Cellular Network Infrastructure - Feature #1975

Redesigned configuration management / MIB

03/09/2017 01:05 AM - laforge

Status:	New	**Start date:**	03/09/2017
Priority:	Low	**Due date:**	
Assignee:	laforge	**% Done:**	0%
Category:			
Target version:			
Spec Reference:			

Description

Using zebra/quagga VTY code was a great idea back then and has served us nicely in all those years. I like the general style of the command-line based interface with its various nodes, the strict syntax checking, the tab completion and the interactive context-sensivive help.

However, it feels a bit 20ieth-century-ish to have to manually write code to parse and to save the respective values. This is not a productive way to spend our development resources, and it is error prone. The "save" can be forgotten, resulting in non-saveable config parameters. The save can store values that the "parse" function will not be able to read again, etc.

Furthermore, the VTY is inherently a human user interface, not intended for programmatic consumption. For programmatic access, we have developed the control interface. In reality though, only 1% of the parameters available in the VTY are exported via the control interface. And rather than adding all the missing bits and pieces with hand-crafted code to the control interface, we should have a generic way to define a new configuration value, and that value should then be automatically parse- and storable via VTY as well as a programmatic interface.

The next "problem' is that the current telnet connections live within the process of the application. This means that a user can effectively "stall" the main application by performing I/O intensive operation on the VTY, or even on as many VTY/telnet sessions as he wants. There shouldn't be any need for this, at least not for something as mundane as performing configuration changes. Of course the VTY has different use cases such as runtime introspection of system state as well as logging. But still.

Yet another concern is "VTY and telnet port proliferation". Particularly with the conversion from NITB to BSC+MSC+HLR, we are yet again getting more network elements with their own telnet ports. Remembering the port numbers is clumsy. It would be more convenient if a single command line interface could provide access to the configuration and the state of multiple different processes / network elements.

Last, but not least, the current implementation is fixed to telnet, without any form of authentication, and without a path to migrate e.g. to something like SSL or SSH. I don't think it is a major concern (you can always SSH to the system and then telnet locally).

So what I had in mind for quite some time (actually since netconf 1.2 about one year ago), is to have some kind of an external "VTY/MIB daemon" (or even separate daemons for each) which maintains a hierarchical database of configuration values. The MIB deamon simply offers an API (via client library) to GET or SET the individual values, or to NOTIFY an application about a changed value. This API is both used by the actual Osmocom programs to obtain their configuration (and obtain updates to it during runtime), as well as by the "VTY daemon" providing interactive shell access to it. Finally, other external applications could use the same interface/client library to do the same. A proxy to SNMP or REST interfaces can be imagined, for even more interfacing with the outside world.

What I don't like about many existing MIBs I've used (as a sysadmin/user, not a developer) is their simple type system. You can often specify any random value to such a MIB, even one that is completely useless. A simple INT or STRING type is not sufficient, we really want the ability to have ENUM types, to have integers with ranges, etc. - just like we have in the VTY.

Also, the MIBs I've used typically are nothing more than a hierarchical key-value store. They do not contain the information required for interactive, context-sensitive help needed for the "VTY" feel :(This is btw also the problem I have with OpenWRT's uci: It just has keys and values, with no way to constrain those values to something that's reasonable to the given program/context that is being configured, and there's no help or other information associated with those keys and values.

So what I would like to see in this context, is some way to have a machine-parseable description of a "MIB/config item", together with syntax, ranges, enum values, help texts, etc (ideally in the C source code, possibly in comments?) which is used by some kind of MIB compiler to build up the hierarchical structure of the MIB and all of its keys as well as their permitted values and syntax reference. This information is then available to the VTY/telnet connections, irrespective of whether a given application [using those values] is running at all.

Once an application starts, it can query for the configuration values it is interested in and populate its internal data structures from it.

Ideally, one would even be able to generate a 'struct' from the MIB compiler, so that there is no need to explicitly call a "get" function on each single configuration value, but one simply gets a C-language 'struct' with all the values filled in by the client library.

As stated above, there should be some way how he MIB can notify applications about changes to certain nodes in the MIB tree, so the application[s] can react to that. I don't have a clear picture yet how transaction logic (like changing multiple values and then committing them at once) would work, or whether applications should even be able to reject/revert a modification?

In either case, I just wanted to share my thoughts on this. I know it sounds rather complex, but I think the investment in some kind of unified configuration database system would save us of a lot of boilerplate code and subtle bugs and inconsistencies.

History

#1 - 09/17/2019 11:12 AM - msuraev

One readily available option would be https://github.com/clicon/clicon - at least from what I can see in http://www.clicon.org/ it matches both "netconf" и "separate daemon" bits.

#2 - 09/17/2019 05:51 PM - laforge

Hi Max,

On Tue, Sep 17, 2019 at 11:12:03AM +0000, msuraev [REDMINE] wrote:

> One readily available option would be https://github.com/clicon/clicon - at least from what I can see in http://www.clicon.org/ it matches both "netconf" и "separate daemon" bits.

Indeed, that's one of the two FOSS options I could find (don't remember the other one, probably it was just called netconfd?). What I would be courious is to find some non-trivial applications that use those respective systems so we can see how an application using the related configuration system looks from the source code side, as well as can play with it for understanding it better.

#3 - 02/16/2021 05:59 PM - laforge