# python-pcapng Documentation

***Release 0.1a***

**Samuele Santi**

**Mar 26, 2018**

# Contents

Contents:

# CHAPTER 1

# Library usage

Use the *FileScanner* class to iterate over blocks in a pcap-ng archive file, like this:

```
from pcapng import FileScanner

with open('/tmp/mycapture.pcap') as fp:
    scanner = FileScanner(fp)
    for block in scanner:
        pass  # do something with the block...
```

Block types can be checked against blocks in *pcapng.blocks*.

CHAPTER 2

# API Documentation

Contents:

## 2.1 pcapng.blocks

Module containing the definition of known / supported "blocks" of the pcap-ng format.

Each block is a struct-like object with some fields and possibly a variable amount of "items" (usually options).

They can optionally expose some other properties, used eg. to provide better access to decoded information, ...

**class** pcapng.blocks.**Block**(*raw*)
Base class for blocks

**schema = []**

**classmethod from_context**(*raw*, *ctx*)

**class** pcapng.blocks.**SectionMemberBlock**(*raw*, *section*)

**classmethod from_context**(*raw*, *ctx*)

pcapng.blocks.**register_block**(*block*)
Handy decorator to register a new known block type

**class** pcapng.blocks.**SectionHeader**(*raw*, *endianness*)

**magic_number = 168627466**

**schema = [('version_major', IntField(size=16, signed=False)), ('version_minor', IntFie**

**register_interface**(*interface*)
Helper method to register an interface within this section

**add_interface_stats**(*interface_stats*)
Helper method to register interface stats within this section

**version**

**length**

**class** pcapng.blocks.**InterfaceDescription**(*raw*, *section*)

**magic_number = 1**

**schema = [('link_type', IntField(size=16, signed=False)), ('reserved', RawBytes(size=2**

**timestamp_resolution**

**statistics**

**link_type_description**

**class** pcapng.blocks.**BlockWithTimestampMixin**
Block mixin adding properties to better access timestamps of blocks that provide one.

**timestamp**

**timestamp_resolution**

**class** pcapng.blocks.**BlockWithInterfaceMixin**

**interface**

**class** pcapng.blocks.**BasePacketBlock**(*raw*, *section*)
Base class for the "EnhancedPacket" and "Packet" blocks

**class** pcapng.blocks.**EnhancedPacket**(*raw*, *section*)

**magic_number = 6**

**schema = [('interface_id', IntField(size=32, signed=False)), ('timestamp_high', IntFie**

**captured_len**

**packet_len**

**packet_data**

**class** pcapng.blocks.**SimplePacket**(*raw*, *section*)

**magic_number = 3**

**schema = [('packet_simple_payload_info', SimplePacketDataField())]**

**packet_len**

**packet_data**

**class** pcapng.blocks.**Packet**(*raw*, *section*)

**magic_number = 2**

**schema = [('interface_id', IntField(size=16, signed=False)), ('drops_count', IntField(**

**captured_len**

**packet_len**

**packet_data**

**class** pcapng.blocks.**NameResolution**(*raw*, *section*)

**magic_number = 4**

**schema = [('records', ListField(NameResolutionRecordField())), ('options', OptionsFiel**

**class** pcapng.blocks.**InterfaceStatistics**(*raw*, *section*)

**magic_number = 5**

**schema = [('interface_id', IntField(size=32, signed=False)), ('timestamp_high', IntFie**

**class** pcapng.blocks.**UnknownBlock**(*block_type*, *data*)
Class used to represent an unknown block.

Its block type and raw data will be stored directly with no further processing.

## 2.2 pcapng.exceptions

**exception** pcapng.exceptions.**PcapngException**
Base for all the pcapng exceptions

**exception** pcapng.exceptions.**PcapngLoadError**
Indicate an error while loading a pcapng file

**exception** pcapng.exceptions.**PcapngDumpError**
Indicate an error while writing a pcapng file

**exception** pcapng.exceptions.**StreamEmpty**
Exception indicating that the end of the stream was reached and exactly zero bytes were read; usually it simply indicates we reached the end of the stream and no further content is available for reading.

**exception** pcapng.exceptions.**CorruptedFile**
Exception used to indicate that something is wrong with the file structure, possibly due to data corruption.

**exception** pcapng.exceptions.**TruncatedFile**
Exception used to indicate that not all the required bytes could be read before stream end, but the read length was non-zero, indicating a possibly truncated stream.

**exception** pcapng.exceptions.**BadMagic**
Exception used to indicate a failure due to some bad magic number encountered (either the file magic or section header byte order marker).

## 2.3 pcapng.scanner

**class** pcapng.scanner.**FileScanner**(*stream*)
pcap-ng file scanner.

This object can be iterated to get blocks out of a pcap-ng stream (a file or file-like object providing a .read() method).

Example usage:

```
from pcapng import FileScanner

with open('/tmp/mycapture.pcap') as fp:
```

```
scanner = FileScanner(fp)
for block in scanner:
    pass  # do something with the block...
```

**Parameters** **stream** – a file-like object from which to read the data. If you need to parse data from some string you have entirely in-memory, just wrap it in a `io.BytesIO` object.

## 2.4 pcapng.structs

Module providing facilities for handling struct-like data.

`pcapng.structs.`**read_int**(*stream*, *size*, *signed=False*, *endianness='='*)
Read (and decode) an integer number from a binary stream.

**Parameters**

- **stream** – an object providing a `read()` method
- **size** – the size, in bits, of the number to be read. Supported sizes are: 8, 16, 32 and 64 bits.
- **signed** – Whether a signed or unsigned number is required. Defaults to `False` (unsigned int).
- **endianness** – specify the endianness to use to decode the number, in the same format used by Python `struct` module. Defaults to '=' (native endianness). '!' means "network" endianness (big endian), '<' little endian, '>' big endian.

**Returns** the read integer number

`pcapng.structs.`**read_section_header**(*stream*)
Read a section header block from a stream.

---

**Note:** The byte order magic will be removed from the returned data This is ok as we don't need it anymore once we determined the correct endianness of the section.

---

**Returns** a dict containing the `'endianness'` and `'data'` keys that will be used to construct a *SectionHeader* instance.

`pcapng.structs.`**read_block_data**(*stream*, *endianness*)
Read block data from a stream.

Each "block" is in the form:

- 32bit integer indicating the size (including header and size)
- block payload (the above-specified number of bytes minus 12)
- 32bit integer indicating the size (again)

**Parameters**

- **stream** – the stream from which to read data
- **endianness** – Endianness marker, one of '<', '>', '!', '='.

pcapng.structs.**read_bytes**(*stream*, *size*)
Read the given amount of raw bytes from a stream.

**Parameters**

- **stream** – the stream from which to read data
- **size** – the size to read, in bytes

**Returns** the read data

**Raises** *StreamEmpty* if zero bytes were read

**Raises** *TruncatedFile* if 0 < bytes < size were read

pcapng.structs.**read_bytes_padded**(*stream*, *size*, *pad_block_size=4*)
Read the given amount of bytes from a stream, plus read and discard any necessary extra byte to align up to the pad_block_size-sized next block.

**Parameters**

- **stream** – the stream from which to read data
- **size** – the size to read, in bytes

**Returns** the read data

**Raises** *StreamEmpty* if zero bytes were read

**Raises** *TruncatedFile* if 0 < bytes < size were read

**class** pcapng.structs.**StructField**
Abstract base class for struct fields

**load**(*stream*, *endianness*)

**class** pcapng.structs.**RawBytes**(*size*)
Field containing a fixed-width amount of raw bytes

**Parameters** **size** – field size, in bytes

**load**(*stream*, *endianness*)

**class** pcapng.structs.**IntField**(*size*, *signed=False*)
Field containing an integer number.

**Parameters**

- **size** – number size, in bits. Currently supported are 8, 16, 32 and 64-bit integers
- **signed** – whether the number is a signed or unsigned integer. Defaults to False (unsigned)

**load**(*stream*, *endianness*)

**class** pcapng.structs.**OptionsField**(*options_schema*)
Field containing some options.

**Parameters** **options_schema** – Same as the schema parameter to *Options* class constructor.

**load**(*stream*, *endianness*)

**class** pcapng.structs.**PacketDataField**
Field containing some "packet data", used in the Packet and EnhancedPacket blocks.

The packet data is composed of three fields (returned in a tuple):

- captured len (uint32)
- packet len (uint32)

- packet data (captured_len-sized binary data)

**load**(*stream*, *endianness*)

**class** pcapng.structs.**SimplePacketDataField**
Field containing packet data from a SimplePacket object.

The packet data is represented by two fields (returned as a tuple):

- packet_len (uint32)
- packet_data (packet_len-sized binary data)

**load**(*stream*, *endianness*)

**class** pcapng.structs.**ListField**(*subfield*)
A list field is a variable amount of fields of some other type. Used for packets containing multiple "items", such as *NameResolution*.

It will keep loading data using a subfield until a *StreamEmpty* excaption is raised, indicating we reached the end of data (note that a sub-field might even "simulate" a stream end once it reaches some end marker in the file).

Values are returned in a list.

**Parameters** **subfield** – a *StructField* sub-class instance to be used to read values from the stream.

**load**(*stream*, *endianness*)

**class** pcapng.structs.**NameResolutionRecordField**
A name resolution record field contains an item of data used in the *NameResolution* block.

it is composed of three fields:

- record type (uint16)
- record length (uint16)
- payload

Accepted types are:

- 0x00 - end marker
- 0x01 - ipv4 address resolution
- 0x02 - ipv6 address resolution

In both cases, the payload is composed of a valid address in the selected IP version, followed by domain name up to the field end.

**load**(*stream*, *endianness*)

pcapng.structs.**read_options**(*stream*, *endianness*)
Read "options" from an "options block" in a stream, until a StreamEmpty exception is caught, or an end marker is reached.

Each option is composed by:

- option_code (uint16)
- value_length (uint16)
- value (value_length-sized binary data)

The end marker is simply an option with code 0x0000 and no payload

**class** pcapng.structs.**Options**(*schema*, *data*, *endianness*)
Wrapper object used to easily access the contents of an "options" field.

Fields can be accessed either by numerical id or by name (if one was specified in the schema).

---

**Note:** When iterating the object (or calling `keys()` / `iterkeys()` / `viewkeys()`) string keys will be returned if possible in place of numeric keys. (The purpose of this is to achieve better readability, for example, when converting to a dictionary).

---

**Parameters**

- **schema** – Definition of the known options: a list of 2- or 3-tuples (the third argument is optional) representing, respectively, the numeric option code, the option name and the value type.

  The following value types are currently supported:

  - `string`: convert value to a unicode string, using utf-8 encoding
  - `{u,i}{8,16,32,64}`: (un)signed integer of the specified length
  - `ipv4`: a single ipv4 address [4 bytes]
  - `ipv4+mask`: an ipv4 address followed by a netmask [8 bytes]
  - `ipv6`: a single ipv6 address [16 bytes]
  - `ipv6+prefix`: an ipv6 address followed by prefix length [17 bytes]
  - `macaddr`: a mac address [6 bytes]
  - `euiaddr`: a eui address [8 bytes]

- **data** – Initial data for the options. A list of two-tuples: `(code, value)`. Items with the same code may be repeated; only the first one will be accessible using subscript `options[code]`; the others can be accessed using *`get_all()`* and related methods
- **endianness** – The current endianness of the section these options came from. Required in order to load numeric fields.

**get_all**(*name*)
Get all values for the given option

**get_raw**(*name*)
Get raw value for the given option

**get_all_raw**(*name*)
Get all raw values for the given option

**iter_all_items**()
Similar to `iteritems()` but will yield a list of values as the second tuple field.

pcapng.structs.**struct_decode**(*schema*, *stream*, *endianness='='*)
Decode structured data from a stream, following a schema.

**Parameters**

- **schema** – a list of two tuples: ``(name, field)``, where `name` is a string representing the attribute name, and `field` is an instance of a *`StructField`* sub-class, providing a `.load()` method to be called on the stream to get the field value.
- **stream** – a file-like object, providing a `.read()` method, from which data will be read.

- **endianness** – endianness specifier, as accepted by Python struct module (one of <> ! =, defaults to =).

**Returns** a dictionary mapping the field names to decoded data

pcapng.structs.**struct_encode**(*schema*, *obj*, *outstream*, *endianness='='*)
In the future, this function will be used to encode a structure into a stream. For the moment, it just raises `NotImplementedError`.

pcapng.structs.**struct_decode_string**(*schema*, *data*)
Utility function to pass a string to *struct_decode()*

pcapng.structs.**struct_encode_string**(*schema*, *obj*)
Utility function to pass a string to *struct_encode()* and get the result back as a bytes string.

## 2.5 pcapng.utils

pcapng.utils.**unpack_ipv4**(*data*)

pcapng.utils.**unpack_ipv6**(*data*)

pcapng.utils.**unpack_macaddr**(*data*)

pcapng.utils.**unpack_euiaddr**(*data*)

pcapng.utils.**unpack_timestamp_resolution**(*data*)
Unpack a timestamp resolution.

Returns a floating point number representing the timestamp resolution (multiplier).

pcapng.utils.**pack_timestamp_resolution**(*base*, *exponent*)
Pack a timestamp resolution.

**Parameters**

- **base** – 2 or 10
- **exponent** – negative power of the base to be encoded

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## p

# Index

## P



## R



## S



## T



## U



## V