Landroid Cloud Client Documentation

Alexander Weggerle

Jul 14, 2021

Contents:

CHAPTER 1

Getting Started

1.1 Installation

Installation using pip

```
pip install landroidcc
```

1.2 Commandline usage

After installation the command 'landroidcc' is available and can be used for example to get the status and start the mower:

```
landroidcc username password --status --start
```

The output will look like:

```
2019-05-12 22:25:32 __init__ _api_authentificate INFO     Successfully logged in
2019-05-12 22:25:33 __init__ on_connect INFO     Successfully connected to the cloud
landroid info
#############
Name:   Schaf
Serial: xxxxxxxxxxxxxxxxxxx
Type:   WR141E

landroid status
###############
LastUpdate: 22:25:34 12/05/2019
State:      Home
Error:      No error
Battery:    100%/9.2C/19.63v
```

1.3 API Usage

For using the landroid client directly from another Python using the Landroid class. The status returned has the type: LandroidStatus

```
from landroidcc import Landroid

landroid = Landroid("user", "pass")
status = landroid.get_status()
print("Battery: {}%".format(status.get_battery().percent))
landroid.start()  # Start mowing
```

1.3.1 Update callback

Once connected the status gets updated automatically once the mower is sending an update. To get a notification it's possible to register a callback function. See *landroidcc.Landroid.set_statuscallback()*

CHAPTER 2

API Documentation

class landroidcc.**Landroid**

connect(*username*, *password*)
Connect to the cloud with the given credentials.

Parameters

- **username** – Username for the cloud login
- **password** – Password for the login

Returns None

disconnect()
Disconnects from the cloud

Returns None

get_status(*refresh=True*)
Returns the last retrieved status from the mower. If refresh is True an update is requested from the mower and the call will block until an update is received.

Once connected the status will automatically updated once the mower sent an automatic update message. This happens every 2-15 minutes and for all state changes.

Parameters **refresh** – Force an update or only return the cached last status

Return type *LandroidStatus*

Returns The status of the mower.

go_home()
Sent the mower the command to go home mowing

Returns None

pause()
Sent the mower the command to pause mowing

Returns None

set_statuscallback(*func*)

Sets a callback function which will be called for any status update from the mower:

```
def callback(status):
  # type: (LandroidStatus) -> None
  print (status)

landroid = Landroid()
landroid.connect("", "")
landroid.set_statuscallback(callback)
```

Parameters **func** – The callback

Returns None

start()

Sent the mower the command to start mowing

Returns None

class landroidcc.**LandroidStatus**(*inputraw*)

class BatteryStatus(*percent*, *charges*, *volts*, *temperature*, *charging*)

charges

Alias for field number 1

charging

Alias for field number 4

percent

Alias for field number 0

temperature

Alias for field number 3

volts

Alias for field number 2

class Orientation(*heading*, *pitch*, *roll*)

heading

Alias for field number 0

pitch

Alias for field number 1

roll

Alias for field number 2

class Statistics(*distance*, *running*, *mowing*)

distance

Alias for field number 0

mowing

Alias for field number 2

running
Alias for field number 1

get_battery()

Returns

Return type *BatteryStatus*

get_error()
Returns the error as string. If there is no error, "No Error" is returned

Returns The error as text

Return type str

get_orientation()

get_raw()
Returns the status update as received directly from MQTT/mower.

Returns Raw status message

Return type dict

get_state()
Returns the state as string

Returns The state as text

Return type str

get_statistics()

get_updated()

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index