python-netsnmpagent

Writing net-snmp AgentX subagents in Python

Pieter Hollants
pieter@hollants.com / @pfhllnts

Linux System Engineer /
„Hardware Competence Center“ Dude
DFS Deutsche Flugsicherung GmbH

FOSDEM '14
Brussels 1 & 2 February 2014
This work sponsored by...

DFS Deutsche Flugsicherung GmbH

- German Air Traffic Control
- 100% government owned (yet)
- 5000 employees, 20 locations
- We produce safety
Why the fuzz?

- REST, node.js et. al – SNMP not particularly sexy?
- But: $employer uses monitoring that speaks SNMP
- Need to integrate centralized hardware monitoring (detect fan failures etc.) among a hardware zoo
- Server vendors with own MIBs, clients need extra work (coretemp, lm_sensors, smartctl)
- >> Define and implement our own Hardware-MIB (for better or worse)
SNMP / net-snmp

- SNMP: Simple Network Monitoring Protocol
  - Versions 1 (RFC 1157), 2c (1901), 3 (2571)
- net-snmp: dominant toolkit to implement SNMP
  - applications (snmpwalk etc.) and libraries
  - snmpd: master agent
  - extensible: dlopen() modules, smux, AgentX
  - C API, mib2c template generator for own agents
  - Agent architecture beyond scope of talk
Python and SNMP

• Why Python? $exboss told me so...
• net-snmp comes with a Python module „netsnmp“
  – 73KB C code that abstracts C api
  – Synchronous client code only
• Idea: access C API from Python directly with ctypes module, imitating AgentX subagents written in C
• Existing python-agentx module on Sourceforge
  – Design issues
  – Orphaned?
Hello python-netsnmpagent

- Two source files
  - netsnmpapi.py (ctypes stuff), 13KB
  - netsnmpagent.py (classes), 33KB (21KB)
- Extensively commented
- Example MIBs/agents included
- Whaddya mean, „coding style“?
- Tested with net-snmp 5.4.2 (SLES11 SP2), 5.4.3 (Ubuntu 12.04 LTS), 5.7.1 (openSUSE 12.x)
simpleInteger OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"A read-write, unsigned, 32-bits integer value."
::= { simpleScalars 1 }

Yes, tables also possible...

A simple scalar 32-bit signed value. Tables are more complex.

translates to something like
.1.3.6.1.2.1.74.1.30187.1.1.1
import netsnmpagent

try:
    agent = netsnmpagent.netsnmpAgent(
        AgentName = "SimpleAgent",
        MIBFiles = "[...]/SIMPLE-MIB.txt"
    )
except netsnmpagent.netsnmpAgentException as e:
    # handle exception
```
simpleInteger = agent.Integer32(
    oidstr = "SIMPLE-MIB::simpleInteger"
)

try:
    agent.start()
except netsnmpagent.netsnmpAgentException as e:
    # handle exception
```

Sort of a class factory. Returns a Python object handling a SNMP object of type "Integer32".

Having objects declared in the MIB alone is not enough – two subagents might use one MIB. So explicitly register what OID this object handles.

Registrations done, connect to snmpd.
while (loop):
    agent.check_and_process()
    simpleInteger.update(simpleInteger.value() * 2)
    print "val: {0}".simpleInteger.value()

Net-snmp API call: block until we have work

- More complete examples (eg. with tables) in the source distribution
- Naturally, a real life agent would be more complex...
  (DFS HW-Agent: ~120KB, ~3000 lines)
To do

• Notifications/traps
• API documentation (doh...)
• Unit tests
  – After all „we produce safety“, right?
Thank you!

- Source: https://github.com/pief/python-netsnmpagent
- PyPI page: https://pypi.python.org/pypi/netsnmpagent
- Binary packages for SUSE: https://build.opensuse.org/package/show?package=python-netsnmpagent&project=home%3Apfhllnts
- Net-SNMP: http://www.net-snmp.org