Plyer Documentation

Release 1.2.5dev

Mathieu Virbel, Akshay Aurora, Gabriel Petier, Ben Rousch

February 07, 2017

Contents

1 Plyer 3

2 Facades 5

3 Indices and tables 11

Python Module Index 13

Plyer is a Python library for accessing features of your hardware / platforms.

CHAPTER 1

Plyer

plyer.**accelerometer** = **<plyer.platforms.linux.accelerometer.LinuxAccelerometer object>**
Accelerometer proxy to *plyer.facades.Accelerometer*

plyer.**audio**
Audio proxy to *plyer.facades.Audio*

plyer.**battery** = **<plyer.facades.battery.Battery object>**
Battery proxy to *plyer.facades.Battery*

plyer.**call** = **<plyer.facades.call.Call object>**
Call proxy to :class *plyer.facades.Call*

plyer.**camera** = **<plyer.facades.camera.Camera object>**
Camera proxy to *plyer.facades.Camera*

plyer.**compass** = **<plyer.facades.compass.Compass object>**
Compass proxy to *plyer.facades.Compass*

plyer.**email** = **<plyer.platforms.linux.email.LinuxEmail object>**
Email proxy to *plyer.facades.Email*

plyer.**filechooser** = **<plyer.platforms.linux.filechooser.LinuxFileChooser object>**
FileChooser proxy to *plyer.facades.FileChooser*

plyer.**gps** = **<plyer.facades.gps.GPS object>**
GPS proxy to *plyer.facades.GPS*

plyer.**gyroscope** = **<plyer.facades.gyroscope.Gyroscope object>**
Gyroscope proxy to *plyer.facades.Gyroscope*

plyer.**irblaster** = **<plyer.facades.irblaster.IrBlaster object>**
IrBlaster proxy to *plyer.facades.IrBlaster*

plyer.**orientation** = **<plyer.facades.orientation.Orientation object>**
Orientation proxy to *plyer.facades.Orientation*

plyer.**notification** = **<plyer.facades.notification.Notification object>**
Notification proxy to *plyer.facades.Notification*

plyer.**sms** = **<plyer.facades.sms.Sms object>**
Sms proxy to *plyer.facades.Sms*

plyer.**tts** = **<plyer.facades.tts.TTS object>**
TTS proxy to *plyer.facades.TTS*

plyer.**uniqueid** = **<plyer.facades.uniqueid.UniqueID object>**
UniqueID proxy to *plyer.facades.UniqueID*

`plyer.`**`vibrator`** **= <plyer.facades.vibrator.Vibrator object>**

Vibrator proxy to *`plyer.facades.Vibrator`*

`plyer.`**`wifi`** **= <plyer.platforms.linux.wifi.LinuxWifi object>**

Wifi proxy to *`plyer.facades.Wifi`*

CHAPTER 2

Facades

Interface of all the features available.

class `plyer.facades.`**`Accelerometer`**

Accelerometer facade.

`acceleration`

Property that returns values of the current acceleration sensors, as a (x, y, z) tuple. Returns (None, None, None) if no data is currently available.

`disable``()`

Disable the accelerometer sensor.

`enable``()`

Activate the accelerometer sensor. Throws an error if the hardware is not available or not implemented on.

class `plyer.facades.`**`Audio`**`(`*file_path*`)`

Audio facade.

`play``()`

Play current recording.

`start``()`

Start record.

`stop``()`

Stop record.

class `plyer.facades.`**`Battery`**

Battery info facade.

`status`

Property that contains a dict with the following fields:

- **isCharging** *(bool)*: Battery is charging
- **percentage** *(float)*: Battery charge remaining

> **Warning:** If any of the fields is not readable, it is set as None.

class `plyer.facades.`**`Call`**

Call facade.

`dialcall``()`

Opens dialing interface.

makecall(*tel*)
Make calls using your device.

Parameters **tel** (*number*) – The reciever

class plyer.facades.**Camera**
Camera facade.

take_picture(*filename*, *on_complete*)
Ask the OS to capture a picture, and store it at filename.

When the capture is done, on_complete will be called with the filename as an argument. If the callback returns True, the filename will be unlinked.

Parameters

- **filename** (*str*) – Name of the image file
- **on_complete** (*callable*) – Callback that will be called when the operation is done

take_video(*filename*, *on_complete*)
Ask the OS to capture a video, and store it at filename.

When the capture is done, on_complete will be called with the filename as an argument. If the callback returns True, the filename will be unlinked.

Parameters

- **filename** (*str*) – Name of the video file
- **on_complete** (*callable*) – Callback that will be called when the operation is done

class plyer.facades.**Compass**
Compass facade.

New in version 1.2.0.

disable()
Disable the compass sensor.

enable()
Activate the compass sensor.

orientation
Property that returns values of the current compass (magnetic field) sensors, as a (x, y, z) tuple. Returns (None, None, None) if no data is currently available.

class plyer.facades.**Email**
Email facade.

send(*recipient=None*, *subject=None*, *text=None*, *create_chooser=None*)
Open an email client message send window, prepopulated with the given arguments.

Parameters

- **recipient** – Recipient of the message (str)
- **subject** – Subject of the message (str)
- **text** – Main body of the message (str)
- **create_chooser** – Whether to display a program chooser to handle the message (bool)

Note: create_chooser is only supported on Android

class `plyer.facades.FileChooser`

File Chooser facade.

choose_dir(**args*, ***kwargs*)

Open the directory chooser. Note that on Windows this is very limited. Consider writing your own chooser if you target that platform and are planning on using unsupported features.

open_file(**args*, ***kwargs*)

Open the file chooser in "open" mode.

save_file(**args*, ***kwargs*)

Open the file chooser in "save" mode. Confirmation will be asked when a file with the same name already exists.

class `plyer.facades.GPS`

GPS facade.

configure(*on_location*, *on_status=None*)

Configure the GPS object. This method should be called before `start()`.

Parameters

- **on_location** (*callable, multiples keys/value will be passed.*) – Function to call when receiving a new location
- **on_status** (*callable, args are "message-type", "status"*) – Function to call when a status message is received

> **Warning:** The *on_location* and *on_status* callables might be called from another thread than the thread used for creating the GPS object.

start(*minTime=1000*, *minDistance=1*)

Start the GPS location updates. Expects 2 parameters:

minTime: milliseconds. (float) minDistance: meters. (float)

stop()

Stop the GPS location updates.

class `plyer.facades.Gyroscope`

Gyroscope facade.

New in version 1.2.0.

disable()

Disable the Gyroscope sensor.

enable()

Activate the Gyroscope sensor.

orientation

Property that returns values of the current Gyroscope sensors, as a (x, y, z) tuple. Returns (None, None, None) if no data is currently available.

class `plyer.facades.IrBlaster`

Infrared blaster facade.

exists()

Check if the device has an infrared emitter.

frequencies

Property which contains a list of frequency ranges supported by the device in the form:

[(from1, to1), (from2, to2), ... (fromN, toN)]

static `microseconds_to_periods` (*frequency*, *pattern*)
Convert a pattern from microseconds to period counts.

static `periods_to_microseconds` (*frequency*, *pattern*)
Convert a pattern from period counts to microseconds.

`transmit` (*frequency*, *pattern*, *mode='period'*)
Transmit an IR sequence.

Parameters

***frequency*: int** Carrier frequency for the IR transmission.

***pattern*: list[int]** Burst pair pattern to transmit.

***mode*: str, defaults to 'period'** Specifies the format of the pattern values. Can be 'period' or 'microseconds'.

class `plyer.facades.Orientation`
Orientation facade.

`set_landscape` (*reverse=False*)
Rotate the app to a landscape orientation.

Parameters `reverse` – If True, uses the opposite of the natural orientation.

`set_portrait` (*reverse=False*)
Rotate the app to a portrait orientation.

Parameters `reverse` – If True, uses the opposite of the natural orientation.

`set_sensor` (*mode='any'*)
Rotate freely following sensor information from the device.

Parameters `mode` – The rotation mode, should be one of 'any' (rotate to any orientation), 'landscape' (choose nearest landscape mode) or 'portrait' (choose nearest portrait mode). Defaults to 'any'.

class `plyer.facades.Notification`
Notification facade.

`notify` (*title='', message='', app_name='', app_icon='', timeout=10, ticker=''*)
Send a notification.

Parameters

- `title` (*str*) – Title of the notification
- `message` (*str*) – Message of the notification
- `app_name` (*str*) – Name of the app launching this notification
- `app_icon` (*str*) – Icon to be displayed along with the message
- `timeout` (*int*) – time to display the message for, defaults to 10
- `ticker` (*str*) – text to display on status bar as the notification arrives

class `plyer.facades.Sms`
Sms facade.

`send` (*recipient*, *message*)
Send SMS or open SMS interface.

Parameters

- **recipient** (*number*) – The reveiver
- **message** (*str*) – the message

class plyer.facades.**TTS**
TextToSpeech facade.

speak(*message=''*)
Use text to speech capabilities to speak the message.

Parameters **message** (*str*) – What to speak

class plyer.facades.**UniqueID**
UniqueID facade.

id
Property that returns the unique id of the platform.

class plyer.facades.**Vibrator**
Vibration facade.

cancel()
Cancels any current vibration, and stops the vibrator.

exists()
Check if the device has a vibrator. Returns True or False.

pattern(*pattern=(0, 1), repeat=-1*)
Ask the vibrator to vibrate with the given pattern, with an optional repeat.

Parameters

- **pattern** – Pattern to vibrate with. Should be a list of times in seconds. The first number is how long to wait before vibrating, and subsequent numbers are times to vibrate and not vibrate alternately. Defaults to `[0, 1]`.
- **repeat** – Index at which to repeat the pattern. When the vibration pattern reaches this index, it will start again from the beginning. Defaults to `-1`, which means no repeat.

vibrate(*time=1*)
Ask the vibrator to vibrate for the given period.

Parameters **time** – Time to vibrate for, in seconds. Default is 1.

class plyer.facades.**Wifi**
Wifi Facade.

connect(*network, parameters*)
Method to connect to some network.

disconnect()
To disconnect from some network.

get_available_wifi()
Returns a list of all the available wifi.

get_network_info(*name*)
Return a dictionary of secified network.

is_enabled()
Returns *True'if the Wifi is enables else 'False*.

start_scanning()
Turn on scanning.

class `plyer.facades.`**`Flash`**

Flash facade.

`off`()

Deactiavte the flash

`on`()

Activate the flash

`release`()

Release any access to the Flash / Camera. Call this when you're done using the Flash. This will release the Camera, and stop any process.

Next call to *_on* will reactivate it.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

p

plyer, 1
plyer.facades, 4

A

acceleration (plyer.facades.Accelerometer attribute), 5
Accelerometer (class in plyer.facades), 5
accelerometer (in module plyer), 3
Audio (class in plyer.facades), 5
audio (in module plyer), 3

B

Battery (class in plyer.facades), 5
battery (in module plyer), 3

C

Call (class in plyer.facades), 5
call (in module plyer), 3
Camera (class in plyer.facades), 6
camera (in module plyer), 3
cancel() (plyer.facades.Vibrator method), 9
choose_dir() (plyer.facades.FileChooser method), 7
Compass (class in plyer.facades), 6
compass (in module plyer), 3
configure() (plyer.facades.GPS method), 7
connect() (plyer.facades.Wifi method), 9

D

dialcall() (plyer.facades.Call method), 5
disable() (plyer.facades.Accelerometer method), 5
disable() (plyer.facades.Compass method), 6
disable() (plyer.facades.Gyroscope method), 7
disconnect() (plyer.facades.Wifi method), 9

E

Email (class in plyer.facades), 6
email (in module plyer), 3
enable() (plyer.facades.Accelerometer method), 5
enable() (plyer.facades.Compass method), 6
enable() (plyer.facades.Gyroscope method), 7
exists() (plyer.facades.IrBlaster method), 7
exists() (plyer.facades.Vibrator method), 9

F

FileChooser (class in plyer.facades), 6
filechooser (in module plyer), 3
Flash (class in plyer.facades), 9
frequencies (plyer.facades.IrBlaster attribute), 7

G

get_available_wifi() (plyer.facades.Wifi method), 9
get_network_info() (plyer.facades.Wifi method), 9
GPS (class in plyer.facades), 7
gps (in module plyer), 3
Gyroscope (class in plyer.facades), 7
gyroscope (in module plyer), 3

I

id (plyer.facades.UniqueID attribute), 9
IrBlaster (class in plyer.facades), 7
irblaster (in module plyer), 3
is_enabled() (plyer.facades.Wifi method), 9

M

makecall() (plyer.facades.Call method), 5
microseconds_to_periods() (plyer.facades.IrBlaster static method), 8

N

Notification (class in plyer.facades), 8
notification (in module plyer), 3
notify() (plyer.facades.Notification method), 8

O

off() (plyer.facades.Flash method), 10
on() (plyer.facades.Flash method), 10
open_file() (plyer.facades.FileChooser method), 7
Orientation (class in plyer.facades), 8
orientation (in module plyer), 3
orientation (plyer.facades.Compass attribute), 6
orientation (plyer.facades.Gyroscope attribute), 7

P

pattern() (plyer.facades.Vibrator method), 9
periods_to_microseconds() (plyer.facades.IrBlaster static method), 8
play() (plyer.facades.Audio method), 5
plyer (module), 1
plyer.facades (module), 4

R

release() (plyer.facades.Flash method), 10

S

save_file() (plyer.facades.FileChooser method), 7
send() (plyer.facades.Email method), 6
send() (plyer.facades.Sms method), 8
set_landscape() (plyer.facades.Orientation method), 8
set_portrait() (plyer.facades.Orientation method), 8
set_sensor() (plyer.facades.Orientation method), 8
Sms (class in plyer.facades), 8
sms (in module plyer), 3
speak() (plyer.facades.TTS method), 9
start() (plyer.facades.Audio method), 5
start() (plyer.facades.GPS method), 7
start_scanning() (plyer.facades.Wifi method), 9
status (plyer.facades.Battery attribute), 5
stop() (plyer.facades.Audio method), 5
stop() (plyer.facades.GPS method), 7

T

take_picture() (plyer.facades.Camera method), 6
take_video() (plyer.facades.Camera method), 6
transmit() (plyer.facades.IrBlaster method), 8
TTS (class in plyer.facades), 9
tts (in module plyer), 3

U

UniqueID (class in plyer.facades), 9
uniqueid (in module plyer), 3

V

vibrate() (plyer.facades.Vibrator method), 9
Vibrator (class in plyer.facades), 9
vibrator (in module plyer), 3

W

Wifi (class in plyer.facades), 9
wifi (in module plyer), 4