
pysatl Documentation

Sebastien Riou <matic@nimp.co.uk>

Apr 18, 2020

Contents

1	Installation	2
1.1	From PyPI	2
2	Classes	3
2.1	PySatl	3
2.2	CAPDU	4
2.3	RAPDU	4
2.4	Utils	4
2.5	SocketComDriver	5
2.6	StreamComDriver	6
3	Indices and tables	7
	Index	8

`pysatl` is a package to communicate using [SATL](#) protocol.

SATL stands for ‘Simple APDU Transport Layer’. It is a simple way to exchange ISO7816-4 APDUs over interfaces not covered in ISO7816-3.

Other pages (online)

- [project page on GitHub](#)
- [Download Page](#) with releases
- This page, when viewed online is at <https://satl.readthedocs.io/en/latest/>

Installation

This installs a package that can be used from Python (`import pysatl`).

1.1 From PyPI

To install for the current user:

```
python3 -m pip install --user pysatl
```

To install for all users on the system, administrator rights (root) may be required.

```
python3 -m pip install pysatl
```

Classes

2.1 PySatl

class `pysatl.PySatl` (*is_master*, *com_driver*, *skip_init=False*)

SATL main class

Generic SATL implementation. It interface to actual hardware via a “communication driver” which shall implement few functions. See [SocketComDriver](#) and [StreamComDriver](#) for example.

Parameters

- **is_master** (*bool*) – Set to `True` to be master, `False` to be slave
- **com_driver** (*object*) – A SATL communication driver
- **skip_init** (*bool*) – If `True` the initialization phase is skipped

property `DATA_SIZE_LIMIT`

max data field length

Type `int`

property `INITIAL_BUFFER_LENGTH`

initial length of buffer for the initialization phase

Type `int`

property `LENLEN`

length in bytes of the length fields

Type `int`

property `com`

Communication hardware driver

Type `object`

property `is_master`

`True` if master, `False` if slave

Type `bool`

property `other_bufferlen`

buffer length of the other side

Type `int`

rx ()

Receive

Returns If master, a [RAPDU](#). If slave, a [CAPDU](#).

tx (*apdu*)

Transmit

Parameters `apdu` (*object*) – if master, `apdu` shall be a [CAPDU](#). If slave, a [RAPDU](#).

2.2 CAPDU

```
class pysatl.CAPDU (CLA, INS, P1, P2, DATA=bytearray(b''), LE=0)
    ISO7816-4 C-APDU
```

All parameters are read/write attributes. There is no restriction on *CLA* and *INS* values. There is no check on *DATA* length and *LE* value.

2.3 RAPDU

```
class pysatl.RAPDU (SW1, SW2, DATA=bytearray(b''))
    ISO7816-4 R-APDU
```

All parameters are read/write attributes. There is no restriction on *SW1* and *SW2* values. There is no check on *DATA* length.

2.4 Utils

```
class pysatl.Utils
    Helper class
```

```
    static ba (hexstr_or_int)
```

Extract hex numbers from a string and returns them as a bytearray. It also handles int and list of int as argument. If it cannot convert, it raises `ValueError`.

```
    static hexstr (bytes, head="", separator=' ', tail="")
```

Returns a hex string representing bytes.

Parameters

- **bytes** – a list of bytes to stringify, e.g. `[59, 22]` or a bytearray
- **head** – the string you want in front of each byte. Empty by default.
- **separator** – the string you want between each byte. One space by default.
- **tail** – the string you want after each byte. Empty by default.

```
    static int_to_ba (x, width=-1, byteorder='little')
```

```
    static int_to_bytes (x, width=-1, byteorder='little')
```

```
    static pad (buf, granularity)
```

pad the buffer if necessary (with zeroes)

```
    static padlen (l, granularity)
```

compute the length of the pad for data of length *l* to get the requested granularity

```
    static to_int (ba, byteorder='little')
```

2.5 SocketComDriver

class pysatl.SocketComDriver(*sock*, *bufferlen*=4, *granularity*=1, *sfr_granularity*=1, *ack*=True)

Parameterized model for a communication peripheral and low level rx/tx functions

Parameters

- **sock** (*socket*) – *socket* object used for communication
- **bufferlen** (*int*) – Number of bytes that can be received in a row at max rate
- **granularity** (*int*) – Smallest number of bytes that can be transported over the link
- **ack** (*bool*) – if False, *tx_ack()* and *rx_ack()* do nothing

class SocketAsStream(*sock*)

read (*length*)

write (*data*)

property **ack**

if False, *tx_ack()* and *rx_ack()* do nothing

Type bool

property **bufferlen**

Number of bytes that can be received in a row at max rate

Type int

property **granularity**

Smallest number of bytes that can be transported over the link

Type int

rx (*length*)

Receive data

Parameters **length** (*int*) – length to receive, shall be compatible with *granularity()* and smaller or equal to *bufferlen()*

Returns received data, padded with zeroes if necessary to be compatible with *sfr_granularity()*

Return type bytes

rx_ack ()

property **sfr_granularity**

Smallest number of bytes that can be accessed via the hardware on this side

Type int

property **sock**

socket object used for communication

Type socket

tx (*data*)

Transmit data

Parameters **data** (*bytes*) – bytes to transmit, shall be compatible with *sfr_granularity()* and *granularity()*

tx_ack ()

2.6 StreamComDriver

class pysatl.**StreamComDriver** (*stream*, *bufferlen*=3, *granularity*=1, *sfr_granularity*=1, *ack*=False)

Parameterized model for a communication peripheral and low level rx/tx functions

property **ack**

if False, *tx_ack()* and *rx_ack()* do nothing

Type bool

property **bufferlen**

Number of bytes that can be received in a row at max rate

Type int

property **granularity**

Smallest number of bytes that can be transported over the link

Type int

rx (*length*)

Receive data

Parameters **length** (*int*) – length to receive, shall be compatible with *granularity()* and smaller or equal to *bufferlen()*

Returns received data, padded with zeroes if necessary to be compatible with *sfr_granularity()*

Return type bytes

rx_ack ()

property **sfr_granularity**

Smallest number of bytes that can be accessed via the hardware on this side

Type int

property **sream**

stream object used for communication

Type stream

tx (*data*)

Transmit data

Parameters **data** (*bytes*) – bytes to transmit, shall be compatible with *sfr_granularity()* and *granularity()*

tx_ack ()

Indices and tables

- `genindex`
- `search`

Index

A

`ack()` (*pysatl.SocketComDriver* property), 5
`ack()` (*pysatl.StreamComDriver* property), 6

B

`ba()` (*pysatl.Utls* static method), 4
`bufferlen()` (*pysatl.SocketComDriver* property), 5
`bufferlen()` (*pysatl.StreamComDriver* property), 6

C

CAPDU (class in *pysatl*), 4
`com()` (*pysatl.PySatl* property), 3

D

`DATA_SIZE_LIMIT()` (*pysatl.PySatl* property), 3

G

`granularity()` (*pysatl.SocketComDriver* property), 5
`granularity()` (*pysatl.StreamComDriver* property), 6

H

`hexstr()` (*pysatl.Utls* static method), 4

I

`INITIAL_BUFFER_LENGTH()` (*pysatl.PySatl* property), 3
`int_to_ba()` (*pysatl.Utls* static method), 4
`int_to_bytes()` (*pysatl.Utls* static method), 4
`is_master()` (*pysatl.PySatl* property), 3

L

`LENLEN()` (*pysatl.PySatl* property), 3

O

`other_bufferlen()` (*pysatl.PySatl* property), 3

P

`pad()` (*pysatl.Utls* static method), 4
`padlen()` (*pysatl.Utls* static method), 4
PySatl (class in *pysatl*), 3

R

RAPDU (class in *pysatl*), 4
`read()` (*pysatl.SocketComDriver.SocketAsStream* method), 5
`rx()` (*pysatl.PySatl* method), 3
`rx()` (*pysatl.SocketComDriver* method), 5
`rx()` (*pysatl.StreamComDriver* method), 6
`rx_ack()` (*pysatl.SocketComDriver* method), 5
`rx_ack()` (*pysatl.StreamComDriver* method), 6

S

`sfr_granularity()` (*pysatl.SocketComDriver* property), 5
`sfr_granularity()` (*pysatl.StreamComDriver* property), 6
`sock()` (*pysatl.SocketComDriver* property), 5
SocketComDriver (class in *pysatl*), 5
SocketComDriver.SocketAsStream (class in *pysatl*), 5
`sream()` (*pysatl.StreamComDriver* property), 6
StreamComDriver (class in *pysatl*), 6

T

`to_int()` (*pysatl.Utls* static method), 4
`tx()` (*pysatl.PySatl* method), 3
`tx()` (*pysatl.SocketComDriver* method), 5
`tx()` (*pysatl.StreamComDriver* method), 6
`tx_ack()` (*pysatl.SocketComDriver* method), 5
`tx_ack()` (*pysatl.StreamComDriver* method), 6

U

Utls (class in *pysatl*), 4

W

`write()` (*pysatl.SocketComDriver.SocketAsStream* method), 5