astm Documentation

Release 0.4.1

Alexander Shorin

March 16, 2013

CONTENTS

Contents:

CHAPTER

ONE

ASTM.CONSTANTS :: ASTM CONSTANT VALUES

astm.constants.**ACK = '\x06'**
Command accepted token.

astm.constants.**COMPONENT_SEP = '^'**
Field components delimeter.

astm.constants.**CRLF = '\r\n'**
CR + LF shortcut.

astm.constants.**ENCODING = 'latin-1'**
ASTM specification base encoding.

astm.constants.**ENQ = '\x05'**
ASTM session initialization token.

astm.constants.**EOT = '\x04'**
ASTM session termination token.

astm.constants.**ESCAPE_SEP = '&'**
Date escape token.

astm.constants.**ETB = '\x17'**
Message chunk end token.

astm.constants.**ETX = '\x03'**
Message end token.

astm.constants.**FIELD_SEP = '|'**
Record fields delimeter.

astm.constants.**NAK = '\x15'**
Command rejected token.

astm.constants.**RECORD_SEP = '\r'**
Message records delimeter.

astm.constants.**REPEAT_SEP = '\\'**
Delimeter for repeated fields.

astm.constants.**STX = '\x02'**
Message start token.

CHAPTER

TWO

ASTM.CODEC :: BASE DECODING AND ENCODING FUNCTIONS

astm.codec.**decode**(*data*, *encoding='latin-1'*)

Common ASTM decoding function that tries to guess which kind of data it handles.

If *data* starts with STX character (`0x02`) than probably it is full ASTM message with checksum and other system characters.

If *data* starts with digit character (`0-9`) than probably it is frame of records leading by his sequence number. No checksum is expected in this case.

Otherwise it counts *data* as regular record structure.

Note, that *data* should be bytes, not unicode string even if you know his *encoding*.

Parameters

- **data** (*bytes*) – ASTM data object.
- **encoding** (*str*) – Data encoding.

Returns List of ASTM records with unicode data.

Return type list

astm.codec.**decode_component**(*field*, *encoding*)

Decodes ASTM field component.

astm.codec.**decode_frame**(*frame*, *encoding*)

Decodes ASTM frame: list of records followed by sequence number.

astm.codec.**decode_message**(*message*, *encoding*)

Decodes complete ASTM message that is sent or received due communication routines. It should contains checksum that would be additionally verified.

Parameters

- **message** (*bytes*) – ASTM message.
- **encoding** (*str*) – Data encoding.

Returns

Tuple of three elements:

- `int` frame sequence number.
- `list` of records with unicode data.

- `bytes` checksum.

Raises

- `ValueError` if ASTM message is malformed.
- `AssertionError` if checksum verification fails.

`astm.codec.`**`decode_record`**(*record*, *encoding*)
Decodes ASTM record message.

`astm.codec.`**`decode_repeated_component`**(*component*, *encoding*)
Decodes ASTM field repeated component.

`astm.codec.`**`encode`**(*records*, *encoding='latin-1'*, *size=None*)
Encodes list of records into single ASTM message, also called as "packed" message.

If you need to get each record as standalone message use `iter_encode()` instead.

If the result message is too large (greater than `MAX_MESSAGE_SIZE`), than it will be splitted by chunks.

Parameters

- **records** (*list*) – List of ASTM records.
- **encoding** (*str*) – Data encoding.

Returns List of ASTM message chunks.

Return type list

`astm.codec.`**`encode_component`**(*component*, *encoding*)
Encodes ASTM record field components.

`astm.codec.`**`encode_message`**(*seq*, *records*, *encoding*)
Encodes ASTM message.

Parameters

- **seq** (*int*) – Frame sequence number.
- **records** (*list*) – List of ASTM records.
- **encoding** (*str*) – Data encoding.

Returns ASTM complete message with checksum and other control characters.

Return type str

`astm.codec.`**`encode_record`**(*record*, *encoding*)
Encodes single ASTM record.

Parameters

- **record** (*list*) – ASTM record. Each `str`-typed item counted as field value, one level nested `list` counted as components and second leveled - as repeated components.
- **encoding** (*str*) – Data encoding.

Returns Encoded ASTM record.

Return type str

`astm.codec.`**`encode_repeated_component`**(*components*, *encoding*)
Encodes repeated components.

`astm.codec.`**`is_chunked_message`**(*message*)
Checks plain message for chunked byte.

astm.codec.**iter_encode**(*records*, *encoding='latin-1'*, *size=None*)
Encodes and emits each record as separate message.

If the result message is too large (greater than MAX_MESSAGE_SIZE), than it will be splitted by chunks.

Yields ASTM message chunks.

Return type str

astm.codec.**make_checksum**(*message*)
Calculates checksum for specified message.

Parameters **message** (*bytes*) – ASTM message.

Returns Checksum value that is actually byte sized integer in hex base

Return type bytes

CHAPTER

THREE

ASTM.MAPPING :: MESSAGE OBJECT MAPPINGS

class astm.mapping.**Component**(**args*, ***kwargs*)
ASTM component mapping class.

astm.mapping.**ComponentField**
Mapping field for storing record component.

astm.mapping.**ConstantField**
Mapping field for constant values.

```
>>> class Record(Mapping):
...     type = ConstantField(default='S')
>>> rec = Record()
>>> rec.type
'S'
>>> rec.type = 'W'
Traceback (most recent call last):
    ...
ValueError: Field changing not allowed
```

astm.mapping.**DateField**
Mapping field for storing date/time values.

astm.mapping.**DateTimeField**
Mapping field for storing date/time values.

astm.mapping.**DecimalField**
Mapping field for decimal values.

astm.mapping.**Field**
Base mapping field class.

astm.mapping.**IntegerField**
Mapping field for integer values.

astm.mapping.**NotUsedField**
Mapping field for value that should be used. Acts as placeholder. On attempt to assign something to it raises `UserWarning` and rejects assigned value.

class astm.mapping.**Record**(**args*, ***kwargs*)
ASTM record mapping class.

astm.mapping.**RepeatedComponentField**
Mapping field for storing list of record components.

astm.mapping.**SetField**
Mapping field for predefined set of values.

astm.mapping.**TextField**
Mapping field for string values.

astm.mapping.**TimeField**
Mapping field for storing times.

CHAPTER

FOUR

ASTM.RECORDS :: BASE ASTM RECORDS

Common ASTM records structure.

This module contains base ASTM records mappings with only defined common required fields for most implementations. Others are marked as `NotUsedField` and should be defined explicitly for your ASTM realisation.

astm.records.**HeaderRecord**

#	ASTM Field #	ASTM Name	Python alias
1	7.1.1	ASTM Record Type ID	type
2	7.1.2	Delimiter Definition	delimeter
3	7.1.3	Message Control ID	message_id
4	7.1.4	Access Password	password
5	7.1.5	Sender Name or ID	sender
6	7.1.6	Sender Street Address	address
7	7.1.7	Reserved Field	reserved
8	7.1.8	Sender Telephone Number	phone
9	7.1.9	Characteristics of Sender	caps
10	7.1.10	Receiver ID	receiver
11	7.1.11	Comments	comments
12	7.1.12	Processing ID	processing_id
13	7.1.13	Version Number	version
14	7.1.14	Date/Time of Message	timestamp

alias of `GenericRecord`

astm.records.**PatientRecord**

#	ASTM Field #	ASTM Name	Python alias
1	8.1.1	Record Type ID	type
2	8.1.2	Sequence Number	seq
3	8.1.3	Practice Assigned Patient ID	practice_id
4	8.1.4	Laboratory Assigned Patient ID	laboratory_id
5	8.1.5	Patient ID	id
6	8.1.6	Patient Name	name
7	8.1.7	Mother's Maiden Name	maiden_name
8	8.1.8	Birthdate	birthdate
9	8.1.9	Patient Sex	sex
10	8.1.10	Patient Race-Ethnic Origin	race
11	8.1.11	Patient Address	address
12	8.1.12	Reserved Field	reserved
13	8.1.13	Patient Telephone Number	phone
14	8.1.14	Attending Physician ID	physician_id
15	8.1.15	Special Field #1	special_1
16	8.1.16	Special Field #2	special_2
17	8.1.17	Patient Height	height
18	8.1.18	Patient Weight	weight
19	8.1.19	Patient's Known Diagnosis	diagnosis
20	8.1.20	Patient's Active Medication	medication
21	8.1.21	Patient's Diet	diet
22	8.1.22	Practice Field No. 1	practice_field_1
23	8.1.23	Practice Field No. 2	practice_field_2
24	8.1.24	Admission/Discharge Dates	admission_date
25	8.1.25	Admission Status	admission_status
26	8.1.26	Location	location

alias of `GenericRecord`

`astm.records.`**`OrderRecord`**

#	ASTM Field #	ASTM Name	Python alias
1	9.4.1	Record Type ID	type
2	9.4.2	Sequence Number	seq
3	9.4.3	Specimen ID	sample_id
4	9.4.4	Instrument Specimen ID	instrument
5	9.4.5	Universal Test ID	test
6	9.4.6	Priority	priority
7	9.4.7	Requested/Ordered Date/Time	created_at
8	9.4.8	Specimen Collection Date/Time	sampled_at
9	9.4.9	Collection End Time	collected_at
10	9.4.10	Collection Volume	volume
11	9.4.11	Collector ID	collector
12	9.4.12	Action Code	action_code
13	9.4.13	Danger Code	danger_code
14	9.4.14	Relevant Information	clinical_info
15	9.4.15	Date/Time Specimen Received	delivered_at
16	9.4.16	Specimen Descriptor	biomaterial
17	9.4.17	Ordering Physician	physician
18	9.4.18	Physician's Telephone #	physician_phone
19	9.4.19	User Field No. 1	user_field_1
20	9.4.20	User Field No. 2	user_field_2
21	9.4.21	Laboratory Field No. 1	laboratory_field_1
22	9.4.22	Laboratory Field No. 2	laboratory_field_2
23	9.4.23	Date/Time Reported	modified_at
24	9.4.24	Instrument Charge	instrument_charge
25	9.4.25	Instrument Section ID	instrument_section
26	9.4.26	Report Type	report_type

alias of `GenericRecord`

`astm.records.`**`ResultRecord`**

#	ASTM Field #	ASTM Name	Python alias
1	10.1.1	Record Type ID	type
2	10.1.2	Sequence Number	seq
3	10.1.3	Universal Test ID	test
4	10.1.4	Data or Measurement Value	value
5	10.1.5	Units	units
6	10.1.6	Reference Ranges	references
7	10.1.7	Result Abnormal Flags	abnormal_flag
8	10.1.8	Nature of Abnormal Testing	abnormality_nature
9	10.1.9	Results Status	status
10	10.1.10	Date of Change in Instrument Normative Values	norms_changed_at
11	10.1.11	Operator Identification	operator
12	10.1.12	Date/Time Test Started	started_at
13	10.1.13	Date/Time Test Complete	completed_at
14	10.1.14	Instrument Identification	instrument

alias of `GenericRecord`

`astm.records.`**`CommentRecord`**

#	ASTM Field #	ASTM Name	Python alias
1	11.1.1	Record Type ID	type
2	11.1.2	Sequence Number	seq
3	11.1.3	Comment Source	source
4	11.1.4	Comment Text	data
5	11.1.5	Comment Type	ctype

alias of `GenericRecord`

`astm.records.`**`TerminatorRecord`**

#	ASTM Field #	ASTM Name	Python alias
1	13.1.1	Record Type ID	type
2	13.1.2	Sequence Number	seq
3	13.1.3	Termination code	code

alias of `GenericRecord`

CHAPTER

FIVE

ASTM.ASYNCLIB :: ASYNCHRONOUS SOCKET HANDLER

astm.asynclib.**loop**(*timeout=30.0*, *map=None*, *count=None*)

Enter a polling loop that terminates after count passes or all open channels have been closed. All arguments are optional. The *count* parameter defaults to None, resulting in the loop terminating only when all channels have been closed. The *timeout* argument sets the timeout parameter for the appropriate `select()` or `poll()` call, measured in seconds; the default is 30 seconds. The *use_poll* parameter, if true, indicates that `poll()` should be used in preference to `select()` (the default is `False`).

The *map* parameter is a dictionary whose items are the channels to watch. As channels are closed they are deleted from their map. If *map* is omitted, a global map is used. Channels (instances of `asyncore.dispatcher`, `asynchat.async_chat` and subclasses thereof) can freely be mixed in the map.

class astm.asynclib.**Dispatcher**(*sock=None*, *map=None*)

The `Dispatcher` class is a thin wrapper around a low-level socket object. To make it more useful, it has a few methods for event-handling which are called from the asynchronous loop. Otherwise, it can be treated as a normal non-blocking socket object.

The firing of low-level events at certain times or in certain connection states tells the asynchronous loop that certain higher-level events have taken place. For example, if we have asked for a socket to connect to another host, we know that the connection has been made when the socket becomes writable for the first time (at this point you know that you may write to it with the expectation of success). The implied higher-level events are:

Event	Description
`handle_connect()`	Implied by the first read or write event
`handle_close()`	Implied by a read event with no data available
`handle_accept()`	Implied by a read event on a listening socket

During asynchronous processing, each mapped channel's `readable()` and `writable()` methods are used to determine whether the channel's socket should be added to the list of channels `select()`ed or `poll()`ed for read and write events.

accept()

Accept a connection.

The socket must be bound to an address and listening for connections. The return value can be either `None` or a pair `(conn, address)` where *conn* is a *new* socket object usable to send and receive data on the connection, and *address* is the address bound to the socket on the other end of the connection.

When `None` is returned it means the connection didn't take place, in which case the server should just ignore this event and keep listening for further incoming connections.

bind(*address*)

Bind the socket to *address*.

The socket must not already be bound. The format of *address* depends on the address family — refer to the `socket` documentation for more information. To mark the socket as re-usable (setting the `SO_REUSEADDR` option), call the `Dispatcher` object's `set_reuse_addr()` method.

close()
Close the socket.

All future operations on the socket object will fail. The remote end-point will receive no more data (after queued data is flushed). Sockets are automatically closed when they are garbage-collected.

connect(*address*)
As with the normal socket object, *address* is a tuple with the first element the host to connect to, and the second the port number.

create_socket(*family*, *type*)
This is identical to the creation of a normal socket, and will use the same options for creation. Refer to the `socket` documentation for information on creating sockets.

handle_accept()
Called on listening channels (passive openers) when a connection can be established with a new remote endpoint that has issued a `connect()` call for the local endpoint.

handle_close()
Called when the socket is closed.

handle_connect()
Called when the active opener's socket actually makes a connection. Might send a "welcome" banner, or initiate a protocol negotiation with the remote endpoint, for example.

handle_error()
Called when an exception is raised and not otherwise handled. The default version prints a condensed traceback.

handle_write()
Called when the asynchronous loop detects that a writable socket can be written. Often this method will implement the necessary buffering for performance. For example:

```
def handle_write(self):
    sent = self.send(self.buffer)
    self.buffer = self.buffer[sent:]
```

listen(*num*)
Listen for connections made to the socket.

The *num* argument specifies the maximum number of queued connections and should be at least 1; the maximum value is system-dependent (usually 5).

readable()
Called each time around the asynchronous loop to determine whether a channel's socket should be added to the list on which read events can occur. The default method simply returns `True`, indicating that by default, all channels will be interested in read events.

recv(*buffer_size*)
Read at most *buffer_size* bytes from the socket's remote end-point.

An empty string implies that the channel has been closed from the other end.

send(*data*)
Send *data* to the remote end-point of the socket.

writable()
Called each time around the asynchronous loop to determine whether a channel's socket should be added

to the list on which write events can occur. The default method simply returns `True`, indicating that by default, all channels will be interested in write events.

class `astm.asynclib.`**`AsyncChat`**(*sock=None*, *map=None*)

This class is an abstract subclass of `Dispatcher`. To make practical use of the code you must subclass `AsyncChat`, providing meaningful meth:*found_terminator* method. The `Dispatcher` methods can be used, although not all make sense in a message/response context.

Like `Dispatcher`, `AsyncChat` defines a set of events that are generated by an analysis of socket conditions after a `select()` call. Once the polling loop has been started the `AsyncChat` object's methods are called by the event-processing framework with no action on the part of the programmer.

`close_when_done`()

Automatically close this channel once the outgoing queue is empty.

`discard_buffers`()

In emergencies this method will discard any data held in the input and output buffers.

`encoding` = **'latin-1'**

Default encoding.

`flush`()

Sends all data from outgoing queue.

`found_terminator`()

Called when the incoming data stream matches the `termination` condition. The default method, which must be overridden, raises a `NotImplementedError` exception. The buffered input data should be available via an instance attribute.

`pull`(*data*)

Puts *data* into incoming queue. Also available by alias *collect_incoming_data*.

`push`(*data*)

Pushes data on to the channel's fifo to ensure its transmission. This is all you need to do to have the channel write the data out to the network.

`readable`()

Predicate for inclusion in the readable for select()

`recv_buffer_size` = **4096**

The asynchronous input buffer size.

`send_buffer_size` = **4096**

The asynchronous output buffer size.

`strip_terminator` = **True**

Remove terminator from the result data.

`terminator`

The input delimiter andthe terminating condition to be recognized on the channel. Maay be any of three types of value, corresponding to three different ways to handle incoming protocol data.

term	Description
string	Will call `found_terminator()` when the string is found in the input stream
integer	Will call `found_terminator()` when the indicated number of characters have been received
`None`	The channel continues to collect data forever

Note that any data following the terminator will be available for reading by the channel after `found_terminator()` is called.

use_encoding = False
Encoding usage is not enabled by default, because that is a sign of an application bug that we don't want to pass silently.

writable()
Predicate for inclusion in the writable for select()

CHAPTER

SIX

ASTM PROTOCOL IMPLEMENTATION

6.1 astm.protocol :: Common protocol routines

astm.protocol.**STATE** = **ASTMState(init=0, opened=1, transfer=2)**
ASTM protocol states set.

class astm.protocol.**ASTMProtocol**(*sock=None*, *map=None*, *timeout=None*)
Common ASTM protocol routines.

astm_header
ASTM header record class.

alias of GenericRecord

astm_terminator
ASTM terminator record class.

alias of GenericRecord

dispatch(*data*)
Dispatcher of received data.

is_chunked_transfer = **None**
Flag about chunked transfer.

on_ack()
Calls on <ACK> message receiving.

on_enq()
Calls on <ENQ> message receiving.

on_eot()
Calls on <EOT> message receiving.

on_init_state()
Calls on set state INIT (0)

on_message()
Calls on ASTM message receiving.

on_nak()
Calls on <NAK> message receiving.

on_opened_state()
Calls on set state OPENED (1)

on_timeout()
Calls when timeout event occurs. Used to limit time for waiting response data.

on_transfer_state()
Calls on set state TRANSFER (2)

set_init_state()
Sets handler state to INIT (0).

In ASTM specification this state also called as *neutral* which means that handler is ready to establish data transfer.

set_opened_state()
Sets handler state to OPENED (1).

Intermediate state that only means for client implementation. On this state client had already sent <ENQ> and awaits for <ACK> or <NAK> response. On <ACK> it switched his state to *transfer*.

set_transfer_state()
Sets handler state to TRANSFER (2).

In this state handler is able to send or receive ASTM messages depending on his role (client or server). At the end of data transfer client should send <EOT> and switch state to *init*.

state
ASTM handler state value:

- `init`: Neutral state
- `opened`: ENQ message was sent, waiting for ACK
- `transfer`: Data transfer processing

timeout = None
Operation timeout value.

6.2 `astm.server` :: ASTM Server

class `astm.server.`**BaseRecordsDispatcher**(*encoding=None*)
Dispatcher of received ASTM records by `RequestHandler`.

encoding = 'latin-1'
Encoding of received messages.

on_comment(*record*)
Comment record handler.

on_header(*record*)
Header record handler.

on_order(*record*)
Order record handler.

on_patient(*record*)
Patient record handler.

on_result(*record*)
Result record handler.

on_terminator(*record*)
Terminator record handler.

on_unknown(*record*)
Fallback handler for dispatcher.

class `astm.server.`**`RequestHandler`**(*sock*, *dispatcher*)
ASTM protocol request handler.

Parameters

- **sock** – Socket object.
- **dispatcher** (`BaseRecordsDispatcher`) – Request handler records dispatcher instance.

class `astm.server.`**`Server`**(*host='localhost'*, *port=15200*, *request=None*, *dispatcher=None*)
Asyncore driven ASTM server.

Parameters

- **host** (*str*) – Server IP address or hostname.
- **port** (*int*) – Server port number.
- **request** – Custom server request handler. If omitted the `RequestHandler` will be used by default.
- **dispatcher** – Custom request handler records dispatcher. If omitted the `BaseRecordsDispatcher` will be used by default.

`dispatcher`
alias of `BaseRecordsDispatcher`

`request`
alias of `RequestHandler`

`serve_forever`(**args*, ***kwargs*)
Enters into the `polling loop` to let server handle incoming requests.

6.3 `astm.client` :: ASTM Client

class `astm.client.`**`Client`**(*emitter*, *host='localhost'*, *port=15200*, *timeout=20*, *retry_attempts=3*, *records_sm=<astm.client.RecordsStateMachine object at 0x2893a90>*)
Common ASTM client implementation.

Parameters

- **emitter** (*function*) – Generator function that will produce ASTM records.
- **host** (*str*) – Server IP address or hostname.
- **port** (*int*) – Server port number.
- **timeout** (*int*) – Time to wait for response from server. If response wasn't received, the `on_timeout()` will be called. If `None` this timer will be disabled.
- **retry_attempts** (*int*) – Number or attempts to send record to server.
- **records_sm** – Records type state machine that controls right order of generated records by the emitter. The default state machine may be replaced by any callable which takes single argument as record type.

Type callable

`emit_header`()
Returns Header record.

`emit_terminator`()
Returns Terminator record.

push(*data*, *with_timer=True*)

Pushes data on to the channel's fifo to ensure its transmission with optional timer. Timer is used to control receiving response for sent data within specified time frame. If it's doesn't `on_timeout()` method will be called and data may be sent once again.

Parameters

- **data** (*str*) – Sending data.
- **with_timer** (*bool*) – Flag to use timer.

push_record(*record*)

Sends single ASTM record and autoincrement frame sequence number.

Parameters **record** (list or `Record`) – ASTM record object.

Records should be sent in specific order or `AssertionError` will be raised:

Legend:

- H: `HeaderRecord`
- P: `PatientRecord`
- O: `OrderRecord`
- R: `ResultRecord`
- C: `CommentRecord`
- L: `TerminatorRecord`

Previous record type	Current record type
None, this is first record	H
H	P, L
P	P, O, C, L
O	O, R, C, L
R	R, C, L
L	H

run(**args*, ***kwargs*)

Enters into the `polling loop` to let client send outgoing requests.

CHAPTER

SEVEN

ASTM IMPLEMETATION MODULES

7.1 `astm.omnilab` :: Omnilab LabOnline

Based on:

file LABONLINE - HOST connection specifications

author Giuseppe Iannucci

revision 2

date 2011-05-31

7.1.1 `astm.omnilab.client` - LabOnline client implementation

class `astm.omnilab.client.`**`Header`**(**args*, ***kwargs*)

ASTM header record.

Parameters

- **type** (*str*) – Record Type ID. Always `H`.
- **delimeter** (*str*) – Delimiter Definition. Always `\^&`.
- **message_id** (*None*) – Message Control ID. Not used.
- **password** (*None*) – Access Password. Not used.
- **sender** (`Sender`) – Information about sender. Optional.
- **address** (*None*) – Sender Street Address. Not used.
- **reserved** (*None*) – Reserved Field. Not used.
- **phone** (*None*) – Sender Telephone Number. Not used.
- **chars** (*None*) – Sender Characteristics. Not used.
- **receiver** (*None*) – Information about receiver. Not used.
- **comments** (*None*) – Comments. Not used.
- **processing_id** (*str*) – Processing ID. Always `P`.
- **version** (*str*) – ASTM Version Number. Always `E 1394-97`.
- **timestamp** (*datetime.datetime*) – Date and Time of Message

class `astm.omnilab.client.`**`Patient`**(**args*, ***kwargs*)
ASTM patient record.

Parameters

- **type** (*str*) – Record Type ID. Always `P`.
- **seq** (*int*) – Sequence Number. Required.
- **practice_id** (*str*) – Practice Assigned Patient ID. Required. Length: 12.
- **laboratory_id** (*str*) – Laboratory Assigned Patient ID. Required. Length: 16.
- **id** (*None*) – Patient ID. Not used.
- **name** (`PatientName`) – Patient name.
- **maiden_name** (*None*) – Mother's Maiden Name. Not used.
- **birthdate** (*datetime.date*) – Birthdate.
- **sex** (*str*) – Patient Sex. One of: `M` (male), `F` (female), `I` (animal), `None` is unknown.
- **race** (*None*) – Patient Race-Ethnic Origin. Not used.
- **address** (*None*) – Patient Address. Not used.
- **reserved** (*None*) – Reserved Field. Not used.
- **phone** (*None*) – Patient Telephone Number. Not used.
- **physician_id** (*None*) – Attending Physician. Not used.
- **special_1** (*None*) – Special Field #1. Not used.
- **special_2** (*int*) – Patient source. Possible values: - `0`: internal patient; - `1`: external patient.
- **height** (*None*) – Patient Height. Not used.
- **weight** (*None*) – Patient Weight. Not used.
- **diagnosis** (*None*) – Patient's Known Diagnosis. Not used.
- **medications** (*None*) – Patient's Active Medications. Not used.
- **diet** (*None*) – Patient's Diet. Not used.
- **practice_1** (*None*) – Practice Field No. 1. Not used.
- **practice_2** (*None*) – Practice Field No. 2. Not used.
- **admission_date** (*None*) – Admission/Discharge Dates. Not used.
- **admission_status** (*None*) – Admission Status. Not used.
- **location** (*str*) – Patient location. Length: 20.
- **diagnostic_code_nature** (*None*) – Nature of diagnostic code. Not used.
- **diagnostic_code** (*None*) – Diagnostic code. Not used.
- **religion** (*None*) – Patient religion. Not used.
- **martial_status** (*None*) – Martian status. Not used.
- **isolation_status** (*None*) – Isolation status. Not used.
- **language** (*None*) – Language. Not used.
- **hospital_service** (*None*) – Hospital service. Not used.

- **hospital_institution** (*None*) – Hospital institution. Not used.
- **dosage_category** (*None*) – Dosage category. Not used.

class astm.omnilab.client.**Order**(**args*, ***kwargs*)

ASTM order record.

Parameters

- **type** (*str*) – Record Type ID. Always `O`.
- **seq** (*int*) – Sequence Number. Required.
- **sample_id** (*str*) – Sample ID number. Required. Length: 12.
- **instrument** (*None*) – Instrument specimen ID. Not used.
- **test** (`Test`) – Test information structure (aka Universal Test ID).
- **priority** (*str*) – Priority flag. Required. Possible values: - `S`: stat; -`R`: routine.
- **created_at** (*datetime.datetime*) – Ordered date and time. Required.
- **sampled_at** (*datetime.datetime*) – Specimen collection date and time.
- **collected_at** (*None*) – Collection end time. Not used.
- **volume** (*None*) – Collection volume. Not used.
- **collector** (*None*) – Collector ID. Not used.
- **action_code** (*str*) – Action code. Required. Possible values: - `C`: cancel works for specified tests; - `A`: add tests to existed specimen; - `N`: create new order; - `R`: rerun tests for specified order;
- **danger_code** (*None*) – Danger code. Not used.
- **clinical_info** (*None*) – Revelant clinical info. Not used.
- **delivered_at** (*None*) – Date/time specimen received.
- **biomaterial** (*str*) – Sample material code. Length: 20.
- **physician** (*None*) – Ordering Physician. Not used.
- **physician_phone** (*None*) – Physician's phone number. Not used.
- **user_field_1** (*str*) – An optional field, it will be send back unchanged to the host along with the result. Length: 20.
- **user_field_2** (*str*) – An optional field, it will be send back unchanged to the host along with the result. Length: 1024.
- **laboratory_field_1** (*str*) – In multi-laboratory environment it will be used to indicate which laboratory entering the order. Length: 20.
- **laboratory_field_2** (*str*) – Primary tube code. Length: 12.
- **modified_at** (*None*) – Date and time of last result modification. Not used.
- **instrument_charge** (*None*) – Instrument charge to computer system. Not used.
- **instrument_section** (*None*) – Instrument section id. Not used.
- **report_type** (*str*) – Report type. Always `O` which means normal order request.
- **reserved** (*None*) – Reserved. Not used.
- **location_ward** (*None*) – Location ward of specimen collection. Not used.

- **infection_flag** (*None*) – Nosocomial infection flag. Not used.
- **specimen_service** (*None*) – Specimen service. Not used.
- **laboratory** (*str*) – Production laboratory: in multi-laboratory environment indicates laboratory expected to process the order. Length: 20.

class `astm.omnilab.client.`**`Result`**(**args*, ***kwargs*)
ASTM patient record.

Parameters

- **type** (*str*) – Record Type ID. Always `R`.
- **seq** (*int*) – Sequence Number. Required.
- **test** (`Test`) – Test information structure (aka Universal Test ID).
- **value** (*None*) – Measurement value. Numeric, coded or free text value depending on result type. Required. Length: 1024.
- **units** (*None*) – Units. Not used.
- **references** (*None*) – Reference ranges. Not used.
- **abnormal_flag** (*None*) – Result abnormal flag. Not used.
- **abnormality_nature** (*None*) – Nature of abnormality testing. Not used.
- **status** (*None*) – Result status. Not used.
- **normatives_changed_at** (*None*) – Date of changes in instrument normative values or units. Not used.
- **operator** (*None*) – Operator ID. Not used.
- **started_at** (*None*) – When works on test was started on. Not used.
- **completed_at** (*datetime.datetime*) – When works on test was done. Required.
- **instrument** (*None*) – Instrument ID. Not used.

class `astm.omnilab.client.`**`Comment`**(**args*, ***kwargs*)
ASTM patient record.

Parameters

- **type** (*str*) – Record Type ID. Always `C`.
- **seq** (*int*) – Sequence Number. Required.
- **source** (*str*) – Comment source. Always `L`.
- **data** (`CommentData`) – Measurement value. Numeric, coded or free text value depending on result type. Required. Length: 1024.
- **ctype** (*str*) – Comment type. Always `G`.

class `astm.omnilab.client.`**`Terminator`**(**args*, ***kwargs*)
ASTM terminator record.

Parameters

- **type** (*str*) – Record Type ID. Always `L`.
- **seq** (*int*) – Sequential number. Always `1`.
- **code** (*str*) – Termination code. Always `N`.

astm.omnilab.client.**CommentData**
Comment control data structure.

alias of GenericComponent

astm.omnilab.client.**PatientAge**
Patient age structure.

Parameters

- **value** (*int*) – Age value.
- **unit** (*str*) – Age unit. One of: years, months, days.

alias of GenericComponent

astm.omnilab.client.**Test**
Test Component also known as Universal Test ID.

Parameters

- **_** (*None*) – Reserved. Not used.
- **__** (*None*) – Reserved. Not used.
- **___** (*None*) – Reserved. Not used.
- **assay_code** (*str*) – Assay code. Required. Length: 20.
- **assay_name** (*str*) – Assay name. Length: 8.

alias of GenericComponent

7.1.2 astm.omnilab.server - LabOnline server implementation

class astm.omnilab.server.**RecordsDispatcher**(**args*, ***kwargs*)
Omnilab specific records dispatcher. Automatically wraps records by related mappings.

class astm.omnilab.server.**Header**(**args*, ***kwargs*)
ASTM header record.

Parameters

- **type** (*str*) – Record Type ID. Always H.
- **delimeter** (*str*) – Delimiter Definition. Always \^&.
- **message_id** (*None*) – Message Control ID. Not used.
- **password** (*None*) – Access Password. Not used.
- **sender** (Sender) – Information about sender. Optional.
- **address** (*None*) – Sender Street Address. Not used.
- **reserved** (*None*) – Reserved Field. Not used.
- **phone** (*None*) – Sender Telephone Number. Not used.
- **chars** (*None*) – Sender Characteristics. Not used.
- **receiver** (*None*) – Information about receiver. Not used.
- **comments** (*None*) – Comments. Not used.
- **processing_id** (*str*) – Processing ID. Always P.
- **version** (*str*) – ASTM Version Number. Always E 1394-97.

- **timestamp** (*datetime.datetime*) – Date and Time of Message

astm.omnilab.server.**Patient**
alias of CommonPatient

class astm.omnilab.server.**Order**(**args*, ***kwargs*)
ASTM order record.

Parameters

- **type** (*str*) – Record Type ID. Always O.
- **seq** (*int*) – Sequence Number. Required.
- **sample_id** (*str*) – Sample ID number. Required. Length: 12.
- **instrument** (Instrument) – Instrument specimen ID.
- **test** (Test) – Test information structure (aka Universal Test ID).
- **priority** (*str*) – Priority flag. Required. Possible values: - S: stat; -R: routine.
- **created_at** (*datetime.datetime*) – Ordered date and time. Required.
- **sampled_at** (*datetime.datetime*) – Specimen collection date and time.
- **collected_at** (*None*) – Collection end time. Not used.
- **volume** (*None*) – Collection volume. Not used.
- **collector** (*None*) – Collector ID. Not used.
- **action_code** (*str*) – Action code. Required. Possible values: - None: normal order result; - Q: quality control;
- **danger_code** (*None*) – Danger code. Not used.
- **clinical_info** (*None*) – Revelant clinical info. Not used.
- **delivered_at** (*None*) – Date/time specimen received.
- **biomaterial** (*str*) – Sample material code. Length: 20.
- **physician** (*None*) – Ordering Physician. Not used.
- **physician_phone** (*None*) – Physician's phone number. Not used.
- **user_field_1** (*str*) – An optional field, it will be send back unchanged to the host along with the result. Length: 20.
- **user_field_2** (*str*) – An optional field, it will be send back unchanged to the host along with the result. Length: 1024.
- **laboratory_field_1** (*None*) – Laboratory field #1. Not used.
- **laboratory_field_2** (*str*) – Primary tube code. Length: 12.
- **modified_at** (*None*) – Date and time of last result modification. Not used.
- **instrument_charge** (*None*) – Instrument charge to computer system. Not used.
- **instrument_section** (*None*) – Instrument section id. Not used.
- **report_type** (*str*) – Report type. Always F which means final order request.
- **reserved** (*None*) – Reserved. Not used.
- **location_ward** (*None*) – Location ward of specimen collection. Not used.
- **infection_flag** (*None*) – Nosocomial infection flag. Not used.

- **specimen_service** (*None*) – Specimen service. Not used.
- **laboratory** (*None*) – Production laboratory. Not used.

class astm.omnilab.server.**Result**(**args*, ***kwargs*)
ASTM patient record.

Parameters

- **type** (*str*) – Record Type ID. Always `R`.
- **seq** (*int*) – Sequence Number. Required.
- **test** (`Test`) – Test information structure (aka Universal Test ID).
- **value** (*None*) – Measurement value. Numeric, coded or free text value depending on result type. Required. Length: 1024.
- **units** (*str*) – Units. Length: 20.
- **references** (*str*) – Normal reference value interval.
- **abnormal_flag** (*str*) – Result abnormal flag. Possible values: - `0`: normal result; - `1`: result out of normal values; - `2`: result out of attention values; - `3`: result out of panic values; +10 Delta-check; +1000 Device alarm. Length: 4.
- **abnormality_nature** (*str*) – Nature of abnormality testing. Possible values: - `N`: normal value; - `L`: below low normal range; - `H`: above high normal range; - `LL`: below low critical range; - `HH`: above high critical range.
- **status** (*str*) – Result status. `F` indicates a final result; `R` indicating rerun. Length: 1.
- **normatives_changed_at** (*None*) – Date of changes in instrument normative values or units. Not used.
- **operator** (`Operator`) – Operator ID.
- **started_at** (*datetime.datetime*) – When works on test was started on.
- **completed_at** (*datetime.datetime*) – When works on test was done.
- **instrument** (`Instrument`) – Instrument ID. Required.

class astm.omnilab.server.**Terminator**(**args*, ***kwargs*)
ASTM terminator record.

Parameters

- **type** (*str*) – Record Type ID. Always `L`.
- **seq** (*int*) – Sequential number. Always `1`.
- **code** (*str*) – Termination code. Always `N`.

astm.omnilab.server.**CommentData**
Comment control text structure.

alias of `GenericComponent`

astm.omnilab.server.**CompletionDate**
Completion date time information.

Parameters

- **labonline** (*datetime.datetime*) – Completion date time on LabOnline.
- **analyzer** (*datetime.datetime*) – Completion date time on analyser.

alias of GenericComponent

astm.omnilab.server.**Instrument**
Instrument (analyser) information structure.

Parameters

- _ (*None*) – Reserved. Not used.
- **rack** (*str*) – Rack number. Length: 5.
- **position** (*str*) – Position number. Length: 3.

alias of GenericComponent

astm.omnilab.server.**Operator**
Information about operator that validated results.

Parameters

- **code_on_labonline** (*str*) – Operator code on LabOnline. Length: 12.
- **code_on_analyzer** (*str*) – Operator code on analyser. Length: 20.

alias of GenericComponent

astm.omnilab.server.**Test**
Test Component also known as Universal Test ID.

Parameters

- _ (*None*) – Reserved. Not used.
- __ (*None*) – Reserved. Not used.
- ___ (*None*) – Reserved. Not used.
- **assay_code** (*str*) – Assay code. Required. Length: 20.
- **assay_name** (*str*) – Assay name. Length: 8.
- **dilution** (*str*) – Dilution. Length: 10.
- **status** (*str*) – Assay status. Length: 1.
- **reagent_lot** (*str*) – Reagent lot. Length: 15.
- **reagent_number** (*str*) – Reagent serial number. Length: 5.
- **control_lot** (*str*) – Control lot number. Length: 25.
- **type** (*str*) – Result type value. One of: CE, TX.

alias of GenericComponent

CHAPTER

EIGHT

CHANGELOG

8.1 Release 0.4.1 (2013-02-01)

- Fix timer for Python 2.x

8.2 Release 0.4 (2013-02-01)

- Fix astm.codec module: now it only decodes bytes to unicode and encodes unicode to bytes;
- Add records dispatcher for server request handler;
- Add session support for astm client emitter;
- Repeat ENQ on timeout;
- Code cleanup and refactoring;
- Set minimal Python version to 2.6, but 3.2-3.3 also works well.

8.3 Release 0.3 (2012-12-15)

- Refactor OmniLab module;
- Rename astm.proto module to astm.protocol;
- Add support for network operation timeouts;
- Client now better communicates with ASTM records emitter;
- Improve logging;
- Code cleanup;
- More tests for base functionality.

8.4 Release 0.2 (2012-12-11)

- Fork, mix and refactor asyncore/asynchat modules to astm.asynclib module which provides more suitable methods to implement asynchronous operations for our task;
- Implement ASTM client and server that handles common protocol routines.

8.5 Release 0.1 (2012-12-09)

- Base decode/encode functions for ASTM data format;
- Small object model mapping based on couchdb-python solution that helps to design records as classes, access to fields by name alias and provide autoconversion between Python objects and ASTM raw values;
- Add demo support of OmniLab LabOnline product.

CHAPTER

NINE

LICENCE

Copyright (C) 2012 Alexander Shorin
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER

TEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

a