pysatl Documentation

Sebastien Riou <matic@nimp.co.uk>

Mar 12, 2023

Contents

`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*.

property spy_frame_rx
functions called to spy on each rx frame, without padding

property spy_frame_tx
functions called to spy on each tx frame, without padding

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.

static from_bytes(*apdu_bytes*)
Create a CAPDU from its bytes representation

static from_hexstr(*hexstr*)
Convert a string of hex digits to CAPDU

to_ba()
Convert to a bytearray

to_bytes()
Convert to bytes

to_hexstr(*, *skip_long_data=False*, *separator=''*)
Convert to a string of hex digits

to_str(*, *skip_long_data=False*)
String representation

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* valuesself. There is no check on DATA length.

static from_bytes(*apdu_bytes*)
Create a RAPDU from its bytes representation

static from_hexstr(*hexstr*)
Convert a string of hex digits to RAPDU

matchDATA(*dataPat*)
Check if DATA match a given hexadecimal pattern, 'X' match anything

matchSW(*swPat*)
Check if Status Word match a given hexadecimal pattern, 'X' match anything

swBytes()
SW1 and SW2 as bytearray

to_ba()
Convert to bytearray

to_bytes()
Convert to bytes

to_hexstr(*, *skip_long_data=False*)
Convert to a string of hex digits

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=''*, ***, *skip_long_data=False*)
Returns an 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 bytes. Empty by default.
- **separator** – the string you want between each bytes. One space by default.
- **tail** – the string you want after each bytes. 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

property spy_frame_rx
functions called to spy on each rx frame

property spy_frame_tx
functions called to spy on each tx frame

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 spy_frame_rx
functions called to spy on each rx frame

property spy_frame_tx
functions called to spy on each tx frame

property stream
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()

CHAPTER 3

Indices and tables

- genindex
- search

Index