pylibsnmp

Release 0.3

Abdul Zagirov

Jun 07, 2022

CONTENTS

pylibsnmp is a Python library for working with network devices via snmp.

Check out the *Usage* section for further information, including how to *Installation* the project.

Note: This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

To use pylibsnmp, first install it using pip:

```
(.venv) $ pip install pylibsnmp
```

1.2 API

pylibsnmp.device	Python module with NetDevice class used to create network devices with SNMP protocol enabled.
pylibsnmp.helpers	Python module with helper functions used in the project.

1.2.1 pylibsnmp.device

Python module with NetDevice class used to create network devices with SNMP protocol enabled.

SNMP v1 and v2 are supported.

Pass ip address, snmp community, port and version in order to be able to initialize the connection with the device.

IP address is required while snmp community, port and version are optional.

Classes

NetDevice([address, community, port, version])	Class for creating snmp enabled network devices.

class pylibsnmp.device.**NetDevice**(*address='127.0.0.1'*, *community='public'*, *port=161*, *version=2*)

Class for creating snmp enabled network devices.

__COEFFICIENT = 1000000

__VERSIONS = (1, 2)

__DEFAULT = {'ADDRESS': '127.0.0.1', 'COMMUNITY': 'public', 'PORT': 161, 'VERSION': 2}

__DELIMITERS = (':', '-', '.')

__init__(*address='127.0.0.1'*, *community='public'*, *port=161*, *version=2*) → None

Class constructor.

params:

address: {str} - device ip address
community: {str} - snmp community {default: "public"}
port: {int} - snmp port {default: 161}
version: {int} - snmp version {default: 2}

__str__() → str

Returns information about object in human readable format.

property address: str

IP address

property community: str

SNMP community

property port: int

SNMP port

property version: int

SNMP version

property autoupdate: bool

Enable/disable device information autoupdate

property contact: str

Contact

property description: str

Description

property indexes: List[int]

List of interface numbers

property location: str

Location

property name: str

Name

property number: int

property types: List[str]

List of interface types

property updatetime: int

Autoupdate interval

property uptime: *str*

Uptime

connect() → bool

Initiates connection with the device using parameters passed in constructor.

disconnect() → None

Correctly drops connection with the device.

get_if_admin_status(*port: int*) → str

The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. When a managed system initializes, all interfaces start with ifAdminStatus in the down(2) state. As a result of either explicit management action or per configuration information retained by the managed system, ifAdminStatus is then changed to either the up(1) or testing(3) states (or remains in the down(2) state).

get_if_description(*port: int*) → str

A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the interface hardware/software.

get_if_in_octets(*port: int*) → str

The total number of octets received on the interface, including framing characters.

get_if_in_broadcast(*port: int*) → str

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a broadcast address at this sub-layer. This object is a 64-bit version of ifInBroadcastPkts.

get_if_in_errors(*port: int*) → str

For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character-oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol.

get_if_in_discards(*port: int*) → str

The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

get_if_in_multicast(*port: int*) → str

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast address at this sub-layer. For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifInMulticastPkts.

get_if_in_non_unicast(*port: int*) → str

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast or broadcast address at this sub-layer.

get_if_in_unicast(*port: int*) → str

The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer.

get_if_last_change(*port: int*) → str

The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a zero value.

get_if_mtu(*port: int*) → str

The size of the largest packet which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

get_if_oper_status(*port: int*) → str

The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. If ifAdminStatus is down(2) then ifOperStatus should be down(2). If ifAdminStatus is changed to up(1) then ifOperStatus should change to up(1) if the interface is ready to transmit and receive network traffic; it should change to dormant(5) if the interface is waiting for external actions (such as a serial line waiting for an incoming connection); it should remain in the down(2) state if and only if there is a fault that prevents it from going to the up(1) state; it should remain in the notPresent(6) state if the interface has missing (typically, hardware) components.

get_if_out_octets(*port: int*) → str

The total number of octets transmitted out of the interface, including framing characters.

get_if_out_broadcast(*port: int*) → str

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast address at this sub-layer, including those that were discarded or not sent. This object is a 64-bit version of ifOutBroadcastPkts.

get_if_out_errors(*port: int*) → str

For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors.

get_if_out_discards(*port: int*) → str

The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

get_if_out_multicast(*port: int*) → str

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. For a MAC layer protocol, this includes both Group and Functional addresses. This object is a 64-bit version of ifOutMulticastPkts.

get_if_out_non_unicast(*port: int*) → str

The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.

get_if_out_unicast(*port: int*) → str

The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.

get_if_phys_address(*port: int*, *delimiter: str = ':'*) → str

The interface's address at its protocol sub-layer. For example, for an 802.x interface, this object normally contains a MAC address. The interface's media-specific MIB must define the bit and byte ordering and the format of the value of this object. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

get_if_speed(*port: int*) → str

An estimate of the interface's current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal

bandwidth. If the bandwidth of the interface is greater than the maximum value reportable by this object then this object should report its maximum value (4,294,967,295) and ifHighSpeed must be used to report the interace's speed. For a sub-layer which has no concept of bandwidth, this object should be zero.

get_if_type(*port: int*) → str

The type of interface. Additional values for ifType are assigned by the Internet Assigned Numbers Authority (IANA), through updating the syntax of the IANAifType textual convention.

get_if_unknown_protos(*port: int*) → str

For packet-oriented interfaces, the number of packets received via the interface which were discarded because of an unknown or unsupported protocol. For character-oriented or fixed-length interfaces that support protocol multiplexing the number of transmission units received via the interface which were discarded because of an unknown or unsupported protocol. For any interface that does not support protocol multiplexing, this counter will always be 0.

__change_autoupdate()

Function activates/deactivates autoupdate functionality.

__get_contact() → str

The textual identification of the contact person for this managed node, together with information on how to contact this person.

__get_description() → str

A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters.

__get_if_data(*snmp_oid: str*, *if_port: int*, *error_msg: str*) → str

Function used in receiving interface related information.

__get_if_indexes() → List[int]

A unique value, greater than zero, for each interface. It is recommended that values are assigned contiguously starting from 1. The value for each interface sub-layer must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

__get_if_number() → str

The number of network interfaces (regardless of their current state) present on this system.

__get_if_types() → List[str]

Returns list of interface types.

__get_location() → str

The physical location of this node (e.g., "telephone closet, 3rd floor").

__get_name() → str

An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name.

__get_sys_data(*snmp_oid: str*, *error_msg: str*) → str

Function used in receiving device related information.

__get_uptime() → str

The time (in hundredths of a second) since the network management portion of the system was last re-initialized.

__populate() → None

Populates device fields with necessary data.

1.2.2 pylibsnmp.helpers

Python module with helper functions used in the project.

Functions

get_bits(octets)	Converts octets to bits.
get_mac_from_octets(octets[, delimiter])	Converts octets to mac address.
get_speed(bits)	Converts bits to Kbits/s, Mbits/s or Gbits/s according to the bits count.
get_unit(bits)	Returns unit type according to the bits count.
is_ip_address(address)	Checks the string to be in ip address format.
is_port_number(port)	Checks whether port number argument has an appropriate value.

Classes

SetInterval(func, sec)	Class for creating python alternative to JavaScript setInterval function.

class pylibsnmp.helpers.**SetInterval**(*func: Callable*, *sec: int*)

Class for creating python alternative to JavaScript setInterval function.

__init__(*func: Callable*, *sec: int*) → None

Class constructor.

params:

func: {Callable} - function to execute
sec: {int} - interval in seconds to execute func

cancel()

Cancels Timer object in order for the application to end correctly.

pylibsnmp.helpers.**get_bits**(*octets: int*) → int

Converts octets to bits.

An octet is really just a fancy name for a "byte". So if you multiply this number by 8 you get bits.

pylibsnmp.helpers.**get_mac_from_octets**(*octets: str*, *delimiter: str = ':'*) → str

Converts octets to mac address.

When requesting physical address of the device using snmp responce comes in the format of octets.

In order to convert it to mac address: | - get list of ascii codes of octets | - convert it to bytearray | - convert it to hex format

pylibsnmp.helpers.**get_speed**(*bits: int*) → int

Converts bits to Kbits/s, Mbits/s or Gbits/s according to the bits count.

pylibsnmp.helpers.**get_unit**(*bits: int*) → str

Returns unit type according to the bits count.

pylibsnmp.helpers.**is_ip_address**(*address: str*) → bool

Checks the string to be in ip address format.

IP address have to: | - have four octets | - each octet must be from 0 to 255 | - each octet must be in digital format

pylibsnmp.helpers.**is_port_number**(*port: int*) → bool

Checks whether port number argument has an appropriate value.

Port number has to be in the range of 1 and 65535.

PYTHON MODULE INDEX

p

INDEX

Symbols

A

C

D

G

I

L

M

N

P

S

T

U

V