Lessons Learned from 30 Years of Net-SNMP

Wes Hardaker

November 17, 2024

1 Simplified History

This story is not about *SNMP* [**rfc3410**], even though it's in the title. It's about my history in starting in network manager and what became Net-SNMP, and why it became popular.

The *TL;DR*? **It's about empowering your users with simplicity.**

Note: this document is entirely from the perspective of an operator – not from the prospective of a network management tool vendor.

2 Net-SNMP's Origin Story

Thirty years ago (give or take) I was working for UCDavis using a popular network management platform called "HP OpenView" (HPOV). It had a wonderful *traffic light* feature: if something was green you could assume it was ok, but if something was yellow or red – it had issues that needed fixing. Our goal was to add as much problem detection as we could to this traffic light display system.

So, my first network management job was to teach HPOV about a few particular local problems we constantly had. Examples included NFS mounts that were failing, disks that were full, or certain unix daemons that weren't running. The devil was in the details: not every machine was running NFS, different disks had different critical threshold events, and not every machine was running every type of daemon. This is a real pain to encode on the management side of things.

So my solution involved learning what the *Simple Network Management Platform* (SNMP) was, because that's what I was told HPOV spoke the best. I quickly learned that the *Simple* in the SNMP title was related to the format of the packet, not how easy it was to use. And this was the problem, as I needed something that was simple and easy to use *as an administrator*. And rather than have the management station figure out the individual characteristics of each machine – I wanted an SNMP agent that a very basic configuration file on each machine that simply reported a flag: *thing X is ok? 1 or 0.* **This** is what SNMP was good at – reporting *simple values in simple ways.* If the error flag was 1, color the node and parent node red. Done.

So I started out porting the *CMU-SNMP* library and agent to our SunOS, Ultrix and HPUX systems. CMU-SNMP was the open source reference toolkit at the time, but its code was very specific to a particular version of SunOS (that we weren't using). So I simply ripped the code out that we didn't need (ok, technically *#ifdef/ed it out). Then I created a new MIB table with a column containing my needed /1s or 0s*, along with a title keyword column, and an error string colmun. And I made the configuration file be very very simple:

```
disk / 10%
proc mountd
```

And the results of this could be looked at with human eyes just by running a simple SNMP walk for each table. Here's an example output of the process monitoring table:

```
$ ./snmpwalk localhost prTable
UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prNames.1 = STRING: mountd
UCD-SNMP-MIB::prMin.1 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 0
UCD-SNMP-MIB::prCount.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: error(1)
UCD-SNMP-MIB::prErrMessage.1 = STRING: No mountd process running
```

The results can be quickly read by just about anyone, and HPOV simply watched the *prErrorFlag* value and ignored all the other columns. Done.

3 Becoming a Popular Open Source Package

So I released it, as we did back in the days before *GitHub*: I emailed a link to the HPOV mailing list, pointing to a patch to the base *CMU-SNMP* software. Within a few days it had achieved a few hundred downloads, which wasn't bad for a first release sent as a patch in the 1990s. Over the next few years, I added more and more simple monitoring features (load average, custom shell commands to run, etc). The popularity grew, and eventually I re-branded the code to become *UCD-SNMP* and released it as a packaged tar-ball. (I even switched from using *RCS* to *CVS* for version control.)

Many years later I left UCDavis but continued maintaining the project, again rebranding it to *Net-SNMP* after it attracted a decent core set of all-volunteer developers. The project moved to SourceForge where it attracted even more attention and even more contributed bug reports, patches and feature suggestions. Today, it is available as an easily installable package on pretty much every operating system but MS Windows (and our pre-built download for windows continues to be a popular add-on that can replace the Windows default SNMP agent)

So, **WHY** was this **SIMPLE** approach so successful?

4 Empower operators: Make simple things simple

Because I was developing the toolkit while also working as a network operator, I understood the importance of not having the time to spend tinkering with large, complex configuration systems to get something up and running. I couldn't tweak HPOV itself easily either (it was fairly customizable, but it wasn't "easy"). So I stuck to my simple approach: *make it very easy to quickly monitor new problems.*

Unfortunately, every operating environment is a unique snowflake. Developers might wish that one management platform could easily manage all the devices in a network, but everyone has slightly different hardware deployments, slightly different software deployments, slightly different firmware and software versions, etc.

Now, there are definitely limitations. I actually tried to make Net-SNMP easier to use for configuration, for which the SNMP protocol was never known to be good at. Unfortunately although we added a lot of features to make it easier, it still never became easy. Hence netconf was born in hopes to make an easier to use language.

5 Today's landscape

If we look at today's most successful management technologies. What do they have in common? Simplicity of use!

Prometheus/Alertmanager: allows you to run your own shell scripts to create data that is automatically consumed, graphed and made alert-able.

Ansible: Do these things in this order and only interrupt me if it brakes. Repeatable, simple and easy to follow configuration directives.

Web-APIs: Can I get the data with *curl*? Win!

JSON: Can I parse the results with 2 lines of code in every language? Win!

YAML: Can I understand a config file without actually fully understanding the encoding used? (yes you can still get in trouble here).

Docker: set up this exact environment with no deviations please. I wouldn't call it "simple" but it is at least repeatable.

SNMP: still around, for better or worse. Left as an exercises to the reader.

I use pretty much all of these technologies over every other option because they do not require a specialized degree or a lot of cash to get them up and running. They just work and do not consume extensive time, which is critical as an operator.

5.1 What doesn't work well?

That's easy: anything that's not simple. Yes, large relational or nested data models exist in many forms (MIBs, PIBs, YANGs,). But the instant you need to do table joins, or deep dives into structures you've just lost the quick-get-things-done-today operator. Large complex models likely are more useful to full network management applications, as tool vendors can afford significantly more development hours than a network operational team can.

6 Conclusion

So what can we learn from this? I've hinted at, but will now state more clearly: there are [at least] really two different audiences for standardized network management protocols: network management toolkit vendors, and the operators that actually run the systems. They have different requirements with respect to the needed simplicity of protocol solutions. We need to enable success for both.