pyps4-2ndscreen

Release 1.3.1

ktnrg45

Aug 03, 2021

CONTENTS

1 PyPS4-2ndScreen **1**
- 1.1 Description . . . 1
- 1.2 Features . . . 1
- 1.3 Compatibility . . . 1
- 1.4 Installation . . . 2
- 1.5 Protocol . . . 2
- 1.6 Ports . . . 2
- 1.7 Cover Art Issues . . . 2
- 1.8 Known Issues . . . 2
- 1.9 Credits . . . 3
- 1.10 References . . . 3

2 Usage **5**
- 2.1 Using Ps4Async Example . . . 5
- 2.2 Getting Credentials . . . 7

3 Command Line **9**
- 3.1 Commands . . . 9
- 3.2 Arguments . . . 10
- 3.3 Buttons . . . 10

4 Credential **11**
- 4.1 Service . . . 11

5 DDP (Device Discovery Protocol) **13**
- 5.1 DDP Protocol . . . 13

6 PS4 **15**
- 6.1 Base Version . . . 15
- 6.2 Async Version . . . 16
- 6.3 Legacy Version . . . 17

7 Indices and tables **19**

Index **21**

CHAPTER

ONE

PYPS4-2NDSCREEN

1.1 Description

A full Python implementation based on the Node.js package, ps4-waker, which is an unofficial API for the PS4 2nd Screen App.

This module is mainly targeted towards developers although the module does include a basic CLI.

Disclaimer: This project/module and I are not affiliated with or endorsed by Sony Interactive Entertainment LLC. As such this project may break at any time.

1.2 Features

This module can perform almost every feature found in the PS4 Second Screen App.

- PS4 power and playing media state/status reporting
- Remote control
- Power on and standby control
- Starting a specific game/media
- Media information retrieval from the Playstation Store

1.3 Compatibility

Tested on:

- Environment: Python 3.6/3.7/3.8
- Operating System: Debian

1.4 Installation

Package can be installed with pip or from source.

It is advised to install the module in a virtual env.

Create virtual env first:

```
python -m venv .
source bin/activate
```

To install from pip:

```
pip install pyps4-2ndscreen
```

To install from source clone this repository and run from top-level:

```
pip install -r requirements.txt
python setup.py install
```

1.5 Protocol

UDP is used to get status updates and retrieve user credentials. TCP is used to send commands to the PS4 Console.

1.6 Ports

This module uses UDP port 1987 by default as the source port for polling the PS4.

PS4 listens on ports 987 (Priveleged) to fetch user PSN credentials.

In order to obtain user credentials, the Python Interpreter needs access to port 987 on the host system. The credential service pretends to be a PS4 console and will receive broadcast packets from the PS4 2nd Screen app on port 987.

Example:

```
sudo setcap 'cap_net_bind_service=+ep' /usr/bin/python3.5
```

This is so you do not need sudo/root priveleges to run.

1.7 Cover Art Issues

If you find that media art cannot be found. Please post an issue with your Region, Country, Title of game, an ID of game.

1.8 Known Issues

- PS Command inconsistent.
- On-Screen Keyboard is not implemented.

1.9 Credits

Thanks to hthiery for writing the underlying socket protocol in Python. https://github.com/hthiery/python-ps4

1.10 References

- https://github.com/dsokoloski/ps4-wake
- https://github.com/dhleong/ps4-waker
- https://github.com/hthiery/python-ps4

CHAPTER

TWO

USAGE

API should be accessed initially like the following examples.

Most functions can be accessed from the high-level PS4 object. You will need to pass in the IP address of your PS4 and your PSN credentials.

There are several async/asyncio coroutine functions in this module. These functions should be accessed with an Asyncio loop.

There are two versions of the Ps4 object/class: *pyps4_2ndscreen.ps4.Ps4Legacy* and *pyps4_2ndscreen.ps4.Ps4Async*. The *pyps4_2ndscreen.ps4.Ps4Async* version is recommended over the *pyps4_2ndscreen.ps4.Ps4Legacy* version which may be deprecated in the future. The difference between the two is that the *pyps4_2ndscreen.ps4.Ps4Legacy* class uses synchronous sockets (socket.socket) while the *pyps4_2ndscreen.ps4.Ps4Async* class uses asyncio transports and protocols. If using the Async version, a running asyncio event loop is required.

pyps4_2ndscreen.ps4.Ps4Legacy is suited for running single commands.

pyps4_2ndscreen.ps4.Ps4Async is best suited for a runtime environment/application.

2.1 Using Ps4Async Example

2.1.1 Initializing

- First start an asyncio event loop in the main thread.

```
import asyncio

task = asyncio.ensure_future(YourProgram())
loop = asyncio.get_event_loop()
loop.run_until_complete()
```

- Next you need to init the Device Discovery Protocol. This will enable you to get regular status updates.

```
from pyps4_2ndscreen.ddp import async_create_ddp_endpoint

_, ddp_protocol = await async_create_ddp_endpoint()
```

- Then you can instantiate the Ps4Async class and assign the ddp_protocol object to the ps4 object.

```
from pyps4_2ndscreen.ps4 import Ps4Async

ip_address = '192.168.0.3'
```

(continues on next page)

(continued from previous page)

```
creds = 'yourcredentials'
ps4 = Ps4Async(ip_address, creds)
ps4.set_protocol(ddp_protocol)
```

2.1.2 Getting Status

Status messages include various details such as the PS4 Standby/On status and the current game playing.

To get the status of the PS4 simply call:

```
status = ps4.get_status()
```

When the PS4 console is on and you have assigned the DDP Protocol to the Ps4 instance, you can add a callback to be called when the PS4 status updates. The callback must be a callable with no arguments.

```
ps4.add_callback(YourCallback)
```

The DDP protocol will now handle polling the PS4 console. However, when the PS4 goes into standby or is turned off, you will have to poll the PS4 yourself.

2.1.3 Controlling

To turn on from standby call:

```
ps4.wakeup()
```

In order to control the PS4 in other ways, you have to login as a registered PSN user on your PS4.

- First you should connect. This merely connects a TCP connection with the PS4. Sometimes the PS4 will refuse the connection, especially right after turning it on.

```
await ps4.async_connect()
```

- Then you can login.

```
await ps4.login()
```

The following are supported actions. (Logging in should be handled automatically):

- Turning Off

```
await ps4.standby()
```

- Launching a game or an app

```
await ps4.start_title('CUS10000')
```

- Navigation functions

```
await ps4.remote_control('ps')
```

To logout we simply drop the connection.

```
await ps4.close()
```

2.1.4 Getting Title Information

The PS4 object provides a coroutine to fetch information of a title from the PSN store. You will need the following information: - The title ID - The title name - PSN Region of title

The first two can be retreived from the status dictionary like so.

```
status = ps4.status
title_id = status.get('running-app-titleid')
title_name = status.get('running-app-name')
```

The PSN region needs to be a key in the following dictionary. You can verify like below:

```
from pyps4_2ndscreen.media_art import COUNTRIES

def check_region():
    YourRegion = 'United States'
    if YourRegion in COUNTRIES:
        return True
    return False
```

You can call the search coroutine now and retrieve info like the url for the cover:

```
result = await ps4.async_get_ps_store_data(title_name, title_id, YourRegion)
cover = result.cover_art
```

2.2 Getting Credentials

Your PSN Credentials can be generated by running a CLI command or by using the Python interpreter:

Terminal Command:

```
pyps4-2ndscreen credential
```

If your system does not have setcap utilities your may run the following command:

```
sudo ./bin/pyps4-2ndscreen credential
```

or

Python:

```
from pyps4_2ndscreen.credential import Credentials
creds = Credentials()

YourCredentials = creds.start()
```

This will start the credential service and will return the credentials for the PSN Account. You will need to get the PS4 Second Screen app for Android or iOS to do this. Once you have logged in with your acccount in the app and started the service, refresh the devices in the app and select the device named 'pyps4-2ndscreen'.

CHAPTER

THREE

COMMAND LINE

Usage: pyps4-2ndscreen [OPTIONS] COMMAND [ARGS]...

Pyps4-2ndscreen CLI. Allows for simple commands from terminal.

Example:

```
pyps4-2ndscreen -p 1234 start CUSA10000 -i 192.168.0.1 -c yourCredentials
```

Options:

--version Show the version and exit.

-v, --debug Enable debug logging.

-p, --port INTEGER Local UDP Port to use.

--help Show this message and exit.

Parameters:

--ip_address, -i IP Address of PS4.

--credentials, -c Credentials to use.

3.1 Commands

Table 1: Commands

Command	Description
credential	Get PSN Credentials.
link	Configure/Link PS4.
remote	Send Remote Control.
search	Search for PS4 devices.
standby	Place PS4 in Standby.
start	Start title.
status	Get status of PS4.
wakeup	Wakeup PS4.
interactive	Toggle interactive mode for continuous control.

3.2 Arguments

Table 2: Arguments

Argument	Description
Title ID	Title ID when using command *start*.
Button	Button when using command *remote*.

3.3 Buttons

Note: Not to be confused with DualShock4 Buttons.

Table 3: Buttons

Button	Description
ps	PS (PlayStation)
ps_hold	PS Hold/Long Press
enter	Enter
option	Option
left	Swipe Left
right	Swipe Right
up	Swipe Up
down	Swipe Down

CHAPTER

FOUR

CREDENTIAL

The *Credential* module allows for fetching of PSN user credentials using the official **PS4 2nd Screen App** on iOS and Android.

4.1 Service

The *pyps4_2ndscreen.credential.Credentials* class is a service to allow fetching of PSN credentials. **Note:** The service requires *port 987*. This is a priveleged port so some operating systems will not allow you to use this port without root/sudo privileges. If you would like to use the service with normal privileges you can try the command below:

```
sudo setcap 'cap_net_bind_service=+ep' /usr/bin/python3.5
```

This command works for Debian based systems. The */usr/bin/python3.5* should be replaced with the absolute path to your Python interpreter.

class pyps4_2ndscreen.credential.**Credentials**(*device_name: Optional[str] = 'pyps4-2ndscreen'*, *start: Optional[bool] = True*)

Bases: object

The PSN Credentials Service. Masquerades as a PS4 to get credentials.

Service listens on port 987 (Priveleged).

Parameters

- **device_name** – Name to display as
- **start** – Start on init

listen(*timeout: Optional[int] = 120*)

Listen and respond to requests.

Parameters **timeout** – Timeout in seconds

CHAPTER

FIVE

DDP (DEVICE DISCOVERY PROTOCOL)

The DDP module allows for discovery of PS4 devices.

5.1 DDP Protocol

The *pyps4_2ndscreen.ddp.DDPProtocol* is a handler for DDP/UDP messages. This class must be used in the event loop. It can handle multiple *pyps4_2ndscreen.ps4.Ps4Async* objects.

class pyps4_2ndscreen.ddp.**DDPProtocol**(*max_polls=5*)

Bases: asyncio.protocols.DatagramProtocol

Async UDP Client.

set_max_polls(*poll_count: int*)

Set number of unreturned polls neeeded to assume no status.

close()

Close Transport.

add_callback(*ps4*, *callback*)

Add callback to list. One per PS4 Object.

remove_callback(*ps4*, *callback*)

Remove callback from list.

property local_port

Return local port.

property remote_port

Return remote port.

property polls_disabled

Return true if polls disabled.

async ddp.**async_create_ddp_endpoint**(*port=1987*)

Create Async UDP endpoint.

CHAPTER

SIX

PS4

The PS4 class is the main object for interfacing with a PS4 console. You should only call methods directly from this class. There are two versions: *pyps4_2ndscreen.ps4.Ps4Async* and *pyps4_2ndscreen.ps4.Ps4Legacy*.

6.1 Base Version

The *pyps4_2ndscreen.ps4.Ps4Base* class should not be used directly.

class pyps4_2ndscreen.ps4.**Ps4Base**(*host: str*, *credential: str*, *device_name: Optional[str] = 'pyps4-2ndscreen'*, *port: Optional[int] = 0*)

Bases: object

The PS4 base object. Should not be initialized directly.

Parameters

- **host** – The host PS4 IP address
- **credential** – The credentials of a PSN account
- **device_name** – Name for client device
- **port** – Local UDP Port to use

change_port(*port*)
Change DDP Port.

get_status() → dict
Return current status info.

async async_get_ps_store_data(*title: str*, *title_id: str*, *region: str*) → pyps4_2ndscreen.media_art.ResultItem
Return title data from PS Store.

property port
Return local port.

property status_code
Return status code.

property is_running
Return True if the PS4 is running.

property is_standby
Return True if the PS4 is in standby.

property is_available
Return True if the PS4 is available.

property connected
Return True if connected to PS4.

property system_version
Return the system version.

property host_id
Return the host id/MAC address.

property host_name
Return the host name.

property running_app_titleid
Return the title ID of the running application.

property running_app_name
Return the name of the running application.

property running_app_ps_cover
Return the URL for the title cover art.

property running_app_ps_name
Return the name fetched from PS Store.

6.2 Async Version

pyps4_2ndscreen.ps4.Ps4Async is the recommended class. It is best suited for runtime applications. You should have an asyncio event loop running to call/await its coroutines.

class pyps4_2ndscreen.ps4.**Ps4Async**(*host: str*, *credential: str*, *device_name: Optional[str] = 'pyps4-2ndscreen'*, *port: Optional[int] = 0*)
Bases: *pyps4_2ndscreen.ps4.Ps4Base*

Async Version of Ps4 Class.

Parameters

- **host** – The host PS4 IP address
- **credential** – The credentials of a PSN account
- **device_name** – Name for device

set_login_delay(*value: int*)
Set delay for login.

set_protocol(*ddp_protocol: pyps4_2ndscreen.ddp.DDPProtocol*)
Attach DDP protocol.

Parameters **ddp_protocol** –

class *pyps4_2ndscreen.ddp.DDPProtocol*

add_callback(*callback: callable*)
Add status updated callback.

Parameters **callback** – Callback to call on status updated; No args

get_status() → dict
Get current status info.

wakeup(*ignore_conflict=False*)
Send Wakeup packet.

async change_ddp_endpoint(*port: int*, *close_old: bool = False*)
Return True if new endpoint is created.

async get_ddp_endpoint()
Return True if endpoint is created from socket.

async login(*pin: Optional[str] = ''*)
Send Login Packet.

Parameters **pin** – Pin to send. Requred when linking.

async standby(*ignore_conflict=False*)
Send Standby Packet.

async toggle()
Toggle Power.

async start_title(*title_id: str*, *running_id: Optional[str] = None*)
Send start title packet.

Closes current title if title_id is running_id

Parameters

- **title_id** – Title to start; CUSA00000
- **running_id** – Title currently running

async remote_control(*button_name: str*, *hold_time: Optional[int] = 0*)
Send remote control command packet. Is coroutine.

Parameters

- **button_name** – Button to send to PS4.
- **hold_time** – Time to hold in millis. Only affects PS command.

async close()
Close Connection.

async async_connect(*auto_login: Optional[bool] = True*)
Connect.

Parameters **auto_login** – If true will login automatically if powering on.

property login_delay
Return login delay value.

6.3 Legacy Version

pyps4_2ndscreen.ps4.Ps4Legacy is best suited for one-time commands.

class pyps4_2ndscreen.ps4.**Ps4Legacy**(*host: str*, *credential: str*, *device_name: Optional[str] = 'pyps4-2ndscreen'*, *auto_close: Optional[bool] = True*, *port: Optional[int] = 0*)
Bases: *pyps4_2ndscreen.ps4.Ps4Base*

Legacy PS4 Class. Sync Version.

Parameters

- **host** – The host PS4 IP address
- **credential** – The credentials of a PSN account
- **device_name** – Name for device

close()
Close the connection to the PS4.

wakeup()
Send Wakeup Packet.

login(*pin: Optional[str] = ''*) → bool
Send Login Packet.

Parameters **pin** – Pin to send. Requred when linking.

standby() → bool
Send Standby Packet.

start_title(*title_id*, *running_id: Optional[str] = None*) → bool
Send Start title packet.

Close current title if title_id is running_id

Parameters

- **title_id** – Title to start; CUSA00000
- **running_id** – Title currently running

remote_control(*button_name*, *hold_time: Optional[int] = 0*) → bool
Send remote control command packet.

Parameters

- **button_name** – Button to send to PS4.
- **hold_time** – Time to hold in millis. Only affects PS command.

send_status() → bool
Send connection status ack to PS4.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

A

add_callback() (*pyps4_2ndscreen.ddp.DDPProtocol method*), 13
add_callback() (*pyps4_2ndscreen.ps4.Ps4Async method*), 16
async_connect() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
async_create_ddp_endpoint() (*pyps4_2ndscreen.ddp method*), 13
async_get_ps_store_data() (*pyps4_2ndscreen.ps4.Ps4Base method*), 15

C

change_ddp_endpoint() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
change_port() (*pyps4_2ndscreen.ps4.Ps4Base method*), 15
close() (*pyps4_2ndscreen.ddp.DDPProtocol method*), 13
close() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
close() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18
connected() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16
Credentials (*class in pyps4_2ndscreen.credential*), 11

D

DDPProtocol (*class in pyps4_2ndscreen.ddp*), 13

G

get_ddp_endpoint() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
get_status() (*pyps4_2ndscreen.ps4.Ps4Async method*), 16
get_status() (*pyps4_2ndscreen.ps4.Ps4Base method*), 15

H

host_id() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16
host_name() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16

I

is_available() (*pyps4_2ndscreen.ps4.Ps4Base property*), 15
is_running() (*pyps4_2ndscreen.ps4.Ps4Base property*), 15
is_standby() (*pyps4_2ndscreen.ps4.Ps4Base property*), 15

L

listen() (*pyps4_2ndscreen.credential.Credentials method*), 11
local_port() (*pyps4_2ndscreen.ddp.DDPProtocol property*), 13
login() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
login() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18
login_delay() (*pyps4_2ndscreen.ps4.Ps4Async property*), 17

P

polls_disabled() (*pyps4_2ndscreen.ddp.DDPProtocol property*), 13
port() (*pyps4_2ndscreen.ps4.Ps4Base property*), 15
Ps4Async (*class in pyps4_2ndscreen.ps4*), 16
Ps4Base (*class in pyps4_2ndscreen.ps4*), 15
Ps4Legacy (*class in pyps4_2ndscreen.ps4*), 17

R

remote_control() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
remote_control() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18
remote_port() (*pyps4_2ndscreen.ddp.DDPProtocol property*), 13
remove_callback() (*pyps4_2ndscreen.ddp.DDPProtocol method*), 13

running_app_name() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16
running_app_ps_cover() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16
running_app_ps_name() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16
running_app_titleid() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16

S

send_status() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18
set_login_delay() (*pyps4_2ndscreen.ps4.Ps4Async method*), 16
set_max_polls() (*pyps4_2ndscreen.ddp.DDPProtocol method*), 13
set_protocol() (*pyps4_2ndscreen.ps4.Ps4Async method*), 16
standby() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
standby() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18
start_title() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17
start_title() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18
status_code() (*pyps4_2ndscreen.ps4.Ps4Base property*), 15
system_version() (*pyps4_2ndscreen.ps4.Ps4Base property*), 16

T

toggle() (*pyps4_2ndscreen.ps4.Ps4Async method*), 17

W

wakeup() (*pyps4_2ndscreen.ps4.Ps4Async method*), 16
wakeup() (*pyps4_2ndscreen.ps4.Ps4Legacy method*), 18