Easy SNMP

Jul 16, 2022

Contents

A blazingly fast and Pythonic SNMP library based on the official Net-SNMP bindings

CHAPTER 1

Introduction

This is a fork of the official Net-SNMP Python Bindings but attempts to bring a more Pythonic interface to the library. Check out the Net-SNMP website for more information about SNMP.

This module provides a full featured SNMP client API supporting all dialects of the SNMP protocol.

CHAPTER 2

Why Another Library?

- The original Net-SNMP Python library is a great starting point but is quite un-Pythonic and lacks proper unit tests and documentation.
- PySNMP is written in pure Python and therefore has a huge performance hit. In some brief tests, I estimate that both the Net-SNMP Python bindings and Easy SNMP are more than 4 times faster. Further to this, PySNMP has an even less Pythonic interface than the official Net-SNMP bindings.
- Many other libraries like Snimpy are sadly based on PySNMP and so they suffer the same performance penalty.

CHAPTER 3

Installation

EasySNMP has been tested and is supported on systems running Net-SNMP 5.7.x and newer. All non-EOL versions of Python 3 are fully supported, with 2.7 and recent EOL versions of Python 3 receiving partial support.

If your OS ships with a supported version of Net-SNMP, then you can install it without compiling it via your package manager:

On RHEL / CentOS systems:

```
sudo yum install net-snmp-devel
```

On Debian / Ubuntu systems:

```
sudo apt-get install libsnmp-dev snmp-mibs-downloader
```

On macOS systems:

```
brew install net-snmp
```

If your OS doesn't ship with Net-SNMP 5.7.x or newer, please follow instructions provided on the Net-SNMP install page to build and install Net-SNMP on your system.

You'll also need to ensure that you have the following packages installed so that Easy SNMP installs correctly:

On RHEL / CentOS systems:

```
sudo yum install gcc python-devel
```

On Debian / Ubuntu systems:

```
sudo apt-get install gcc python-dev
```

On macOS systems:

```
brew install gcc
```

Install Easy SNMP via pip as follows:

```
pip install easysnmp
```

CHAPTER 4

Quick Start

There are primarily two ways you can use the Easy SNMP library.

The first is with the use of a Session object which is most suitable when you are planning on requesting multiple pieces of SNMP data from a source.

```
from easysnmp import Session

# Create an SNMP session to be used for all our requests
session = Session(hostname='localhost', community='public', version=2)

# You may retrieve an individual OID using an SNMP GET
location = session.get('sysLocation.0')

# You may also specify the OID as a tuple (name, index)
# Note: the index is specified as a string as it can be of other types than
# just a regular integer
contact = session.get(('sysContact', '0'))

# And of course, you may use the numeric OID too
description = session.get('.1.3.6.1.2.1.1.1.0')

# Set a variable using an SNMP SET
session.set('sysLocation.0', 'The SNMP Lab')

# Perform an SNMP walk
system_items = session.walk('system')

# Each returned item can be used normally as its related type (str or int)
# but also has several extended attributes with SNMP-specific information
for item in system_items:
    print '{oid}.{oid_index} {snmp_type} = {value}'.format(
        oid=item.oid,
        oid_index=item.oid_index,
        snmp_type=item.snmp_type,
```

(continues on next page)

(continued from previous page)

```
        value=item.value
    )
```

You may also use Easy SNMP via its simple interface which is intended for one-off operations where you wish to specify all details in the request:

```
from easysnmp import snmp_get, snmp_set, snmp_walk

# Grab a single piece of information using an SNMP GET
snmp_get('sysDescr.0', hostname='localhost', community='public', version=1)

# Perform an SNMP SET to update data
snmp_set(
    'sysLocation.0', 'My Cool Place',
    hostname='localhost', community='public', version=1
)

# Perform an SNMP walk
snmp_walk('system', hostname='localhost', community='public', version=1)
```

CHAPTER 5

API

5.1 Session API

class easysnmp.**Session**(*hostname='localhost'*, *version=3*, *community='public'*, *timeout=1*, *retries=3*, *remote_port=0*, *local_port=0*, *security_level='no_auth_or_privacy'*, *security_username='initial'*, *privacy_protocol='DEFAULT'*, *privacy_password=''*, *auth_protocol='DEFAULT'*, *auth_password=''*, *context_engine_id=''*, *security_engine_id=''*, *context=''*, *engine_boots=0*, *engine_time=0*, *our_identity=''*, *their_identity=''*, *their_hostname=''*, *trust_cert=''*, *use_long_names=False*, *use_numeric=False*, *use_sprint_value=False*, *use_enums=False*, *best_guess=0*, *retry_no_such=False*, *abort_on_nonexistent=False*)

A Net-SNMP session which may be setup once and then used to query and manipulate SNMP data.

Note: This class transparently uses `interface` to create a session instance from the Net-SNMP library. Most variable values are not synchronized between the `Session` and `interface`. If you intend to make changes to the `Session` instead of creating a new one, you must manually call the *update_session()* method

Parameters

- **hostname** – hostname or IP address of SNMP agent
- **version** – the SNMP version to use; 1, 2 (equivalent to 2c) or 3
- **community** – SNMP community string (used for both R/W) (v1 & v2)
- **timeout** – seconds before retry
- **retries** – retries before failure
- **remote_port** – allow remote UDP port to be overridden (this will communicate on port 161 at its default setting)

- **local_port** – allow overriding of the local SNMP port
- **security_level** – security level (no_auth_or_privacy, auth_without_privacy or auth_with_privacy) (v3)
- **security_username** – security name (v3)
- **privacy_protocol** – privacy protocol (v3)
- **privacy_password** – privacy passphrase (v3)
- **auth_protocol** – authentication protocol (MD5 or SHA) (v3)
- **auth_password** – authentication passphrase (v3)
- **context_engine_id** – context engine ID, will be probed if not supplied (v3)
- **security_engine_id** – security engine ID, will be probed if not supplied (v3)
- **context** – context name (v3)
- **engine_boots** – the number of times the SNMP engine has re-booted/re-initialized since SNMP engine ID was last configured (v3)
- **engine_time** – the number of seconds since the engine_boots counter was last incremented (v3)
- **our_identity** – the fingerprint or file name for the local X.509 certificate to use for our identity (run net-snmp-cert to create and manage certificates) (v3 TLS / DTLS)
- **their_identity** – the fingerprint or file name for the local X.509 certificate to use for their identity (v3 TLS / DTLS)
- **their_hostname** – their hostname to expect; either their_hostname or a trusted certificate plus a hostname is needed to validate the server is the proper server (v3 TLS / DTLS)
- **trust_cert** – a trusted certificate to use for validating certificates; typically this would be a CA certificate (v3 TLS / DTLS)
- **use_long_names** – set to True to have <tags> for getnext methods generated preferring longer Mib name convention (e.g., system.sysDescr vs just sysDescr)
- **use_numeric** – set to True to have <tags> returned by the get methods untranslated (i.e. dotted-decimal). Setting the use_long_names value for the session is highly recommended
- **use_sprint_value** – set to True to have return values for get and getnext methods formatted with the libraries sprint_value function. This will result in certain data types being returned in non-canonical format Note: values returned with this option set may not be appropriate for set operations
- **use_enums** – set to True to have integer return values converted to enumeration identifiers if possible, these values will also be acceptable when supplied to set operations
- **best_guess** – this setting controls how oids are parsed; setting to 0 causes a regular lookup. setting to 1 causes a regular expression match (defined as -Ib in snmpcmd); setting to 2 causes a random access lookup (defined as -IR in snmpcmd).
- **retry_no_such** – if enabled NOSUCH errors in get pdus will be repaired, removing the SNMP variable in error, and resent; undef will be returned for all NOSUCH SNMP variables, when set to False this feature is disabled and the entire get request will fail on any NOSUCH error (applies to v1 only)
- **abort_on_nonexistent** – raise an exception if no object or no instance is found for the given oid and oid index

bulkwalk(*oids='.1.3.6.1.2.1'*, *non_repeaters=0*, *max_repetitions=10*)
Uses SNMP GETBULK operation using the prepared session to automatically retrieve multiple pieces of information in an OID

Parameters **oids** – you may pass in a single item (multiple values currently experimental) which may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))

Returns a list of SNMPVariable objects containing the values that were retrieved via SNMP

get(*oids*)
Perform an SNMP GET operation using the prepared session to retrieve a particular piece of information.

Parameters **oids** – you may pass in a list of OIDs or single item; each item may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))

Returns an SNMPVariable object containing the value that was retrieved or a list of objects when you send in a list of OIDs

get_bulk(*oids*, *non_repeaters=0*, *max_repetitions=10*)
Performs a bulk SNMP GET operation using the prepared session to retrieve multiple pieces of information in a single packet.

Parameters

- **oids** – you may pass in a list of OIDs or single item; each item may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))
- **non_repeaters** – the number of objects that are only expected to return a single GETNEXT instance, not multiple instances
- **max_repetitions** – the number of objects that should be returned for all the repeating OIDs

Returns a list of SNMPVariable objects containing the values that were retrieved via SNMP

get_next(*oids*)
Uses an SNMP GETNEXT operation using the prepared session to retrieve the next variable after the chosen item.

Parameters **oids** – you may pass in a list of OIDs or single item; each item may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))

Returns an SNMPVariable object containing the value that was retrieved or a list of objects when you send in a list of OIDs

set(*oid*, *value*, *snmp_type=None*)
Perform an SNMP SET operation using the prepared session.

Parameters

- **oid** – the OID that you wish to set which may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))
- **value** – the value to set the OID to
- **snmp_type** – if a numeric OID is used and the object is not in the parsed MIB, a type must be explicitly supplied

Returns a boolean indicating the success of the operation

set_multiple(*oid_values*)
Perform an SNMP SET operation on multiple OIDs with multiple values using the prepared session.

Parameters **oid_values** – a list of tuples whereby each tuple contains a (oid, value) or an (oid, value, snmp_type)

Returns a list of SNMPVariable objects containing the values that were retrieved via SNMP

update_session(***kwargs*)
(Re)creates the underlying Net-SNMP session object.

While it is recommended to create a new `Session` instance instead, this method has been added for your convenience in case you really need it (we've mis-typed the community string before in our interactive sessions and totally understand your pain).

Keywords passed to the method will be assigned to the instance if they match existing attribute names. A warning will be emitted if something is passed that does not match anything that already exists.

Listing 1: Example usage

```
s = Session(version=2, community='readonly', hostname='localhost')
# Whoops, wrong hostname and community string. Let's change that
s.update_session(community='readwrite', hostname'remotehost')
# Actually I need to use version 1
s.version = 1
s.update_session()
```

walk(*oids='.1.3.6.1.2.1'*)
Uses SNMP GETNEXT operation using the prepared session to automatically retrieve multiple pieces of information in an OID.

Parameters **oids** – you may pass in a single item (multiple values currently experimental) which may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))

Returns a list of SNMPVariable objects containing the values that were retrieved via SNMP

5.2 Easy API

easysnmp.**snmp_get**(*oids*, ***session_kargs*)
Perform an SNMP GET operation to retrieve a particular piece of information.

Parameters

- **oids** – you may pass in a list of OIDs or single item; each item may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))
- **session_kargs** – keyword arguments which will be sent used when constructing the session for this operation; all parameters in the Session class are supported

easysnmp.**snmp_set**(*oid*, *value*, *type=None*, ***session_kargs*)
Perform an SNMP SET operation to update a particular piece of information.

Parameters

- **oid** – the OID that you wish to set which may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))

- **value** – the value to set the OID to
- **snmp_type** – if a numeric OID is used and the object is not in the parsed MIB, a type must be explicitly supplied
- **session_kargs** – keyword arguments which will be sent used when constructing the session for this operation; all parameters in the Session class are supported

easysnmp.**snmp_set_multiple**(*oid_values*, ***session_kargs*)
Perform multiple SNMP SET operations to update various pieces of information at the same time.

Parameters

- **oid_values** – a list of tuples whereby each tuple contains a (oid, value) or an (oid, value, snmp_type)
- **session_kargs** – keyword arguments which will be sent used when constructing the session for this operation; all parameters in the Session class are supported

easysnmp.**snmp_get_next**(*oids*, ***session_kargs*)
Uses an SNMP GETNEXT operation to retrieve the next variable after the chosen item.

Parameters

- **oids** – you may pass in a list of OIDs or single item; each item may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))
- **session_kargs** – keyword arguments which will be sent used when constructing the session for this operation; all parameters in the Session class are supported

easysnmp.**snmp_get_bulk**(*oids*, *non_repeaters=0*, *max_repetitions=10*, ***session_kargs*)
Performs a bulk SNMP GET operation to retrieve multiple pieces of information in a single packet.

Parameters

- **oids** – you may pass in a list of OIDs or single item; each item may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))
- **non_repeaters** – the number of objects that are only expected to return a single GETNEXT instance, not multiple instances
- **max_repetitions** – the number of objects that should be returned for all the repeating OIDs
- **session_kargs** – keyword arguments which will be sent used when constructing the session for this operation; all parameters in the Session class are supported

easysnmp.**snmp_walk**(*oids='.1.3.6.1.2.1'*, ***session_kargs*)
Uses SNMP GETNEXT operation to automatically retrieve multiple pieces of information in an OID for you.

Parameters

- **oids** – you may pass in a single item (multiple values currently experimental) which may be a string representing the entire OID (e.g. 'sysDescr.0') or may be a tuple containing the name as its first item and index as its second (e.g. ('sysDescr', 0))
- **session_kargs** – keyword arguments which will be sent used when constructing the session for this operation; all parameters in the Session class are supported

easysnmp.**snmp_bulkwalk**(*oids='.1.3.6.1.2.1'*, *non_repeaters=0*, *max_repetitions=10*, ***session_kargs*)
Uses SNMP GETBULK operation using the prepared session to automatically retrieve multiple pieces of information in an OID

Parameters

- **oids** – you may pass in a single item * string representing the entire OID (e.g. 'sysDescr.0') * tuple (name, index) (e.g. ('sysDescr', 0)) * list of OIDs
- **non_repeaters** – the number of objects that are only expected to return a single GET-NEXT instance, not multiple instances
- **max_repetitions** – the number of objects that should be returned for all the repeating OIDs

Returns a list of SNMPVariable objects containing the values that were retrieved via SNMP

5.3 Exceptions

class easysnmp.**EasySNMPError**
The base Easy SNMP exception which covers all exceptions raised.

class easysnmp.**EasySNMPConnectionError**
Indicates a problem connecting to the remote host.

class easysnmp.**EasySNMPTimeoutError**
Raised when an SNMP request times out.

class easysnmp.**EasySNMPUnknownObjectIDError**
Raised when a nonexistent OID is requested.

class easysnmp.**EasySNMPNoSuchObjectError**
Raised when an OID is requested which may have some form of existence but is an invalid object name.

class easysnmp.**EasySNMPNoSuchInstanceError**
Raised when a particular OID index requested from Net-SNMP doesn't exist.

class easysnmp.**EasySNMPUndeterminedTypeError**
Raised when the type cannot be determined when setting the value of an OID.

Index