volume of. If not specified, affects all audio devices. May not be *VolumeDevice.UNKNOWN*.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.
- VolumeOutOfRangeError – The specified volume is out of range for the target device.
- TargetNotSupportedError – The specified audio device is not supported by the target device.
- InternalError – An internal error occurred.

mute()
Mutes the current audio output device on the target device.

Raises ApiError – The request to the target device failed.

set_mute(*mute*)
Mutes or unmutes the current audio output device on the target device.

Parameters **mute** (*bool*) – If True, mutes the device. Otherwise, unmutes the device.

Raises

- TypeError – One or more arguments is the incorrect type.
- ApiError – The request to the target device failed.

set_output_device(*output_device*)
Sets which audio output device the target device should use.

Parameters **output_device** (AudioOutput) – The output device to use. May not be *AudioOutput.UNKNOWN*.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.
- InternalError – An internal error occurred.

set_speaker_settings(*settings*)
Configures the settings relating to speakers on the target device.

Parameters **settings** (*dict*) – Must contain one or more of the following *SpeakerSetting* keys.

- *SpeakerSetting.TV_POSITION* (*TvPosition*): The physical location of the device. May not be *TvPosition.UNKNOWN*.
- *SpeakerSetting.SUBWOOFER_LEVEL* (*int*): The configured volume of the subwoofer. Generally a value between 0 and 24, but may vary by device.
- *SpeakerSetting.SUBWOOFER_PHASE* (*SubwooferPhase*): The phase setting of the subwoofer. May not be *SubwooferPhase.UNKNOWN*.
- *SpeakerSetting.SUBWOOFER_FREQUENCY* (*int*): The confiugred frequency at which the subwoofer activates. Generally a value between 0 and 30, but may vary by device.
- *SpeakerSetting.SUBWOOFER_POWER* (*bool*): whether the subwoofer is powered on or not.

Raises

- TypeError – One or more members of the dict is the incorrect type.
- ValueError – One or more members of the dict is invalid.
- ApiError – The request to the target device failed.
- InternalError – An internal error occurred.

set_volume_level(*volume*, *show_ui=True*, *device=None*)
Sets the volume level of the specified audio output device on the target device.

Parameters

- **volume** (*int*) – The volume to set on the target device. Generally this is on a scale from 0 to 100, but this may vary by device.
- **show_ui** (*bool, optional*) – Defaults to True. Whether to display the volume UI on the target device when changing volume.
- **device** (VolumeDevice, *optional*) – Defaults to *None*. Specifies which audio device to change the volume of. If not specified, affects all audio devices. May not be *VolumeDevice.UNKNOWN*.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.

- `ApiError` – The request to the target device failed.
- `VolumeOutOfRangeError` – The specified volume is out of range for the target device.
- `TargetNotSupportedError` – The specified audio device is not supported by the target device.
- `InternalError` – An internal error occurred.

unmute()
Unmutes the current audio output device on the target device.

Raises `ApiError` – The request to the target device failed.

class braviaproapi.bravia.**AudioOutput**
Bases: `enum.Enum`

Describes the audio output device used by the target device.

UNKNOWN
The audio output was not recognized.

SPEAKER
An external speaker.

SPEAKER_HDMI
An external HDMI-connected speaker.

HDMI
HDMI audio output.

AUDIO_SYSTEM
Internal speakers.

class braviaproapi.bravia.**TvPosition**
Bases: `enum.Enum`

Describes the mounting position of the device.

UNKNOWN
The TV position was not recognized.

TABLE_TOP
The TV is standing on a table.

WALL_MOUNT
The TV is mounted on a wall.

class braviaproapi.bravia.**SubwooferPhase**
Bases: `enum.Enum`

Describes the phase polarity setting of the wireless subwoofer.

UNKNOWN
The subwoofer phase was not recognized.

NORMAL
The subwoofer is using normal polarity.

REVERSE
The subwoofer is using reverse polarity.

class braviaproapi.bravia.**VolumeDevice**
Bases: `enum.Enum`

Describes the output device that the volume level is applied to.

UNKNOWN
The volume device was not recognized.

SPEAKERS
The speaker output.

HEADPHONES
The headphone output.

class braviaproapi.bravia.**SpeakerSetting**
Bases: enum.Enum

Describes available settings relating to audio.

UNKNOWN
The SpeakerSetting was not recognized.

TV_POSITION
The mounting position of the device.

SUBWOOFER_LEVEL
The volume level of the wireless subwoofer.

SUBWOOFER_FREQUENCY
The frequency setting of the wireless subwoofer.

SUBWOOFER_PHASE
The phase polarity of the wireless subwoofer.

SUBWOOFER_POWER
Whether the wireless subwoofer is powered on or not.

braviaproapi.bravia.AvContent

class braviaproapi.bravia.**AvContent**(*bravia_client*, *http_client*)
Bases: object

Provides functionality for controlling what is played on the target device.

Parameters

- **bravia_client** – The parent BraviaClient instance.
- **http_client** – The *Http* instance associated with the parent client.

get_content_count(*source*)
Returns a count of the number of available contents for a given source.

Parameters **source** (–) – The URI of the source to enumerate. See the Sony documentation for more information.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.

Returns The count of available content.

Return type int

get_content_list(*source*)
Returns a list of available content for a given source.

Parameters **source** (–) – The URI of the source to enumerate. See the Sony documentation for more information.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.

Returns

A list of dicts containing the following keys. If no content is available, returns *None*.

- channel_info (*dict* or None): If the content is a television channel, this provides the following information. If the content is not a television channel, this is *None*.
  - channel_full (*str*): The full channel number (e.g. 7.2)
  - channel_main (*str*): The primary channel number (e.g. 7)
  - channel_sub (*str* or None): The subchannel number, if available (e.g. 2)
  - visible (*bool*): Whether the channel is enabled ("visible") on the television.
- index (*str*): The position of the content in the list.
- name (*str or None*): The title of the content, if applicable.
- uri (*str or None*): The URI at which the content can be accessed, if applicable.

Return type list(dict) or None

get_external_input_status()
Returns information about the target device's external inputs.

Raises ApiError – The request to the target device failed.

Returns

A list of dicts with the following keys:

- uri (*str or None*): The URI at which the input can be accessed, if applicable.
- name (*str or None*): The system title of the input, if applicable.
- connected (*bool*): True if the input is currently connected, False otherwise.
- custom_label (*str or None*): The user-entered title of the input, if set.
- icon (*InputIcon*): The icon for the input. If no appropriate icon is available, this is *InputIcon.UNKNOWN*.
- has_signal (*bool*): True if input is currently sending a signal to the target device, False otherwise.

Return type list(dict)

get_playing_content_info()
Returns information about the currently playing content on the target device.

Raises ApiError – The request to the target device failed.

Returns

A dict containing the following keys. If no content is playing, returns *None*.

- uri (*str or None*): The URI at which the content can be accessed, if applicable.
- source (*str or None*): The source that the content resides within, if applicable.
- name (*str or None*): The title of the playing content, if applicable.

Return type dict or None

get_scheme_list()
Returns a list of available content schemes the target device supports.

Raises ApiError – The request to the target device failed.

Returns A list of string names of available schemes.

Return type list(str)

get_source_list(*scheme*)
Returns a list of available source types for a given content scheme.

Parameters **scheme** (*str*) – The scheme for which to get sources (retrieve this from *get_scheme_list()*).

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.

Returns A list of string source URIs for the specified scheme. If scheme is not supported, returns *None*.

Return type list(str) or None

set_play_content(*uri*)
Activates the specified content on the target device.

Parameters **uri** (*str*) – The URI at which the content can be accessed. Find the URI from the results of the *get_content_list()* function.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.

class braviaproapi.bravia.**InputIcon**
Bases: enum.Enum

Describes icon types for input sources.

UNKNOWN
The icon type was not recognized.

COMPOSITE
Composite input

SVIDEO
S-Video input

COMPOSITE_COMPONENTD
Japanese D-terminal composite/component input

COMPONENTD
Japanese D-terminal component input

COMPONENT
Component (YPbPr) input

SCART
SCART RGB input

HDMI
HDMI input

VGA
VGA D-sub input

TUNER
Coaxial TV tuner input

TAPE
Tape input

DISC
Disc input

COMPLEX
Complex input

AV_AMP
Audio amplifier input

HOME_THEATER
Home theater system input

GAME
Video game input

CAMCORDER
Camcorder input

DIGITAL_CAMERA
Digital camera input

PC
Computer input

TV
Television input

AUDIO_SYSTEM
Audio system input

RECORDING_DEVICE
Recorder device input

PLAYBACK_DEVICE
Player device input

TUNER_DEVICE
Television tuner device input

WIFI_DISPLAY
Wi-Fi display input

braviaproapi.bravia.Encryption

class braviaproapi.bravia.**Encryption**(*bravia_client*, *http_client*)
Bases: object

Provides functionality for encrypted communication with the target device.

Parameters

- **bravia_client** – The parent BraviaClient instance.
- **http_client** – The *Http* instance associated with the parent client.

aes_decrypt_b64(*message*)
Decrypts AES messages sent from the target device.

Parameters **message** (*str*) – The message to decrypt.

Raises TypeError – One or more arguments is the incorrect type.

Returns The decrypted string.

Return type str

aes_encrypt_b64(*message*)
Encrypts AES messages to be sent to the target device.

Parameters **message** (*str*) – The message to encrypt.

Raises TypeError – One or more arguments is the incorrect type.

Returns The encrypted string.

Return type str

get_public_key()
Gets the target device's public encryption key.

Raises ApiError – The request to the target device failed.

Returns The device's public key, base64-encoded. If the device does not have a public key, returns *None*.

Return type str or None

get_rsa_encrypted_common_key()
Returns a common key to be used in encrypted communication with the target device.

This common key is generated when the Bravia client is initialized and used throghout the life of the application.

Returns

An AES common key, encrypted with RSA, to be sent to the target device. If no encryption capability is available on the target device, returns *None*.

Return type str

braviaproapi.bravia.Http

class braviaproapi.bravia.**Http**(*host*, *psk*)
Bases: object

Handles HTTP messaging to the API server.

Parameters

- **host** (*str*) – The HTTP hostname at which the server is running.
- **psk** (*str*) – The pre-shared key configured on the server.

remote_request(*remote_code*)
Sends an HTTP request to the Sony API to execute an IRCC remote code.

Parameters **remote_code** (*str*) – The base64-encoded IRCC code to send (see *ButtonCode* for examples).

Raises HttpError – The HTTP call failed.

request(*endpoint*, *method*, *params=None*, *version='1.0'*)
Sends a JSON-RPC request to the API server.

Parameters

- **endpoint** (*str*) – The API endpoint to send to.
- **method** (*str*) – The RPC method to execute.
- **params** (*dict, optional*) – Defaults to *None*. Parameters to send on the request.
- **version** (*str, optional*) – Defaults to "1.0". The version of the API endpoint to request.

Raises HttpError – The HTTP call failed. Refer to the *error_code* attribute for details.

Returns The Sony API returns a list of results. If only one result is returned (the majority of the time), this method extracts it and returns it alone. If more than one result is returned, this method returns the full list. If no results were found, this method returns *None*.

Return type list or None

braviaproapi.bravia.Remote

class braviaproapi.bravia.**Remote**(*bravia_client*, *http_client*)
Bases: object

Provides remote control functionality for the target device.

Parameters

- **bravia_client** – The parent BraviaClient instance.
- **http_client** – The *Http* instance associated with the parent client.

send_button(*button*)
Sends a remote control button press to the target device. Button codes can come from the predefined ButtonCode enum, or *System.get_remote_control_info()* can return a device-specific list.

Parameters **button** (*ButtonCode or str*) – The button code to send.

Raises

- TypeError – One or more arguments is the incorrect type.

- `ApiError` – The request to the target device failed.

class braviaproapi.bravia.**ButtonCode**

Bases: `enum.Enum`

Describes the default button codes for the IRCC remote control interface.

POWER
Power on/off

INPUT
Change input source

SYNC_MENU
Open the Bravia Sync menu

HDMI_1
Switch to HDMI 1 source

HDMI_2
Switch to HDMI 2 source

HDMI_3
Switch to HDMI 3 source

HDMI_4
Switch to HDMI 4 source

NUM_1
‘1’ key

NUM_2
‘2’ key

NUM_3
‘3’ key

NUM_4
‘4’ key

NUM_5
‘5’ key

NUM_6
‘6’ key

NUM_7
‘7’ key

NUM_8
‘8’ key

NUM_9
‘9’ key

NUM_0
‘0’ key

DOT
‘.’ or ‘-’ key used for tuner subchannels

CAPTION
Set closed captioning mode

RED
Red favorite key

GREEN
Green favorite key

YELLOW
Yellow favorite key

BLUE
Blue favorite key

UP
Up directional key

DOWN
Down directional key

RIGHT
Right directional key

LEFT
Left directional key

CONFIRM
Confirm/OK key

HELP
Opens system help

DISPLAY
Opens display options

OPTIONS
Opens options menu (Action Menu)

BACK
Returns to previous screen

HOME
Goes to home screen

VOLUME_UP
Increase volume by one unit

VOLUME_DOWN
Decrease volume by one unit

MUTE
Mute audio

AUDIO
Switch audio mode

CHANNEL_UP
Go to next TV channel

CHANNEL_DOWN
Go to previous TV channel

PLAY
Play content

STOP
Stop content

FLASH_PLUS
Fast forward

FLASH_MINUS
Rewind

PREV
Go to previous track

NEXT
Go to next track

braviaproapi.bravia.System

class braviaproapi.bravia.**System**(*bravia_client*, *http_client*)
Bases: object

Provides functionality for configuring the target device.

Parameters

- **bravia_client** – The parent BraviaClient instance.
- **http_client** – The *Http* instance associated with the parent client.

get_current_time()
Gets the current system time, if set.

Raises ApiError – The request to the target device failed.

Returns The current system time. If the time is not set, returns *None*.

Return type DateTime or None

get_interface_information()
Returns information about the server on the target device. This is used internally to check the current API version.

Raises ApiError – The request to the target device failed.

Returns

A dict containing the following keys:

- product_category (*str or None*): The device's category name.
- model_name (*str or None*): The model of the device.
- product_name (*str or None*): The product name of the device;
- server_name (*str or None*): The name of the server, if the device supports multiple.
- interface_version (*str or None*): The semver API version.

Return type dict

get_led_status()
Returns the current mode of the device's LED and whether it is enabled.

Raises ApiError – The request to the target device failed.

Returns

A dict containing the following keys, or *None* if the LED mode cannot be determined.

- status (*bool or None*): Whether the LED is enabled or not.
- mode (*LedMode*): Which LED mode the target device is currently using.

Return type dict or None

get_network_settings(*interface=None*)
Returns informaton about the target device's network configuration.

Parameters **interface** (*str, optional*) – Defaults to *None* (all interfaces). The interface to get information about.

Raises ApiError – The request to the target device failed.

Returns

A list of dicts containing the following keys. If an interface is specified and not found, returns *None*.

- name (*str or None*): The name of the interface.
- mac (*str or None*): The MAC address of the interface.
- ip_v4 (*str or None*): The IPv4 address of the interface, if available.
- ip_v6 (*str or None*): The IPv6 address of the interface, if available.
- netmask (*str or None*): The network mask for the interface.
- gateway (*str or None*): The configured gateway address for the interface.
- dns_servers (*list(str)*): A list of DNS servers configured on the interface.

Return type list(dict) or None

get_power_saving_mode()
Returns the current power saving mode of the device.

Raises ApiError – The request to the target device failed.

Returns The current power saving mode.

Return type *PowerSavingMode*

get_power_status()
Returns the current power state of the target device:

Raises ApiError – The request to the target device failed.

Returns True if device is awake, False if the device is in standby.

Return type bool

get_remote_access_status()
Returns whether remote access is enabled on the target device.

Raises ApiError – The request to the target device failed.

Returns True if remote access is enabled, False otherwise.

Return type bool

get_remote_control_info()
Returns a list of IRCC remote codes supported by the target device.

Raises `ApiError` – The request to the target device failed.

Returns A mapping of remote control button name (*str*) to IRCC code (*str*).

Return type dict

get_system_information()
Returns information about the target device.

Raises `ApiError` – The request to the target device failed.

Returns

A dict containing the following keys:

- product (*str or None*): The product name.
- language (*str or None*): The configured UI language.
- model (*str or None*): The device model.
- serial (*str or None*): The serial number of the device.
- mac (*str or None*): The device's MAC address.
- name (*str or None*): The name of the device.
- generation (*str or None*): The semver representation of the device's generation.

Return type dict

get_wake_on_lan_mac()
Returns the Wake-on-LAN (WOL) MAC address for the target device, if available.

Raises `ApiError` – The request to the target device failed.

Returns String MAC address of the device (format *00:00:00:00:00:00*), or None if Wake-on-LAN is not available.

Return type str or None

get_wake_on_lan_status()
Returns whether the Wake-on-LAN (WOL) function of the target device is enabled.

Raises `ApiError` – The request to the target device failed.

Returns True if Wake-on-LAN is enabled, False if not.

Return type bool

power_off()
Puts the target device into standby.

Raises `ApiError` – The request to the target device failed.

power_on()
Wakes up the target device.

Raises `ApiError` – The request to the target device failed.

request_reboot()
Reboots the target device.

Raises `ApiError` – The request to the target device failed.

set_language(*language*)
Sets the UI language of the target device. Language availabilit depends on the device's region settings.

Parameters **language** (*str*) – The ISO-639-3 code for the desired language.

Raises

- `TypeError` – One or more arguments is the incorrect type.
- `ApiError` – The request to the target device failed.
- `LanguageNotSupportedError` – The specified language is not supported by the device.

set_led_status(*mode*)
Sets the LED mode of the target device.

Parameters **mode** (`LedMode`) – The LED mode to set. May not be *LedMode.UNKNOWN*.

Raises

- `TypeError` – One or more arguments is the incorrect type.
- `ValueError` – One or more arguments is invalid.
- `ApiError` – The request to the target device failed.
- `InternalError` – An internal error occurred.

set_power_saving_mode(*mode*)
Sets the specified power saving mode on the target device.

Parameters **mode** (`PowerSavingMode`) – The power saving mode to set. May not be *PowerSavingMode.UNKNOWN*.

Raises

- `TypeError` – One or more arguments is the incorrect type.
- `ValueError` – One or more arguments is invalid.
- `ApiError` – The request to the target device failed.
- `InternalError` – An internal error occurred.

set_power_status(*power_state*)
Wakes or sleeps the target device.

Parameters **power_state** (*bool*) – True to wake, False to sleep.

Raises

- `TypeError` – One or more arguments is the incorrect type.
- `ApiError` – The request to the target device failed.

set_wake_on_lan_status(*enabled*)
Enables or disables Wake-on-LAN (WOL) on the target device.

Parameters **enabled** (*bool*) – Whether to enable Wake-on-LAN.

Raises

- `TypeError` – One or more arguments is the incorrect type.
- `ApiError` – The request to the target device failed.

class braviaproapi.bravia.**LedMode**
Bases: `enum.Enum`

Describes the mode of the LED indicator on the device.

UNKNOWN
The LED mode was not recognized.

DEMO
The LED is in demo mode.

AUTO_BRIGHTNESS
The LED adjusts its brightness based on the ambient light.

DARK
The LED is dimmed.

SIMPLE_RESPONSE
The LED lights only when responding to a command.

OFF
The LED is disabled.

class braviaproapi.bravia.**PowerSavingMode**
Bases: enum.Enum

Describes the device's power saving mode.

UNKNOWN
The power saving mode was not recognized.

OFF
Power saving is disabled.

LOW
Power saving mode is set to low.

HIGH
Power saving mode is set to high.

PICTURE_OFF
The display is disabled.

braviaproapi.bravia.VideoScreen

class braviaproapi.bravia.**VideoScreen**(*bravia_client*, *http_client*)
Bases: object

Provides functionality for configuring the target device's display.

Parameters

- **bravia_client** – The parent BraviaClient instance.
- **http_client** – The *Http* instance associated with the parent client.

set_scene_setting(*setting*)
Sets the scene mode for the display.

Parameters **setting** (SceneMode) – The scene mode to set. May not be *SceneMode.UNKNOWN*.

Raises

- TypeError – One or more arguments is the incorrect type.
- ValueError – One or more arguments is invalid.
- ApiError – The request to the target device failed.
- InvalidStateError – The target device is off or does not support this mode for the current input.

- `InternalError` – An internal error occurred.

class braviaproapi.bravia.**SceneMode**
Bases: `enum.Enum`

Specifies the screen mode of the target device.

UNKNOWN
The screen mode was not recognized.

AUTO
Automatically sets the scene based on content.

AUTO_24P_SYNC
Automatically selects "Cinema" mode for 24Hz content, otherwise same as AUTO.

GENERAL
Turns off scene select.

braviaproapi.bravia.errors package

braviaproapi.bravia.errors.ApiError

class braviaproapi.bravia.errors.**ApiError**
Bases: `Exception`

An error occurred while making an API request.

braviaproapi.bravia.errors.AppLaunchError

class braviaproapi.bravia.errors.**AppLaunchError**
Bases: `braviaproapi.bravia.errors.apierror.ApiError`

The requested app could not be launched.

braviaproapi.bravia.errors.EncryptionError

class braviaproapi.bravia.errors.**EncryptionError**
Bases: `braviaproapi.bravia.errors.apierror.ApiError`

An error occurred while encrypting or decrypting a message.

braviaproapi.bravia.errors.HttpError

class braviaproapi.bravia.errors.**HttpError**(*message*, *error_code=None*)
Bases: `Exception`

An error occurred while communicating with the API.

error_code
The error code returned by the API, if any.

Type str or None

braviaproapi.bravia.errors.InternalError

class braviaproapi.bravia.errors.**InternalError**
Bases: Exception

An internal error occurred in the API client.

braviaproapi.bravia.errors.InvalidStateError

class braviaproapi.bravia.errors.**InvalidStateError**
Bases: braviaproapi.bravia.errors.apierror.ApiError

The device is not in a state where it can accept the request.

braviaproapi.bravia.errors.LanguageNotSupportedError

class braviaproapi.bravia.errors.**LanguageNotSupportedError**
Bases: braviaproapi.bravia.errors.apierror.ApiError

The specified UI language is not supported by the device.

braviaproapi.bravia.errors.NoFocusedTextFieldError

class braviaproapi.bravia.errors.**NoFocusedTextFieldError**
Bases: braviaproapi.bravia.errors.apierror.ApiError

There is no text field focused on the device.

braviaproapi.bravia.errors.TargetNotSupportedError

class braviaproapi.bravia.errors.**TargetNotSupportedError**
Bases: braviaproapi.bravia.errors.apierror.ApiError

The specified target is not supported by the device.

braviaproapi.bravia.errors.VolumeOutOfRangeError

class braviaproapi.bravia.errors.**VolumeOutOfRangeError**
Bases: braviaproapi.bravia.errors.apierror.ApiError

The specified volume level is out of range.

Index

F

G

H

I

L

M

N

O

P

R

S