pyTeliumManager Documentation

Release 2.3.0

Ahmed TAHRI

Sep 18, 2021

Contents

CHAPTER 1

Overview

This module allow you to manipulate your Ingenico payment device such as IWL250, iCT250 for instance. Accept USB Emulated Serial Device or Native RS-232 Serial Link.

CHAPTER 2

Features

- Ask for payment in any currency.
- Verify transaction afterward and extract payment source data if needed.

Contents:

2.1 Requirements

This package is intended to be cross-platform. Unix, Linux and NT systems are supported.

2.1.1 Libs

- Python >= 2.7 or Python >= 3.4
- pySerial >= 3.3
- pyCountry >= 17.0

2.1.2 Device

In order to accept communication of any kind, configure device as follows:

1. Press “F” button.
2. Press 0 - Telium Manager
3. Press 5 - Init.
4. Press 1 - Settings
5. Select - Cashdraw/Checkout connect.
6. Select “Enable”

7. Then select your preferred interface (USB, COM1, COM2)

Finally, reboot your device.

2.2 Installation

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

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

2.2.1 From PyPI

pyTeliumManager can be installed from PyPI:

```
pip install pyTeliumManager
```

2.2.2 From git via dev-master

You can install from dev-master branch using git:

```
git clone https://github.com/Ousret/pyTeliumManager.git
cd pyTeliumManager/
python setup.py install
```

2.3 Classes

2.3.1 Transaction details

class TeliumAsk

__init__(*pos_number*, *answer_flag*, *transaction_type*, *payment_mode*, *currency_numeric*, *delay*, *authorization*, *amount*)

Parameters

- **pos_number** (*str*) – Checkout unique identifier from '01' to '99'.
- **answer_flag** (*str*) – Answer report size. use *TERMINAL_ANSWER_SET_FULLSIZED* for complete details or *TERMINAL_ANSWER_SET_SMALLSIZED* for limited answer report. Limited report does not show payment source id, e.g. credit card numbers.
- **transaction_type** (*str*) – If transaction is about CREDIT, DEBIT, etc.. . Use at least one of listed possible values: *TERMINAL_MODE_PAYMENT_DEBIT*, *TERMINAL_MODE_PAYMENT_CREDIT*, *TERMINAL_MODE_PAYMENT_REFUND*, *TERMINAL_MODE_PAYMENT_AUTO*.
- **payment_mode** (*str*) – Type of payment support. Use at least one of listed possible values: *TERMINAL_TYPE_PAYMENT_CARD*, *TERMINAL_TYPE_PAYMENT_CHECK*, *TERMINAL_TYPE_PAYMENT_AMEX*, *TERMINAL_TYPE_PAYMENT_CETELEM*, *TERMINAL_TYPE_PAYMENT_COFINOGA*, *TERMINAL_TYPE_PAYMENT_DINERS*, *TERMINAL_TYPE_PAYMENT_FRANFINANCE*, *TERMINAL_TYPE_PAYMENT_JCB*,

TERMINAL_TYPE_PAYMENT_ACCORD_FINANCE, *TERMINAL_TYPE_PAYMENT_MONEO*, *TERMINAL_TYPE_PAYMENT_CUP*, *TERMINAL_TYPE_PAYMENT_FINTRAX_EMV*, *TERMINAL_TYPE_PAYMENT_OTHER*.

- **currency_numeric** (*str*) – Currency ISO format. Two ISO currency are available as constant. TERMINAL_NUMERIC_CURRENCY_EUR: EUR - € - ISO;978. TERMINAL_NUMERIC_CURRENCY_USD: USD - $ - ISO;840.
- **delay** (*str*) – Describe if answer should be immediate (without valid status) or after transaction. Use at least one of listed possible values: *TERMINAL_REQUEST_ANSWER_WAIT_FOR_TRANSACTION*, *TERMINAL_REQUEST_ANSWER_INSTANT*.
- **authorization** (*str*) – Describe if the terminal has to manually authorize payment.

  Use at least one of listed possible values: *TERMINAL_FORCE_AUTHORIZATION_ENABLE*, *TERMINAL_FORCE_AUTHORIZATION_DISABLE*.
- **amount** (*float*) – Payment amount, min 0.01, max 99999.99.

This object is meant to be translated into a bytes sequence and transferred to your terminal.

encode()

Returns Raw string array with payment information

Return type str

Raises ***SequenceDoesNotMatchLengthException*** – Will be raised if the string sequence doesn't match required length. Check your instance params.

Translate object into a string sequence ready to be sent to device.

static decode(*data*)

Parameters **data** (*bytes*) – Raw bytes sequence to be converted into TeliumAsk instance.

Returns Create a new TeliumAsk.

Return type *TeliumAsk*

Raises

- ***LrcChecksumException*** – Will be raised if LRC checksum doesn't match.
- ***SequenceDoesNotMatchLengthException*** – Will be raised if the string sequence doesn't match required length.

Create a new instance of TeliumAsk from a bytes sequence previously generated with encode(). This is no use in a production environment.

static new_payment(*amount*, *payment_mode='debit'*, *target_currency='USD'*, *checkout_unique_id='1'*, *wait_for_transaction_to_end=True*, *collect_payment_source_info=True*, *force_bank_verification=False*)

Parameters

- **amount** (*float*) – Amount requested
- **payment_mode** (*str*) – Specify transaction type. (debit, credit or refund)
- **target_currency** (*str*) – Target currency, must be written in letters. (EUR, USD, etc..)
- **checkout_unique_id** (*str*) – Unique checkout identifer.

- **wait_for_transaction_to_end** (*bool*) – Set to True if you need valid transaction status otherwise, set it to False.
- **collect_payment_source_info** (*bool*) – If you want to retrieve specifics data about payment source identification.
- **force_bank_verification** (*bool*) – Set it to True if your business need to enforce payment verification.

Returns Ready to use TeliumAsk instance

Return type *TeliumAsk*

Create new TeliumAsk in order to prepare payment. Most commonly used.

2.3.2 Transaction results

class TeliumResponse

__init__(*pos_number*, *transaction_result*, *amount*, *payment_mode*, *report*, *currency_numeric*, *private*)

Parameters

- **pos_number** (*str*) – Checkout unique identifier from '01' to '99'.
- **transaction_result** (*int*) – Transaction result.
- **amount** (*float*) – Payment authorized/acquired amount.
- **payment_mode** (*str*) – Type of payment support.
- **report** (*str*) – Contains payment source unique identifier like credit-card numbers when fullsized report is enabled.
- **currency_numeric** (*str*) – Currency ISO format.
- **private** (*str*) – If supported by your device, contains transaction unique identifier.

has_succeeded

Getter True if transaction has been authorized, False otherwise.

Type bool

report

Getter Contain data like the card numbers for instance. Should be handled wisely.

Type str

transaction_id

Getter If supported by your device, contains transaction unique identifier.

Type bool

card_id

Getter Read card numbers if available.

Type str|None

card_id_sha512

Getter Return payment source id hash repr (sha512)

Type str|None

card_type

Getter Return if available payment card type

Type payment_card_identifier.PaymentCard|None

2.3.3 Device management

class Telium

__init__(*path='/dev/ttyACM0', baudrate=9600, timeout=1, open_on_create=True, debugging=False*)

Parameters

- **path** – Device path.
- **baudrate** (*int*) – Baud rate such as 9600 or 115200 etc. Constructor do recommend to set it as 9600.
- **timeout** (*float*) – Set a read timeout value.
- **open_on_create** (*bool*) – Specify if device should be immedialty opened on instance creation.
- **debugging** (*bool*) – Set it to True if you want to diagnose your device. Will print to stdout bunch of useful data.

The port is immediately opened on object creation if open_on_create toggle is True.

path is the device path: depending on operating system. e.g. `/dev/ttyACM0` on GNU/Linux or `COM3` on Windows. Please be aware that a proper driver is needed on Windows in order to create an emulated serial device.

Possible values for the parameter *timeout* which controls the behavior of the device instance:

- `timeout = None`: wait forever / until requested number of bytes are received, not recommended.
- `timeout = 0`: non-blocking mode, return immediately in any case, returning zero or more, up to the requested number of bytes, use it only when your computer is really fast unless you don't care about reliability.
- `timeout = x`: set timeout to `x` seconds (float allowed) returns immediately when the requested number of bytes are available, otherwise wait until the timeout expires and return all bytes that were received until then.

static get()

Returns Fresh new Telium instance or None

Return type Telium|None

Auto-create a new instance of Telium. The device path will be inferred based on most common location. This won't be reliable if you have more than one emulated serial device plugged-in. Does not work on NT platform.

ask(*telium_ask*)

Parameters **telium_ask** (TeliumAsk) – Payment details

Returns True if device has accepted it, False otherwise.

Return type bool

Initialize payment to terminal

verify(*telium_ask*)

Parameters **telium_ask** (TeliumAsk) – Payment details previously used on ask()

Returns Transaction results as TeliumResponse, None if nothing was caught from device.

Return type TeliumResponse|None

Wait for answer and convert it to TeliumResponse.

close()

Returns True if device was previously opened and now closed. False otherwise.

Return type bool

Close device if currently opened. Recommended practice, don't let Python close it from garbage collector.

timeout

Getter Current timeout set on read.

Type float

2.3.4 Native serial proxy class

Use this class instead of Telium if you're using native serial conn, see examples.

class TeliumNativeSerial

2.4 Exceptions

exception SignalDoesNotExistException
Trying to send a unknown signal to device.

exception DataFormatUnsupportedException
Exception raised when trying to send something other than a string sequence to device.

exception TerminalInitializationFailedException
Exception raised when your device doesn't respond with 'ACK' signal when receiving 'ENQ' signal. Could mean that the device is busy or not well configured.

exception TerminalUnrecognizedConstantException
Exception raised when you've built a TeliumAsk instance without proposed constant from package.

exception LrcChecksumException
Exception raised when your raw bytes sequence does not match computed LRC with actual one from the sequence. Could mean that your serial/usb conn isn't stable.

exception SequenceDoesNotMatchLengthException
Exception raised when trying to translate object via encode() or decode() doesn't match required output length. Could mean that your device is currently unsupported.

exception IllegalAmountException
Exception raised when asking for an amount is bellow TERMINAL_MINIMAL_AMOUNT_REQUESTABLE and higher than TERMINAL_MAXIMAL_AMOUNT_REQUESTABLE.

2.5 Constants

Answer flag

Fullsized report contains payment unique identifier like credit-card numbers, smallsized does not.

TERMINAL_ANSWER_SET_FULLSIZED

TERMINAL_ANSWER_SET_SMALLSIZED

Transaction type

TERMINAL_MODE_PAYMENT_DEBIT

TERMINAL_MODE_PAYMENT_CREDIT

TERMINAL_MODE_PAYMENT_REFUND

TERMINAL_MODE_PAYMENT_AUTO

Payment mode

TERMINAL_TYPE_PAYMENT_CARD

TERMINAL_TYPE_PAYMENT_CHECK

TERMINAL_TYPE_PAYMENT_AMEX

TERMINAL_TYPE_PAYMENT_CETELEM

TERMINAL_TYPE_PAYMENT_COFINOGA

TERMINAL_TYPE_PAYMENT_DINERS

TERMINAL_TYPE_PAYMENT_FRANFINANCE

TERMINAL_TYPE_PAYMENT_JCB

TERMINAL_TYPE_PAYMENT_ACCORD_FINANCE

TERMINAL_TYPE_PAYMENT_MONEO

TERMINAL_TYPE_PAYMENT_CUP

TERMINAL_TYPE_PAYMENT_FINTRAX_EMV

TERMINAL_TYPE_PAYMENT_OTHER

Delay

Instant answer won't contain a valid transaction status.

TERMINAL_REQUEST_ANSWER_WAIT_FOR_TRANSACTION

TERMINAL_REQUEST_ANSWER_INSTANT

Authorization

Forced authorization control isn't recommended because it could be significantly slower. You might have some ext. fees when using GPRS based payment device.

TERMINAL_FORCE_AUTHORIZATION_ENABLE

TERMINAL_FORCE_AUTHORIZATION_DISABLE

2.6 Example

2.6.1 Most basic usage

Example of usage:

```
# Open device
my_device = Telium('/dev/ttyACM0')

# Construct our payment infos
my_payment = TeliumAsk(
    '1',  # Checkout ID 1
    TERMINAL_ANSWER_SET_FULLSIZED,  # Ask for fullsized report
    TERMINAL_MODE_PAYMENT_DEBIT,  # Ask for debit
    TERMINAL_TYPE_PAYMENT_CARD,  # Using a card
    TERMINAL_NUMERIC_CURRENCY_EUR,  # Set currency to EUR
    TERMINAL_REQUEST_ANSWER_WAIT_FOR_TRANSACTION,  # Wait for transaction to end␣
↪before getting final answer
    TERMINAL_FORCE_AUTHORIZATION_DISABLE,  # Let device choose if we should ask for␣
↪authorization
    12.5  # Ask for 12.5 EUR
)

# Send payment infos to device
my_device.ask(my_payment)

# Wait for terminal to answer
my_answer = my_device.verify(my_payment)

if my_answer is not None:
    # Print answered data from terminal
    print(my_answer.__dict__)
```

2.6.2 Create TeliumAsk instance from static method

Create instance:

```
my_payment = TeliumAsk.new_payment(
    12.5,  # Amount you want
    payment_mode='debit',  # other mode: credit or refund.
    target_currency='EUR',
    wait_for_transaction_to_end=True,  # If you need valid transaction status
    collect_payment_source_info=True,  # If you need to identify payment source
    force_bank_verification=False   # Set it to True if you absolutly need more␣
↪guarantee in this transaction. Could result in slower authorization from bank.
)
```

2.6.3 Use Ingenico payment device thought not emulated serial link

Init:

```
# It's as easy as this
my_device = TeliumNativeSerial('/dev/ttyS4')
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

V