
pyHomeKit Documentation

Release 0.0.1

Henri Dwyer

Oct 05, 2017

Contents

1	PyHomeKit - HomeKit for Python	1
1.1	Getting Started	1
1.2	Links	2
2	pyhomekit	3
2.1	pyhomekit package	3
3	Changelog	11
3.1	0.0.1.4	11
3.2	0.0.1	11
4	Indices and tables	13
	Python Module Index	15

PyHomeKit - HomeKit for Python

Author Henri Dwyer

PyHomeKit is a set of python libraries that let you control HomeKit compatible accessories, both BLE and HTTP.

Important: PyHomeKit is currently in pre-alpha. Many features are not yet implemented or broken.

Getting Started

Usage

Connect to a HAP characteristics and view its signature:

```
import pyhomekit

device_mac = "aa:aa:aa:aa:aa"

characteristic_uuid = "00000000-0000-0000-0000-000000000000"

accessory = pyhomekit.ble.HapAccessory(device_mac, 'random')
accessory.connect()
hap_characteristic = pyhomekit.ble.HapCharacteristic(accessory=accessory,
↳uuid=characteristic_uuid)

print(hap_characteristic.signature)
```

View the debug logs in stdout:

```
import logging

logging.basicConfig()
logging.getLogger('pyhomekit').setLevel(logging.DEBUG)
```

Installation

pyHomeKit is on Pypi, so you can pip install it:

```
pip install pyhomekit
```

If you want to install from source, clone the repository:

```
git clone git://github.com/henridwyer/pyhomekit.git
cd pyhomekit
pip install -r requirements.txt
pip install -e .
```

Then you can build the documentation:

```
make doc
```

And run the tests:

```
make tests
```

Links

For more information about HomeKit, see the [Apple Developer HomeKit page](#).

Bluetooth Low Energy device compatibility is provided by [bluepy](#).

Encryption is provided by the [libsodium](#) library.

pyhomekit package

Module contents

pyhomekit.ble module

Contains all of the HAP-BLE classes.

class `pyhomekit.ble.HapAccessory` (*address: str, address_type: str = 'static'*) → None

Bases: `object`

HAP Accesory.

Parameters

- **address** – MAC address of the accessory
- **address_type** – Type of the address: static or random

characteristic (*uuid: str*) → `sphinx.ext.autodoc.Characteristic`

Return the GATT characteristic for the given UUID.

connect () → None

Connect to BLE peripheral.

discover_hap_characteristics () → `typing.List[pyhomekit.ble.HapCharacteristic]`

Discovers all of the HAP Characteristics and performs a signature read on each one.

get_characteristic (*name: str, uuid: str*) → `pyhomekit.ble.HapCharacteristic`

pair () → None

pair_verify () → None

save_key () → None

class `pyhomekit.ble.HapAccessoryLock` (*address: str, address_type: str = 'static'*) → None

Bases: `pyhomekit.ble.HapAccessory`

administrator_only_access () → None

audio_feedback () → `bytes`

```
current_door_state() → int
lock_control_point() → typing.Any
lock_current_state() → int
lock_last_known_action() → int
lock_management_auto_security_timeout() → None
lock_target_state() → None
logs() → str
motion_detected() → bool
version() → str
```

```
class pyhomekit.ble.HapBlePdu(header: pyhomekit.ble.HapBlePduHeader, TLVs: typing.Sequence[typing.Tuple[int, bytes]]) → None
```

Bases: object

HAP BLE PDU

fragmented

max_len = 512

pdu_fragments() → typing.Iterator[bytes]

raw_data

```
class pyhomekit.ble.HapBlePduHeader(response: bool, continuation: bool) → None
```

Bases: object

Interface for HAP-BLE Headers.

This class is not meant to be instantiated. Use the children `HapBlePduRequestHeader` and `HapBlePduResponseHeader`.

Parameters

- **continuation** – indicates the fragmentation status of the HAP-BLE PDU. False indicates a first fragment or no fragmentation.
- **response** – indicates whether the PDU is a response (versus a request)

control_field

Get Control Field as int.

control_field_bits

Get Control Field as string of bits.

data

```
class pyhomekit.ble.HapBlePduRequestHeader(cid_sid: bytes, op_code: int, response: bool = False, continuation: bool = False, transaction_id: int = None) → None
```

Bases: `pyhomekit.ble.HapBlePduHeader`

HAP-BLE PDU Request Header.

Parameters

- **continuation** – indicates the fragmentation status of the HAP-BLE PDU. False indicates a first fragment or no fragmentation.
- **response** – indicates whether the PDU is a response (versus a request)
- **transaction_id** – Transaction Identifier
- **op_code** – HAP Opcode field, which indicates the opcode for the HAP Request PDU.

- **cid_sid** – Characteristic / Service Instance Identifier is the instance id of the characteristic / service for a particular request.

data

Byte representation of the PDU Header.

Depends on whether it is a continuation header or not.

transaction_id

Get the transaction identifier, or generate a new one if none exists.

The transaction ID is an 8 bit number identifying the transaction number of this PDU. The TID is randomly generated by the originator of the request and is used to match a request/response pair.

```
class pyhomekit.ble.HapBlePduResponseHeader(status_code: int, transaction_id: int, continuation: bool = False, response: bool = True) → None
```

Bases: `pyhomekit.ble.HapBlePduHeader`

HAP-BLE PDU Response Header.

Parameters

- **continuation** – indicates the fragmentation status of the HAP-BLE PDU. False indicates a first fragment or no fragmentation.
- **response** – indicates whether the PDU is a response (versus a request)
- **transaction_id** – Transaction Identifier
- **status_code** – HAP Status code for the request.

data

Byte representation of the PDU Header.

```
classmethod from_data(data: bytes) → pyhomekit.ble.HapBlePduResponseHeader
```

Creates a header from response bytes

```
class pyhomekit.ble.HapCharacteristic(accessory: pyhomekit.ble.HapAccessory, uuid: str, retry: bool = False, retry_max_attempts: int = 1, retry_wait_time: int = 2) → None
```

Bases: `object`

Represents data or an associated behavior of a service.

The characteristic is defined by a universally unique type, and has additional properties that determine how the value of the characteristic can be accessed.

Parameters

- **accessory** – The accessory this characteristic belongs to.
- **uuid** – The UUID of the underlying GATT characteristic
- **retry** – Attempt to reconnect when error.
- **retry_max_attempts** – How many times to attempt reconnection.
- **retry_wait_time** – How long to wait in s between reconnection attempts.

cid

Get the Characteristic ID, reading it from the device if required.

```
read(request_header: pyhomekit.ble.HapBlePduRequestHeader) → typing.Dict[str, typing.Any]
```

Perform a HAP Characteristic read.

Fragmented read if required.

signature

Returns the signature, and adds the attributes.

write (*request_header*: *pyhomekit.ble.HapBlePduRequestHeader*, *TLVs*: *typing.List[typing.Tuple[int, bytes]]*) → *typing.Dict[str, typing.Any]*
Perform a HAP Characteristic write.

Fragmented read/write if required.

write_ktlvs (*request_header*: *pyhomekit.ble.HapBlePduRequestHeader*, *kTLVs*: *typing.Sequence[typing.Tuple[int, bytes]]*) → *typing.Dict[str, typing.Any]*
Perform a HAP Characteristic write for a pairing.

Fragmented read/write if required.

pyhomekit.ble.fragment_tlvs (*header*: *pyhomekit.ble.HapBlePduRequestHeader*, *TLVs*: *typing.List[typing.Tuple[int, bytes]]*) → *typing.Iterator[bytes]*
Returns the fragmented TLVs to write.

pyhomekit.ble.reconnect_callback_factory (*accessory*: *pyhomekit.ble.HapAccessory*) → *typing.Callable[[typing.Any, int], NoneType]*
Factory for creating tenacity before callbacks to reconnect to a peripheral.

pyhomekit.ble.reconnect_tenacity_retry (*reconnect_callback*: *typing.Callable[[typing.Any, int], typing.Any]*, *max_attempts*: *int = 2*, *wait_time*: *int = 2*) → *tenacity.Retrying*
Build tenacity retry object

pyhomekit.constants module

HAP Constants

class *pyhomekit.constants.HapBleOpCodes*
Bases: *object*

HAP Opcode Descriptions.

Characteristic_Execute_Write = 5

Characteristic_Read = 3

Characteristic_Signature_Read = 1

Characteristic_Timed_Write = 4

Characteristic_Write = 2

Service_Signature_Read = 6

class *pyhomekit.constants.HapBleStatusCodes*
Bases: *object*

HAP Status code definitions and descriptions.

Insufficient_Authentication = 5

Insufficient_Authorization = 3

Invalid_Instance_ID = 4

Invalid_Request = 6

Max_Procedures = 2

Success = 0

Unsupported_PDU = 1

class *pyhomekit.constants.HapParamTypes*
Bases: *object*

Additional_Authorization_Data = 2

```
Characteristic_Instance_ID = 5
Characteristic_Type = 4
GATT_Presentation_Format_Descriptor = 12
GATT_User_Description_Descriptor = 11
GATT_Valid_Range = 13
HAP_Characteristic_Properties_Descriptor = 10
HAP_Linked_Services = 16
HAP_Service_Properties = 15
HAP_Step_Value_Descriptor = 14
HAP_Valid_Values_Descriptor = 17
HAP_Valid_Values_Range_Descriptor = 18
Origin_local_vs_remote = 3
Return_Response = 9
Service_Instance_ID = 7
Service_Type = 6
TTL = 8
Value = 1

class pyhomekit.constants.PairingKTLVErrorCodes
    Bases: object

    kTLVError_Authenticatio = 2
    kTLVError_Backof = 3
    kTLVError_Bus = 7
    kTLVError_MaxPeer = 4
    kTLVError_MaxTrie = 5
    kTLVError_Unavailabl = 6
    kTLVError_Unknow = 1

class pyhomekit.constants.PairingKTLVMethodValues
    Bases: object

    Pairing service kTLV method values

    Add_Pairing = 3
    List_Pairings = 5
    Pair_Setup = 1
    Pair_Verify = 2
    Remove_Pairing = 4
    Reserved = 0

class pyhomekit.constants.PairingKTLVValues
    Bases: object

    Pairng service TLV Values.

    kTLVType_Certificate = 9
    kTLVType_EncryptedData = 5
```

```
kTLVType_Error = 7
kTLVType_FragmentData = 12
kTLVType_FragmentLast = 13
kTLVType_Identifier = 1
kTLVType_Method = 0
kTLVType_Permissions = 11
kTLVType_Proof = 4
kTLVType_PublicKey = 3
kTLVType_RetryDelay = 8
kTLVType_Salt = 2
kTLVType_Separator = 255
kTLVType_Signature = 10
kTLVType_State = 6
```

`pyhomekit.constants.identity` (*x: typing.Any*) → *typing.Any*
Identity

`pyhomekit.constants.parse_format` (*b: bytes*) → *typing.Tuple[int, int]*
Parse the bluetooth characteristic presentation format to format and unit code

`pyhomekit.constants.to_bool` (*b: bytes*) → *bool*
Convert to bytes to bool (little endian).

`pyhomekit.constants.to_float` (*b: bytes*) → *int*
Convert to bytes to float (little endian).

`pyhomekit.constants.to_int32` (*b: bytes*) → *int*
Convert to bytes to 32 bit signed int (little endian).

`pyhomekit.constants.to_uint16` (*b: bytes*) → *int*
Convert to bytes to 16 bit unsigned short (little endian).

`pyhomekit.constants.to_uint32` (*b: bytes*) → *int*
Convert to bytes to 32 bit unsigned int (little endian).

`pyhomekit.constants.to_uint64` (*b: bytes*) → *int*
Convert to bytes to 64 bit unsigned int (little endian).

`pyhomekit.constants.to_uint8` (*b: bytes*) → *int*
Convert to bytes to 8 bit unsigned short (little endian).

`pyhomekit.constants.to_utf8` (*b: bytes*) → *str*
Convert bytes to str utf-8 encoded.

`pyhomekit.constants.to_uuid` (*b: bytes*) → *str*
Convert bytes to string representation of uuid.

The bytes are reversed first.

pyhomekit.utils module

Utility functions for BLE

exception `pyhomekit.utils.HapBleError` (*status_code: int = None, name: str = None, message: str = None, *args: str*) → *None*

Bases: *Exception*

HAP Error.

`pyhomekit.utils.iterate_tlv` (*response: bytes*) → `typing.Iterator[typing.Tuple[int, int, bytes]]`

Iterate through response bytes, 1 tlv at a time.

`pyhomekit.utils.parse_ktlvs` (*data: bytes*) → `typing.Dict[str, typing.Any]`

Parse ktlvs.

`pyhomekit.utils.prepare_tlv` (*param_type: int, value: bytes*) → `typing.Iterator[bytes]`

Formats the TLV into the expected format of the PDU.

Parameters

- **param_type** – The name or code for the HAP Parameter type
- **value** – The value in bytes of the parameter.

0.0.1.4

- Read The Docs integration
- Travis integration
- Code quality checking
- Tests

0.0.1

- Initial version

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

p

- `pyhomekit`, [3](#)
- `pyhomekit.ble`, [3](#)
- `pyhomekit.constants`, [6](#)
- `pyhomekit.utils`, [8](#)

A

Add_Pairing (pyhomekit.constants.PairingKTLVMethodValues attribute), 7

Additional_Authorization_Data (pyhomekit.constants.HapParamTypes attribute), 6

administrator_only_access() (pyhomekit.ble.HapAccessoryLock method), 3

audio_feedback() (pyhomekit.ble.HapAccessoryLock method), 3

C

Characteristic_Execute_Write (pyhomekit.constants.HapBleOpCodes attribute), 6

Characteristic_Instance_ID (pyhomekit.constants.HapParamTypes attribute), 6

Characteristic_Read (pyhomekit.constants.HapBleOpCodes attribute), 6

Characteristic_Signature_Read (pyhomekit.constants.HapBleOpCodes attribute), 6

Characteristic_Timed_Write (pyhomekit.constants.HapBleOpCodes attribute), 6

Characteristic_Type (pyhomekit.constants.HapParamTypes attribute), 7

Characteristic_Write (pyhomekit.constants.HapBleOpCodes attribute), 6

characteristic() (pyhomekit.ble.HapAccessory method), 3

cid (pyhomekit.ble.HapCharacteristic attribute), 5

connect() (pyhomekit.ble.HapAccessory method), 3

control_field (pyhomekit.ble.HapBlePduHeader attribute), 4

control_field_bits (pyhomekit.ble.HapBlePduHeader attribute), 4

current_door_state() (pyhomekit.ble.HapAccessoryLock method), 3

D

data (pyhomekit.ble.HapBlePduHeader attribute), 4

data (pyhomekit.ble.HapBlePduRequestHeader attribute), 5

data (pyhomekit.ble.HapBlePduResponseHeader attribute), 5

discover_hap_characteristics() (pyhomekit.ble.HapAccessory method), 3

F

fragment_tlvs() (in module pyhomekit.ble), 6

fragmented (pyhomekit.ble.HapBlePdu attribute), 4

from_data() (pyhomekit.ble.HapBlePduResponseHeader class method), 5

G

GATT_Presentation_Format_Descriptor (pyhomekit.constants.HapParamTypes attribute), 7

GATT_User_Description_Descriptor (pyhomekit.constants.HapParamTypes attribute), 7

GATT_Valid_Range (pyhomekit.constants.HapParamTypes attribute), 7

get_characteristic() (pyhomekit.ble.HapAccessory method), 3

H

HAP_Characteristic_Properties_Descriptor (pyhomekit.constants.HapParamTypes attribute), 7

HAP_Linked_Services (pyhomekit.constants.HapParamTypes attribute), 7

HAP_Service_Properties (pyhomekit.constants.HapParamTypes attribute), 7

HAP_Step_Value_Descriptor (pyhomekit.constants.HapParamTypes attribute), 7

HAP_Valid_Values_Descriptor	(pyhome-kit.constants.HapParamTypes attribute), 7	kTLVType_EncryptedData	(pyhome-kit.constants.PairingKTLVValues attribute), 7
HAP_Valid_Values_Range_Descriptor	(pyhome-kit.constants.HapParamTypes attribute), 7	kTLVType_Error	(pyhome-kit.constants.PairingKTLVValues attribute), 7
HapAccessory (class in pyhomekit.ble), 3		kTLVType_FragmentData	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapAccessoryLock (class in pyhomekit.ble), 3		kTLVType_FragmentLast	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBleError, 8		kTLVType_Identifier	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBleOpCodes (class in pyhomekit.constants), 6		kTLVType_Method	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBlePdu (class in pyhomekit.ble), 4		kTLVType_Permissions	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBlePduHeader (class in pyhomekit.ble), 4		kTLVType_Proof	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBlePduRequestHeader (class in pyhomekit.ble), 4		kTLVType_PublicKey	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBlePduResponseHeader (class in pyhomekit.ble), 5		kTLVType_RetryDelay	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapBleStatusCodes (class in pyhomekit.constants), 6		kTLVType_Salt	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapCharacteristic (class in pyhomekit.ble), 5		kTLVType_Separator	(pyhome-kit.constants.PairingKTLVValues attribute), 8
HapParamTypes (class in pyhomekit.constants), 6		kTLVType_Signature	(pyhome-kit.constants.PairingKTLVValues attribute), 8
I		kTLVType_State	(pyhome-kit.constants.PairingKTLVValues attribute), 8
identity() (in module pyhomekit.constants), 8		L	
Insufficient_Authentication	(pyhome-kit.constants.HapBleStatusCodes attribute), 6	List_Pairings	(pyhome-kit.constants.PairingKTLVMethodValues attribute), 7
Insufficient_Authorization	(pyhome-kit.constants.HapBleStatusCodes attribute), 6	lock_control_point()	(pyhome-kit.ble.HapAccessoryLock method), 4
Invalid_Instance_ID	(pyhome-kit.constants.HapBleStatusCodes attribute), 6	lock_current_state()	(pyhome-kit.ble.HapAccessoryLock method), 4
Invalid_Request	(pyhome-kit.constants.HapBleStatusCodes attribute), 6	lock_last_known_action()	(pyhome-kit.ble.HapAccessoryLock method), 4
iterate_tlv() (in module pyhomekit.utils), 8		lock_management_auto_security_timeout()	(pyhome-kit.ble.HapAccessoryLock method), 4
K		lock_target_state() (pyhomekit.ble.HapAccessoryLock method), 4	
kTLVError_Authenticatio	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVError_Backof	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVError_Bus	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVError_MaxPeer	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVError_MaxTrie	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVError_Unavailabl	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVError_Unknow	(pyhome-kit.constants.PairingKTLVErrorCodes attribute), 7		
kTLVType_Certificate	(pyhome-kit.constants.PairingKTLVValues attribute),		

logs() (pyhomekit.ble.HapAccessoryLock method), 4

M

max_len (pyhomekit.ble.HapBlePdu attribute), 4

Max_Procedures (pyhomekit.constants.HapBleStatusCodes attribute), 6

motion_detected() (pyhomekit.ble.HapAccessoryLock method), 4

O

Origin_local_vs_remote (pyhomekit.constants.HapParamTypes attribute), 7

P

pair() (pyhomekit.ble.HapAccessory method), 3

Pair_Setup (pyhomekit.constants.PairingKTLVMethodValues attribute), 7

Pair_Verify (pyhomekit.constants.PairingKTLVMethodValues attribute), 7

pair_verify() (pyhomekit.ble.HapAccessory method), 3

PairingKTLVErrorCodes (class in pyhomekit.constants), 7

PairingKTLVMethodValues (class in pyhomekit.constants), 7

PairingKTLVValues (class in pyhomekit.constants), 7

parse_format() (in module pyhomekit.constants), 8

parse_ktlvs() (in module pyhomekit.utils), 9

pdu_fragments() (pyhomekit.ble.HapBlePdu method), 4

prepare_tlv() (in module pyhomekit.utils), 9

pyhomekit (module), 3

pyhomekit.ble (module), 3

pyhomekit.constants (module), 6

pyhomekit.utils (module), 8

R

raw_data (pyhomekit.ble.HapBlePdu attribute), 4

read() (pyhomekit.ble.HapCharacteristic method), 5

reconnect_callback_factory() (in module pyhomekit.ble), 6

reconnect_tenacity_retry() (in module pyhomekit.ble), 6

Remove_Pairing (pyhomekit.constants.PairingKTLVMethodValues attribute), 7

Reserved (pyhomekit.constants.PairingKTLVMethodValues attribute), 7

Return_Response (pyhomekit.constants.HapParamTypes attribute), 7

S

save_key() (pyhomekit.ble.HapAccessory method), 3

Service_Instance_ID (pyhomekit.constants.HapParamTypes attribute), 7

Service_Signature_Read (pyhomekit.constants.HapBleOpCodes attribute), 6

Service_Type (pyhomekit.constants.HapParamTypes attribute), 7

signature (pyhomekit.ble.HapCharacteristic attribute), 5

Success (pyhomekit.constants.HapBleStatusCodes attribute), 6

T

to_bool() (in module pyhomekit.constants), 8

to_float() (in module pyhomekit.constants), 8

to_int32() (in module pyhomekit.constants), 8

to_uint16() (in module pyhomekit.constants), 8

to_uint32() (in module pyhomekit.constants), 8

to_uint64() (in module pyhomekit.constants), 8

to_uint8() (in module pyhomekit.constants), 8

to_utf8() (in module pyhomekit.constants), 8

to_uuid() (in module pyhomekit.constants), 8

transaction_id (pyhomekit.ble.HapBlePduRequestHeader attribute), 5

TTL (pyhomekit.constants.HapParamTypes attribute), 7

U

Unsupported_PDU (pyhomekit.constants.HapBleStatusCodes attribute), 6

V

Value (pyhomekit.constants.HapParamTypes attribute), 7

version() (pyhomekit.ble.HapAccessoryLock method), 4

W

write() (pyhomekit.ble.HapCharacteristic method), 5

write_ktlvs() (pyhomekit.ble.HapCharacteristic method), 6