Termineter Documentation

Release 0.1.0

Spencer McIntyre

February 12, 2016

Termineter is a framework written in python to provide a platform for the security testing of smart meters. It implements the C12.18 and C12.19 protocols for communication. Currently supported are Meters using C12.19 with 7-bit character sets. Termineter communicates with Smart Meters via a connection using an ANSI type-2 optical probe with a serial interface. The source code is available on the GitHub homepage.

CHAPTER 1

c1218

1.1 c1218.urlhandler

1.1.1 c1218.urlhandler.protocol_unix

Classes

1.2 c1218.connection

1.2.1 Classes

class c1218.connection.**Connection**(**args*, ***kwargs*)

__init__(**args*, ***kwargs*)
This is a C12.18 driver for serial connections. It relies on PySerial to communicate with an ANSI Type-2 Optical probe to communicate with a device (presumably a smart meter).

Parameters

- **device** (*str*) – A connection string to be passed to the PySerial library. If PySerial is new enough, the serial_for_url function will be used to allow the user to use a rfc2217 bridge.
- **c1218_settings** (*dict*) – A settings dictionary to configure the C1218 parameters of 'nbrpkts' and 'pktsize' If not provided the default settings of 2 (nbrpkts) and 512 (pktsize) will be used.
- **serial_settings** (*dict*) – A PySerial settings dictionary to be applied to the serial connection instance.
- **toggle_control** (*bool*) – Enables or disables automatically settings the toggle bit in C12.18 frames.
- **enable_cache** (*bool*) – Cache specific, read only tables in memory, the first time the table is read it will be stored for retreival on subsequent requests. This is enabled only for specific tables (currently only 0 and 1).

flush_table_cache()

get_table_data(*tableid*, *octetcount=None*, *offset=None*)
Read data from a table. If successful, all of the data from the requested table will be returned.

Parameters

- **`tableid`** (*`int`*) – The table number to read from (0x0000 <= tableid <= 0xffff)
- **`octetcount`** (*`int`*) – Limit the amount of data read, only works if the meter supports this type of reading.
- **`offset`** (*`int`*) – The offset at which to start to read the data from.

`login`(*username=u'0000'*, *userid=0*, *password=None*)
Log into the connected device.

Parameters

- **`username`** (*`str`*) – the username to log in with (len(username) <= 10)
- **`userid`** (*`int`*) – the userid to log in with (0x0000 <= userid <= 0xffff)
- **`password`** (*`str`*) – password to log in with (len(password) <= 20)

Return type bool

`logoff`()
Send a logoff request.

Return type bool

`run_procedure`(*process_number*, *std_vs_mfg*, *params=u''*)
Initiate a C1219 procedure, the request is written to table 7 and the response is read from table 8.

Parameters

- **`process_number`** (*`int`*) – The numeric procedure identifier (0 <= process_number <= 2047).
- **`std_vs_mfg`** (*`bool`*) – Whether the procedure is manufacturer specified or not. True is manufacturer specified.
- **`params`** (*`str`*) – The parameters to pass to the procedure initiation request.

`set_table_cache_policy`(*cache_policy*)

`set_table_data`(*tableid*, *data*, *offset=None*)
Write data to a table.

Parameters

- **`tableid`** (*`int`*) – The table number to write to (0x0000 <= tableid <= 0xffff)
- **`data`** (*`str`*) – The data to write into the table.
- **`offset`** (*`int`*) – The offset at which to start to write the data (0x000000 <= octetcount <= 0xffffff).

`start`()
Send an identity request and then a negotiation request.

`stop`(*force=False*)
Send a terminate request.

Parameters **`force`** (*`bool`*) – ignore the remote devices response

class `c1218.connection.`**`ConnectionBase`**(*device*, *c1218_settings={}*, *serial_settings=None*, *toggle_control=True*, ***kwargs*)

`__init__`(*device*, *c1218_settings={}*, *serial_settings=None*, *toggle_control=True*, ***kwargs*)
This is a C12.18 driver for serial connections. It relies on PySerial to communicate with an ANSI Type-2 Optical probe to communicate with a device (presumably a smart meter).

Parameters

- **`device`** (*`str`*) – A connection string to be passed to the PySerial library. If PySerial is new enough, the serial_for_url function will be used to allow the user to use a rfc2217 bridge.
- **`c1218_settings`** (*`dict`*) – A settings dictionary to configure the C1218 parameters of 'nbrpkts' and 'pktsize' If not provided the default settings of 2 (nbrpkts) and 512 (pktsize) will be used.
- **`serial_settings`** (*`dict`*) – A PySerial settings dictionary to be applied to the serial connection instance.
- **`toggle_control`** (*`bool`*) – Enables or diables automatically settings the toggle bit in C12.18 frames.

`close`()
Send a terminate request and then disconnect from the serial device.

`read`(*size*)
Read raw data from the serial connection. This function is not meant to be called directly.

Parameters **`size`** (*`int`*) – The number of bytes to read from the serial connection.

`recv`(*full_frame=False*)
Receive a C1218Packet, the payload data is returned.

Parameters **`full_frame`** (*`bool`*) – If set to True, the entire C1218 frame is returned instead of just the payload.

`send`(*data*)
This sends a raw C12.18 frame and waits checks for an ACK response. In the event that a NACK is received, this function will attempt to resend the frame up to 3 times.

Parameters **`data`** (str, *`C1218Packet`*) – the data to be transmitted

`write`(*data*)
Write raw data to the serial connection. The CRC must already be included at the end. This function is not meant to be called directly.

Parameters **`data`** (*`str`*) – The raw data to write to the serial connection.

1.3 `c1218.data`

1.3.1 Classes

class `c1218.data.`**`C1218Request`**

`build`()

`name`

static **`parse`**(*data*)

class `c1218.data.`**`C1218LogonRequest`**(*username=u''*, *userid=0*)

__init__(*username=u'', userid=0*)

build()

logon = 'P'

static **parse**(*data*)

set_userid(*userid*)

set_username(*value*)

userid

username

class c1218.data.**C1218SecurityRequest**(*password=u''*)

__init__(*password=u''*)

build()

static **parse**(*data*)

password

security = 'Q'

set_password(*value*)

class c1218.data.**C1218LogoffRequest**

build()

logoff = 'R'

static **parse**(*data*)

class c1218.data.**C1218NegotiateRequest**(*pktsize, nbrpkt, baudrate=None*)

__init__(*pktsize, nbrpkt, baudrate=None*)

build()

negotiate = ''

static **parse**(*data*)

set_baudrate(*baudrate*)

set_nbrpkt(*nbrpkt*)

set_pktsize(*pktsize*)

class c1218.data.**C1218WaitRequest**(*time=1*)

__init__(*time=1*)

build()

static **parse**(*data*)

set_time(*time*)

wait = 'p'

class c1218.data.**C1218IdentRequest**

> **build**()
>
> **ident** = ' '
>
> static **parse**(*data*)

class c1218.data.**C1218TerminateRequest**

> **build**()
>
> static **parse**(*data*)
>
> **terminate** = '!'

class c1218.data.**C1218ReadRequest**(*tableid*, *offset=None*, *octetcount=None*)

> **__init__**(*tableid*, *offset=None*, *octetcount=None*)
>
> **build**()
>
> **octetcount**
>
> **offset**
>
> static **parse**(*data*)
>
> **read** = '0'
>
> **set_octetcount**(*octetcount*)
>
> **set_offset**(*offset*)
>
> **set_tableid**(*tableid*)
>
> **tableid**

class c1218.data.**C1218WriteRequest**(*tableid*, *data*, *offset=None*)

> **__init__**(*tableid*, *data*, *offset=None*)
>
> **build**()
>
> **data**
>
> **offset**
>
> static **parse**(*data*)
>
> **set_data**(*data*)
>
> **set_offset**(*offset*)
>
> **set_tableid**(*tableid*)
>
> **tableid**
>
> **write** = '@'

class c1218.data.**C1218Packet**(*data=None*, *control=None*, *length=None*)

> **__init__**(*data=None*, *control=None*, *length=None*)
>
> **build**()

control = '\x00'

data

identity = '\x00'

static **parse**(*data*)

sequence = '\x00'

set_control(*control*)

set_data(*data*)

set_length(*length*)

start = '\xee'

1.4 c1218.errors

1.4.1 Exceptions

exception c1218.errors.**C1218Error**(*msg*, *code=None*)
This is a generic C1218 Error.

exception c1218.errors.**C1218IOError**(*msg*)
Raised when there is a problem sending or receiving data.

exception c1218.errors.**C1218NegotiateError**(*msg*, *code=None*)
Raised in response to an invalid reply to a Negotiate request.

exception c1218.errors.**C1218ReadTableError**(*msg*, *code=None*)
Raised when a table is not successfully read.

Parameters **errcode** (*int*) – The error that was returned while reading the table.

exception c1218.errors.**C1218WriteTableError**(*msg*, *code=None*)
Raised when a table is not successfully written to.

Parameters **errcode** (*int*) – The error that was returned while writing to the table.

c1219

2.1 c1219.access

2.1.1 c1219.access.general

Classes

class `c1219.access.general.`**`C1219GeneralAccess`**(*conn*)

This class provides generic access to the general configuration tables that are stored in the decade 0x tables.

`__init__`(*conn*)

Initializes a new instance of the class and reads tables from the corresponding decades to populate information.

@type conn: c1218.connection.Connection @param conn: The driver to be used for interacting with the necessary tables.

`char_format`

`device_id`

`ed_mode`

`ed_model`

`encoding`

`fw_revision_no`

`fw_version_no`

`hw_revision_no`

`hw_version_no`

`id_form`

`manufacturer`

`mfg_proc_used`

`mfg_serial_no`

`mfg_tbls_used`

`nameplate_type`

`set_device_id`(*newid*)

std_proc_used

std_revision_no

std_status

std_tbls_used

std_version_no

2.1.2 `c1219.access.log`

Classes

class `c1219.access.log.`**`C1219LogAccess`**(*conn*)

This class provides generic access to the log data tables that are stored in the decade 7x tables.

`__init__`(*conn*)

Initializes a new instance of the class and reads tables from the corresponding decades to populate information.

@type conn: c1218.connection.Connection @param conn: The driver to be used for interacting with the necessary tables.

logs

nbr_event_entries

nbr_history_entries

2.1.3 `c1219.access.security`

Classes

class `c1219.access.security.`**`C1219SecurityAccess`**(*conn*)

This class provides generic access to the security configuration tables that are stored in the decade 4x tables.

`__init__`(*conn*)

Initializes a new instance of the class and reads tables from the corresponding decades to populate information.

@type conn: c1218.connection.Connection @param conn: The driver to be used for interacting with the necessary tables.

key_len

keys

nbr_keys

nbr_passwords

nbr_perm_used

password_len

passwords

procedure_permissions

table_permissions

2.1.4 `c1219.access.telephone`

Classes

class `c1219.access.telephone.`**`C1219TelephoneAccess`**(*conn*)

This class provides generic access to the telephone/modem configuration tables that are stored in the decade 9x tables.

`__init__`(*conn*)

Initializes a new instance of the class and reads tables from the corresponding decades to populate information.

@type conn: c1218.connection.Connection @param conn: The driver to be used for interacting with the necessary tables.

`answer_bit_rate`

`can_answer`

`dial_delay`

`global_bit_rate`

`initiate_call`(*number=None*, *idx=None*)

static **`initiate_call_ex`**(*conn*, *idx*)

`nbr_originate_numbers`

`originate_bit_rate`

`originating_numbers`

`prefix_number`

`primary_phone_number_idx`

`psem_identity`

`secondary_phone_number_idx`

`update_last_call_statuses`()

`use_extended_status`

2.2 `c1219.data`

2.2.1 Classes

class `c1219.data.`**`C1219ProcedureInit`**(*endianess*, *table_proc_nbr*, *std_vs_mfg*, *selector*, *seqnum*, *params=''*)

A C1219 Procedure Request, this data is written to table 7 in order to start a procedure.

Parameters

- **`endianess`** (*`str`*) – The endianess to use when packing values ('>' or '<')
- **`table_proc_nbr`** (*`int`*) – The numeric procedure identifier (0 <= table_proc_nbr <= 2047).
- **`std_vs_mfg`** (*`bool`*) – Whether the procedure is manufacturer specified or not. True is manufacturer specified.

- **selector** (*int*) – Controls how data is returned (0 <= selector <= 15). 0: Post response in PROC_RESPONSE_TBL (#8) on completion. 1: Post response in PROC_RESPONSE_TBL (#8) on exception. 2: Do not post response in PROC_RESPONSE_TBL (#8). 3: Post response in PROC_RESPONSE_TBL (#8) immediately and another response in PROC_RESPONSE_TBL (#8) on completion. 4-15: Reserved.
- **seqnum** (*int*) – The identifier for this procedure to be used for coordination (0x00 <= seqnum <= 0xff).
- **params** (*str*) – The parameters to pass to the procedure initiation request.

__init__(*endianess*, *table_proc_nbr*, *std_vs_mfg*, *selector*, *seqnum*, *params=''*)

build()

static parse(*endianess*, *data*)

2.3 c1219.errors

2.3.1 Exceptions

exception c1219.errors.**C1219ProcedureError**(*msg*)
Raised when a procedure can not be executed.

exception c1219.errors.**C1219ParseError**(*msg*, *tableid=None*)
Raised when there is an error parsing data.

Parameters **tableid** (*int*) – If the data originated from a table, the faulty table can be specified here.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index