
Termineter Documentation

Release 0.1.0

Spencer McIntyre

February 12, 2016

1	c1218	3
2	c1219	9
3	Indices and tables	13
	Python Module Index	15

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](#).

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 retrieval 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 disables 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*=‘’, *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.

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 ($0 \leq \text{selector} \leq 15$). 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 ($0x00 \leq \text{seqnum} \leq 0xff$).
- **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.

Indices and tables

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

C

- c1218, 3
- c1218.connection, 3
- c1218.data, 5
- c1218.errors, 8
- c1218.urlhandler, 3
- c1218.urlhandler.protocol_unix, 3
- c1219, 9
- c1219.access, 9
- c1219.access.general, 9
- c1219.access.log, 10
- c1219.access.security, 10
- c1219.access.telephone, 11
- c1219.data, 11
- c1219.errors, 12

Symbols

- `__init__()` (c1218.connection.Connection method), 3
 - `__init__()` (c1218.connection.ConnectionBase method), 4
 - `__init__()` (c1218.data.C1218LogonRequest method), 5
 - `__init__()` (c1218.data.C1218NegotiateRequest method), 6
 - `__init__()` (c1218.data.C1218Packet method), 7
 - `__init__()` (c1218.data.C1218ReadRequest method), 7
 - `__init__()` (c1218.data.C1218SecurityRequest method), 6
 - `__init__()` (c1218.data.C1218WaitRequest method), 6
 - `__init__()` (c1218.data.C1218WriteRequest method), 7
 - `__init__()` (c1219.access.general.C1219GeneralAccess method), 9
 - `__init__()` (c1219.access.log.C1219LogAccess method), 10
 - `__init__()` (c1219.access.security.C1219SecurityAccess method), 10
 - `__init__()` (c1219.access.telephone.C1219TelephoneAccess method), 11
 - `__init__()` (c1219.data.C1219ProcedureInit method), 12
- A**
- `answer_bit_rate` (c1219.access.telephone.C1219TelephoneAccess attribute), 11
- B**
- `build()` (c1218.data.C1218IdentRequest method), 7
 - `build()` (c1218.data.C1218LogoffRequest method), 6
 - `build()` (c1218.data.C1218LogonRequest method), 6
 - `build()` (c1218.data.C1218NegotiateRequest method), 6
 - `build()` (c1218.data.C1218Packet method), 7
 - `build()` (c1218.data.C1218ReadRequest method), 7
 - `build()` (c1218.data.C1218Request method), 5
 - `build()` (c1218.data.C1218SecurityRequest method), 6
 - `build()` (c1218.data.C1218TerminateRequest method), 7
 - `build()` (c1218.data.C1218WaitRequest method), 6
 - `build()` (c1218.data.C1218WriteRequest method), 7
 - `build()` (c1219.data.C1219ProcedureInit method), 12
- C**
- c1218 (module), 3
 - c1218.connection (module), 3
 - c1218.data (module), 5
 - c1218.errors (module), 8
 - c1218.urlhandler (module), 3
 - c1218.urlhandler.protocol_unix (module), 3
 - C1218Error, 8
 - C1218IdentRequest (class in c1218.data), 6
 - C1218IOError, 8
 - C1218LogoffRequest (class in c1218.data), 6
 - C1218LogonRequest (class in c1218.data), 5
 - C1218NegotiateError, 8
 - C1218NegotiateRequest (class in c1218.data), 6
 - C1218Packet (class in c1218.data), 7
 - C1218ReadRequest (class in c1218.data), 7
 - C1218ReadTableError, 8
 - C1218Request (class in c1218.data), 5
 - C1218SecurityRequest (class in c1218.data), 6
 - C1218TerminateRequest (class in c1218.data), 7
 - C1218WaitRequest (class in c1218.data), 6
 - C1218WriteRequest (class in c1218.data), 7
 - C1218WriteTableError, 8
 - c1219 (module), 9
 - c1219.access (module), 9
 - c1219.access.general (module), 9
 - c1219.access.log (module), 10
 - c1219.access.security (module), 10
 - c1219.access.telephone (module), 11
 - c1219.data (module), 11
 - c1219.errors (module), 12
 - C1219GeneralAccess (class in c1219.access.general), 9
 - C1219LogAccess (class in c1219.access.log), 10
 - C1219ParseError, 12
 - C1219ProcedureError, 12
 - C1219ProcedureInit (class in c1219.data), 11
 - C1219SecurityAccess (class in c1219.access.security), 10
 - C1219TelephoneAccess (class in c1219.access.telephone), 11
 - `can_answer` (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 - `char_format` (c1219.access.general.C1219GeneralAccess attribute), 9

close() (c1218.connection.ConnectionBase method), 5
 Connection (class in c1218.connection), 3
 ConnectionBase (class in c1218.connection), 4
 control (c1218.data.C1218Packet attribute), 7

D

data (c1218.data.C1218Packet attribute), 8
 data (c1218.data.C1218WriteRequest attribute), 7
 device_id (c1219.access.general.C1219GeneralAccess attribute), 9
 dial_delay (c1219.access.telephone.C1219TelephoneAccess attribute), 11

E

ed_mode (c1219.access.general.C1219GeneralAccess attribute), 9
 ed_model (c1219.access.general.C1219GeneralAccess attribute), 9
 encoding (c1219.access.general.C1219GeneralAccess attribute), 9

F

flush_table_cache() (c1218.connection.Connection method), 3
 fw_revision_no (c1219.access.general.C1219GeneralAccess attribute), 9
 fw_version_no (c1219.access.general.C1219GeneralAccess attribute), 9

G

get_table_data() (c1218.connection.Connection method), 3
 global_bit_rate (c1219.access.telephone.C1219TelephoneAccess attribute), 11

H

hw_revision_no (c1219.access.general.C1219GeneralAccess attribute), 9
 hw_version_no (c1219.access.general.C1219GeneralAccess attribute), 9

I

id_form (c1219.access.general.C1219GeneralAccess attribute), 9
 ident (c1218.data.C1218IdentRequest attribute), 7
 identity (c1218.data.C1218Packet attribute), 8
 initiate_call() (c1219.access.telephone.C1219TelephoneAccess method), 11
 initiate_call_ex() (c1219.access.telephone.C1219TelephoneAccess static method), 11

K

key_len (c1219.access.security.C1219SecurityAccess attribute), 10

keys (c1219.access.security.C1219SecurityAccess attribute), 10

L

login() (c1218.connection.Connection method), 4
 logoff (c1218.data.C1218LogoffRequest attribute), 6
 logoff() (c1218.connection.Connection method), 4
 logon (c1218.data.C1218LogonRequest attribute), 6
 logs (c1219.access.log.C1219LogAccess attribute), 10

M

manufacturer (c1219.access.general.C1219GeneralAccess attribute), 9
 mfg_proc_used (c1219.access.general.C1219GeneralAccess attribute), 9
 mfg_serial_no (c1219.access.general.C1219GeneralAccess attribute), 9
 mfg_tbls_used (c1219.access.general.C1219GeneralAccess attribute), 9

N

name (c1218.data.C1218Request attribute), 5
 nameplate_type (c1219.access.general.C1219GeneralAccess attribute), 9
 nbr_event_entries (c1219.access.log.C1219LogAccess attribute), 10
 nbr_history_entries (c1219.access.log.C1219LogAccess attribute), 10
 nbr_keys (c1219.access.security.C1219SecurityAccess attribute), 10
 nbr_originate_numbers (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 ntp_passwords (c1219.access.security.C1219SecurityAccess attribute), 10
 nbr_perm_used (c1219.access.security.C1219SecurityAccess attribute), 10
 negotiate (c1218.data.C1218NegotiateRequest attribute), 6

O

octetcount (c1218.data.C1218ReadRequest attribute), 7
 offset (c1218.data.C1218ReadRequest attribute), 7
 offset (c1218.data.C1218WriteRequest attribute), 7
 originate_bit_rate (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 originating_numbers (c1219.access.telephone.C1219TelephoneAccess attribute), 11

P

parse() (c1218.data.C1218IdentRequest static method), 7
 parse() (c1218.data.C1218LogoffRequest static method), 6
 parse() (c1218.data.C1218LogonRequest static method), 6

parse() (c1218.data.C1218NegotiateRequest static method), 6
 parse() (c1218.data.C1218Packet static method), 8
 parse() (c1218.data.C1218ReadRequest static method), 7
 parse() (c1218.data.C1218Request static method), 5
 parse() (c1218.data.C1218SecurityRequest static method), 6
 parse() (c1218.data.C1218TerminateRequest static method), 7
 parse() (c1218.data.C1218WaitRequest static method), 6
 parse() (c1218.data.C1218WriteRequest static method), 7
 parse() (c1219.data.C1219ProcedureInit static method), 12
 password (c1218.data.C1218SecurityRequest attribute), 6
 password_len (c1219.access.security.C1219SecurityAccess attribute), 10
 passwords (c1219.access.security.C1219SecurityAccess attribute), 10
 prefix_number (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 primary_phone_number_idx (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 procedure_permissions (c1219.access.security.C1219SecurityAccess attribute), 10
 psem_identity (c1219.access.telephone.C1219TelephoneAccess attribute), 11

R

read (c1218.data.C1218ReadRequest attribute), 7
 read() (c1218.connection.ConnectionBase method), 5
 recv() (c1218.connection.ConnectionBase method), 5
 run_procedure() (c1218.connection.Connection method), 4

S

secondary_phone_number_idx (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 security (c1218.data.C1218SecurityRequest attribute), 6
 send() (c1218.connection.ConnectionBase method), 5
 sequence (c1218.data.C1218Packet attribute), 8
 set_baudrate() (c1218.data.C1218NegotiateRequest method), 6
 set_control() (c1218.data.C1218Packet method), 8
 set_data() (c1218.data.C1218Packet method), 8
 set_data() (c1218.data.C1218WriteRequest method), 7
 set_device_id() (c1219.access.general.C1219GeneralAccess method), 9
 set_length() (c1218.data.C1218Packet method), 8
 set_nrpkt() (c1218.data.C1218NegotiateRequest method), 6
 set_octetcount() (c1218.data.C1218ReadRequest method), 7
 set_offset() (c1218.data.C1218ReadRequest method), 7
 set_offset() (c1218.data.C1218WriteRequest method), 7
 set_password() (c1218.data.C1218SecurityRequest method), 6
 set_pktsize() (c1218.data.C1218NegotiateRequest method), 6
 set_table_cache_policy() (c1218.connection.Connection method), 4
 set_table_data() (c1218.connection.Connection method), 4
 set_tableid() (c1218.data.C1218ReadRequest method), 7
 set_tableid() (c1218.data.C1218WriteRequest method), 7
 set_time() (c1218.data.C1218WaitRequest method), 6
 set_userid() (c1218.data.C1218LogonRequest method), 6
 set_username() (c1218.data.C1218LogonRequest method), 6
 start (c1218.data.C1218Packet attribute), 8
 start() (c1218.connection.Connection method), 4
 stats_proc_used (c1219.access.general.C1219GeneralAccess attribute), 9
 std_revision_no (c1219.access.general.C1219GeneralAccess attribute), 10
 std_status (c1219.access.general.C1219GeneralAccess attribute), 10
 std_tbls_used (c1219.access.general.C1219GeneralAccess attribute), 10
 std_version_no (c1219.access.general.C1219GeneralAccess attribute), 10
 stop() (c1218.connection.Connection method), 4

T

table_permissions (c1219.access.security.C1219SecurityAccess attribute), 10
 tableid (c1218.data.C1218ReadRequest attribute), 7
 tableid (c1218.data.C1218WriteRequest attribute), 7
 terminate (c1218.data.C1218TerminateRequest attribute), 7

U

update_last_call_statuses() (c1219.access.telephone.C1219TelephoneAccess method), 11
 use_extended_status (c1219.access.telephone.C1219TelephoneAccess attribute), 11
 userid (c1218.data.C1218LogonRequest attribute), 6
 username (c1218.data.C1218LogonRequest attribute), 6

W

wait (c1218.data.C1218WaitRequest attribute), 6
 write (c1218.data.C1218WriteRequest attribute), 7
 write() (c1218.connection.ConnectionBase method), 5