python-pae

Release 0.1.0

Matthias Valvekens

Nov 14, 2021

CONTENTS:

CHAPTER

ONE

API REFERENCE

This is the API reference for *python-pae*, compiled from the docstrings present in the Python source files.

Warning: Any function, class or method that is *not* covered by this documentation is considered private API by definition.

1.1 python-pae

1.1.1 Submodules

python_pae.abstract module

This module defines the basic abstract building blocks of the API.

class python_pae.abstract.**PAEType**(**args*, ***kwds*)

Bases: `Generic[python_pae.abstract.T]`

Provides a serialisation implementation for a particular type of values.

constant_length: Optional[int] = None

If not `None`, the output length of the *write()* method must always be equal to the value of this property.

Length prefixes for types with a fixed byte length can optionally be omitted.

write(*value: python_pae.abstract.T*, *stream: IO*) → int

Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → python_pae.abstract.T

Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

exception python_pae.abstract.**PAEDecodeError**
Bases: ValueError

Raised if an error occurs during PAE decoding.

python_pae.encode module

This module defines helper functions and coroutines for encoding and decoding PAE values.

python_pae.encode.**marshal**(*value: python_pae.encode.T*, *pae_type:* python_pae.abstract.PAEType*[python_pae.encode.T]*) → bytes
Serialise a value into bytes.

Parameters

- **value** – The value to be processed.
- **pae_type** – The *PAEType* that provides the serialisation logic.

Returns A byte string representing the value passed in.

python_pae.encode.**unmarshal**(*packed: bytes*, *pae_type:* python_pae.abstract.PAEType*[python_pae.encode.T]*) → python_pae.encode.T
Decode a byte string back into a value. Inverse operation of *marshal()*.

Parameters

- **packed** – The byte string to be processed.
- **pae_type** – The *PAEType* that provides the deserialisation logic.

Returns A decoded value.

Raises *python_pae.PAEDecodeError* – if an error occurs in the decoding process.

python_pae.encode.**write_prefixed**(*value: python_pae.encode.T*, *pae_type:* python_pae.abstract.PAEType*[python_pae.encode.T]*, *stream: IO*, *length_type:* python_pae.number.PAENumberType, *prefix_if_constant: bool = True*) → int
Write a value to a stream, prefixed with the length of the serialised payload.

Note: The output stream must be seekable for this to work.

Parameters

- **value** – The value to write.
- **pae_type** – The *PAEType* that provides the serialisation logic.
- **stream** – The output stream to write to.
- **length_type** – Numeric type to use for the length prefix.
- **prefix_if_constant** – Flag toggling whether to apply the length prefix if the type being written is a fixed-width type. Defaults to True.

Returns The number of bytes written (including the length prefix, if present).

python_pae.encode.**read_prefixed_coro**(*pae_type:* python_pae.abstract.PAEType*[python_pae.encode.T]*, *stream: IO*, *length_type:* python_pae.number.PAENumberType, *prefix_if_constant: bool = True*)

Coroutine that reads and parses a length-prefixed value. The coroutine yields at most twice. First, the expected length is yielded. Next, the value is decoded and yielded.

Note: The idea is that the caller can abort the parse based on the length value.

Parameters

- **pae_type** – The *PAEType* that provides the deserialisation logic.
- **stream** – The stream to read from.
- **length_type** – Numeric type to use for the length prefix.
- **prefix_if_constant** – Flag toggling whether to expect a length prefix if the type being read is a fixed-width type. Defaults to `True`.

Raises *python_pae.PAEDecodeError* – if an error occurs in the decoding process.

Returns A generator object.

python_pae.encode.**read_pae_coro**(*stream: IO*, *settings:* python_pae.encode.PAEListSettings, *expected_length=None*)

Coroutine to read a (possibly heterogeneous) PAE-encoded list.

The protocol is as follows:

1. First, the coroutine parses and yields the number of list elements.
2. Then, the caller should `.send()` in a *PAEType* object, after which the coroutine will yield a value.
3. Repeat step 2 for each element of the list.

The coroutine-based approach allows for a degree of freedom in the schema (e.g. optional fields), while still parsing on an on-demand basis.

Parameters

- **stream** – The stream to read from.
- **settings** – List encoding settings.
- **expected_length** – The expected byte length of the encoded list payload. If `None`, the length is not enforced.

Raises *python_pae.PAEDecodeError* – if an error occurs in the decoding process.

Returns A generator object.

class python_pae.encode.**PAEListSettings**(*size_type: python_pae.number.PAENumberType = <uint64 (ULLONG)>*, *length_type: Optional[python_pae.number.PAENumberType] = None*, *prefix_if_constant: bool = True*)

Bases: `object`

List encoding settings. The defaults represent the PASETO version of PAE.

size_type: *python_pae.number.PAENumberType* **= <uint64 (ULLONG)>**

Numeric type to use for the list size.

Note: The default is a 64-bit integer for compatibility with PASETO PAE.

length_type: Optional[*python_pae.number.PAENumberType*] = None
Numeric type to use for the length prefixes of the list items.

Note: If unspecified, will be the same as *size_type*.

prefix_if_constant: bool = True
Flag toggling whether to apply the length prefix if the type being written or read is a fixed-width type. Defaults to `True`.

python_pae.number module

This module defines the unsigned number types for our PAE encoding scheme.

class python_pae.number.**PAENumberType**(*value*)
Bases: *python_pae.abstract.PAEType*[int]

Encodes various unsigned integer types. All are encoded in little-endian order.

property constant_length

unpack(*packed: bytes*)

pack(*value: int*)

write(*value: int*, *stream: IO*) → int
Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → int
Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

python_pae.number.**PAE_UCHAR = <uint8 (UCHAR)>**
Unsigned char, encodes to a single byte.

python_pae.number.**PAE_USHORT = <uint16 (USHORT)>**
Unsigned short, encodes to two bytes.

python_pae.number.**PAE_UINT = <uint32 (UINT)>**
Unsigned int, encodes to four bytes.

python_pae.number.**PAE_ULLONG = <uint64 (ULLONG)>**
Unsigned (long) long, encodes to eight bytes.

python_pae.pae_types module

This module defines the serialisation logic for a number of basic types.

class python_pae.pae_types.**PAEBytes**(**args*, ***kwds*)
Bases: *python_pae.abstract.PAEType*[bytes]

Represents a raw byte string, encoded as the identity.

write(*value: bytes*, *stream: IO*) → int
Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → bytes
Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

class python_pae.pae_types.**PAEString**(**args*, ***kwds*)
Bases: *python_pae.abstract.PAEType*[str]

Represents a text string, encoded in UTF-8.

write(*value: str*, *stream: IO*) → int
Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → str
Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

class python_pae.pae_types.**PAENumberType**(*value*)
Bases: *python_pae.abstract.PAEType*[int]

Encodes various unsigned integer types. All are encoded in little-endian order.

property constant_length

unpack(*packed: bytes*)

pack(*value: int*)

write(*value: int*, *stream: IO*) → int
Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → int
Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

class python_pae.pae_types.**PAEHomogeneousList**(*child_type: python_pae.abstract.PAEType[python_pae.pae_types.S], settings: python_pae.encode.PAEListSettings = PAEListSettings(size_type=<uint64 (ULLONG)>, length_type=None, prefix_if_constant=False)*)
Bases: *python_pae.abstract.PAEType*[List[python_pae.pae_types.S]]

Homogeneous list of length-prefixed items.

Parameters

- **child_type** – The type of the list's elements.
- **settings** – Encoding settings for the list.

write(*value: List[python_pae.pae_types.S]*, *stream: IO*) → int
Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → List[python_pae.pae_types.S]
Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

class python_pae.pae_types.**PAEHeterogeneousList**(*component_types: List[python_pae.abstract.PAEType], settings: python_pae.encode.PAEListSettings = PAEListSettings(size_type=<uint64 (ULLONG)>, length_type=None, prefix_if_constant=True)*)
Bases: *python_pae.abstract.PAEType*[list]

Heterogeneous, fixed-length list of length-prefixed items, or a tuple.

Parameters

- **component_types** – The list of types that appear as the list's components, in order.
- **settings** – Encoding settings for the list.

write(*value: list*, *stream: IO*) → int
Serialise and write a value to a stream, length prefix *not* included.

Parameters

- **value** – The value to write.
- **stream** – The stream to write to.

Returns The number of bytes written.

read(*stream: IO*, *length: int*) → list
Read a value from a stream, length prefix *not* included, and decode it.

Parameters

- **stream** – The stream to write to.
- **length** – The expected length of the content to be read.

Returns The decoded value.

python_pae.pae_types.**DEFAULT_HMG_LIST_SETTINGS = PAEListSettings(size_type=<uint64 (ULLONG)>, length_type=None, prefix_if_constant=False)**
Default list settings for homogeneous lists.

python_pae.pae_types.**DEFAULT_HTRG_LIST_SETTINGS = PAEListSettings(size_type=<uint64 (ULLONG)>, length_type=None, prefix_if_constant=True)**
Default list settings for heterogeneous lists.

1.1.2 Members

This is the main entry point for the PAE encoding/decoding API.

python_pae.**pae_encode**(*lst: List[bytes]*, *size_t: python_pae.number.PAENumberType = <uint64 (ULLONG)>*) → bytes
Encode a list of byte strings in PAE.

Note: By default, this function produces output that is compatible with PASETO PAE.

Parameters

- **lst** – A list of byte strings.
- **size_t** – Numeric type to use for the list's size and its members' length prefixes.

Returns The PAE-encoded list as a byte string.

python_pae.**pae_encode_multiple**(*value_type_pairs*, *size_t: python_pae.number.PAENumberType = <uint64 (ULLONG)>*) → bytes
Encode a list of multiple typed values in PAE.

Parameters

- **value_type_pairs** – A list of tuples of the form `(v, t)`, where `v` is a value, and `t` is a *PAEType* implementation for that value type.

- **size_t** – Numeric type to use for the list's size and its members' length prefixes.

Returns The PAE-encoded list as a byte string.

python_pae.**marshal**(*value: python_pae.encode.T*, *pae_type:* python_pae.abstract.PAEType*[python_pae.encode.T]*) → bytes

Serialise a value into bytes.

Parameters

- **value** – The value to be processed.
- **pae_type** – The *PAEType* that provides the serialisation logic.

Returns A byte string representing the value passed in.

python_pae.**unmarshal**(*packed: bytes*, *pae_type:* python_pae.abstract.PAEType*[python_pae.encode.T]*) → python_pae.encode.T

Decode a byte string back into a value. Inverse operation of *marshal()*.

Parameters

- **packed** – The byte string to be processed.
- **pae_type** – The *PAEType* that provides the deserialisation logic.

Returns A decoded value.

Raises ***python_pae.PAEDecodeError*** – if an error occurs in the decoding process.

class python_pae.**PAEListSettings**(*size_type: python_pae.number.PAENumberType = <uint64 (ULLONG)>*, *length_type: Optional[python_pae.number.PAENumberType] = None*, *prefix_if_constant: bool = True*)

Bases: `object`

List encoding settings. The defaults represent the PASETO version of PAE.

size_type: *python_pae.number.PAENumberType* = <uint64 (ULLONG)>

Numeric type to use for the list size.

Note: The default is a 64-bit integer for compatibility with PASETO PAE.

length_type: Optional[*python_pae.number.PAENumberType*] = None

Numeric type to use for the length prefixes of the list items.

Note: If unspecified, will be the same as *size_type*.

prefix_if_constant: bool = True

Flag toggling whether to apply the length prefix if the type being written or read is a fixed-width type. Defaults to `True`.

exception python_pae.**PAEDecodeError**

Bases: `ValueError`

Raised if an error occurs during PAE decoding.

CHAPTER

TWO

LICENSE

```
MIT License

Copyright (c) 2021 Matthias Valvekens

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

This minimal library (`python-pae`) offers an implementation of (a variant of) PASETO's pre-authentication encoding (PAE) scheme in pure Python.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

FOUR

LINKS

- Homepage
- Documentation
- PyPI

PYTHON MODULE INDEX

p

INDEX

S

U

W