X-CLI : CLI-BASED MANAGEMENT ARCHITECTURE USING XML

Byung-Joon Lee, Taesang Choi and Taesoo Jeong
{bjlee,choits,tsjeong}@etri.re.kr, Internet Traffic Management Team, ETRI, Gajung-Dong, Yusong-Gu, Daejeon, Republic of Korea

Abstract: As Internet technology becomes more complex, the policy information for managing the Internet grows beyond the capability of a simple protocol like SNMP. IETF suggested COPS as an alternative, but it has not been widely accepted. For that reason, many administrators have developed network management systems which control network devices using CLI, but systems based on CLI have a maintenance problem: when the syntax of CLI changes, the implementation of the system must be modified. In this paper, we suggest X-CLI as a solution for this problem, and describe its design principles.

Key words: Network Management, CLI, XML, X-CLI, API

1. INTRODUCTION

Traditionally, SNMP has been a major management protocol for the IP network because of its simplicity. But for the new technologies such as MPLS, VPN or QoS, the policies for managing network have become too complex. IETF standardized COPS to cope with that complexity, but it is not being widely accepted.

For that reason, systems implemented on CLI (Command Line Interface) of the network devices are being widely used. Those systems translate a policy into a sequence of CLI commands, and send those commands to the devices using the TELNET protocol. However, those systems are dependent on the syntax of the CLI command: the syntax change of the CLI commands results in the implementation modification of the policy-to-CLI conversion logic.

In this paper, we suggest X-CLI (XML wrapper API for CLI) as an alternative to SNMP, COPS, and traditional CLI-based solutions. It is based on the concept of XML template which represents a group of CLI commands in a hierarchical manner. The template is converted into the actual CLI commands, and sent to the network devices by the X-CLI API interfaces.

2. XML TEMPLATE

The concept of 'XML template' corresponds to the concept of 'function' in general programming languages like C/C++: it maps a set of CLI commands to some specific configuration action. But the XML template differs from 'function' because it exists as a file, and designed to be able to represent hierarchical dependency, argument dependency and result dependency which are the basic characteristics of the CLI commands of most of the current network devices [1][2].

2.1 The Characteristics of CLI commands

Figure 1 shows the configuration steps for setting up an IBGP session between PE routers for the dissemination of the route information in the VRF table.

```
(config)# router bgp 55555
(config-router)# address-family ipv4 vrf VRF-SEOUL
(config-router-af)# neighbor 203.255.25.15 remote-as 5555
(config-router-af)# neighbor 203.255.25.15 activate
```

Figure 1. BGP configuration steps for VPN at CISCO PE router

As shown in the figure, the CLI commands are executed in a hierarchical manner. For example, the command "address-family ipv4 vrf VRF-SEOUL" cannot be executed without the successful execution of the command "router bgp 55555." We call this kind of behavior 'Hierarchical Dependency.'

In the command "router bgp 55555," the '55555' is the required argument of the CLI command "router bgp <as-number>". Without the argument, the CLI command cannot be executed. We call that kind of behavior 'Argument Dependency.'

As a result of a CLI command execution, one of the three following situations can happen: (1) CLI execution error (2) request more input from the administrator (3) successful execution. When (1) happens, most of the commands scheduled to be given to the devices cannot be delivered. For case (2), every scheduled command hangs until the additional input is given by the administrator. We name this kind of conditional behavior 'Result Dependency.'

2.2 XML Representation of the CLI Commands

As shown in the example XML template of Figure 2, an XML template is the hierarchy of the <cli></cli> XML tags. A containment relationship between <cli> tags represents a hierarchical dependency. Other kinds of the dependencies are represented by the attributes of the <cli> tag.

The attribute 'command' has CLI command string as its value. To represent the 'argument dependency', the command string can contain formal argument names which start with special character '$.' Parenthesis can be used with formal argument names to specify the optional part of the CLI command. The thorough description of the command string syntax can be found in [2].

```
<cli prompt1="#" command="config terminal" errorstr="^">
  <cli tag="bgp1" prompt1="#"
     command="router bgp $asnum" errorstr="^">
    <cli>
      <cli tag="bgp2" prompt1="#"
           command="address-family ipv4 vrf $vrfname"
           errorstr="^">
        <cli>
          <cli tag="bgp3" prompt1="#"
               command="neighbor $ipn1 remote-as $nasnum"
               errorstr="^">
          </cli>
          <cli tag="bgp4" prompt1="#"
               command="neighbor $ipn2 activate"
               errorstr="^">
          </cli>
        </cli>
        <cli prompt1="#" always="true" command="exit"></cli>
      </cli>
    </cli>
    <cli prompt1="#" always="true" command="exit"></cli>
  </cli>
  <cli prompt1="#" always="true" command="exit"></cli>
</cli>
```

Figure 2. XML template for Figure 1

The attribute 'errorstr' and 'always' express the result dependency. When a reply of the network device contains the value of the attribute 'errorstr', it is considered as an error. The attribute 'always' is a flag indicating that the CLI command can be executed in spite of the execution failure of the previous CLI command.

The attribute 'tag' is introduced for uniquely identifying <cli> tag. The prompt attribute 'prompt1' and 'prompt2' are needed to send a CLI command to the network device using TELNET. A client starts sending a CLI command to the server when the value of the attribute 'prompt1' is received from the server, and stops receiving replies from the server until the value of the attribute 'prompt2' is received. If not specified, the value of the attribute 'prompt2' is the same as that of 'prompt1.'

A <cli> tag with no attribute is called a PAT (Pure Aggregation Tag). A PAT specifies that the enclosed <cli> tags can be converted into actual CLI commands repeatedly. The <cli> tags not enclosed by the PAT can only be converted once.

3. X-CLI API

X-CLI API is the interface for manipulating XML template. It provides functionalities for loading an XML template, 'materializing' it, and sending it to the network device. The 'materialization' is the process of converting an XML template into a sequence of CLI commands.

After loaded into the memory by the X-CLI API, a template is translated into the internal data structure of a tree topology. This tree is materialized by traversing it with the arguments given by the X-CLI application programmer [2]. This process is similar to the act of passing arguments to a function which generates some specific control flow. The materialization result of the Figure 2 is shown in the Figure 3. The arguments '55555,' 'VRF-A-Seoul,' '203.255.255.13,' '5555' and '203.255.255.13' are delivered to the XML template in sequence.

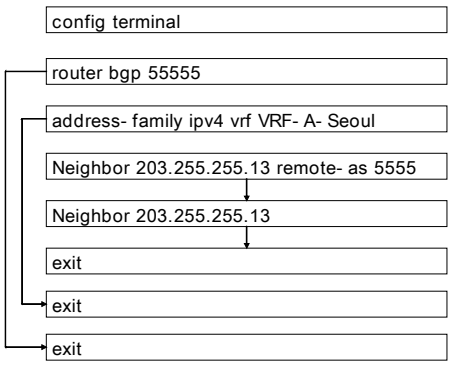

Figure 3. Materialized result of Figure 2

The materialized commands are sent to the network device sequentially. When an error happens, the failure branch target (depicted as an directed edge in Figure 3) is taken. After the branch, only commands which have 'true' value for the attribute 'always' can be sent to the device.

X-CLI API is greatly enhanced recently for the device monitoring. The <cli> tag is extended to express the monitoring actions, and the monitoring result is parsed automatically by the X-CLI API [2]. This feature aids the programmers who want to develop an application which monitors network device statistics using CLI.

4. CONCLUSION

X-CLI enhances the maintainability of the network management software by eliminating a dependency on the syntax of the CLI command from the software. The CLI syntax only exists in the XML templates. And besides, because the X-CLI API uses TELNET as its communication protocol, it can be easily secured by the SSH protocol. Adding support for the SSH to the X-CLI API can be easily done.

X-CLI API is being integrated with Wise<TE> [3], which is the network management server for traffic engineering, QoS and VPN. Wise<TE> is being tested against the several tens of the routers of Juniper and CISCO. The architecture of the Wise<TE> and the details about integration with X-CLI API is described in [2].

REFERENCES

[1]Byung-Joon Lee, Taesang Choi and Taesoo Jeong, "X-CLI: CLI based Policy Enforcement and Monitoring Architecure using XML", APNOMS 2002.

[2]Byung-Joon Lee, Taesang Choi and Taesoo Jeong, "X-CLI: CLI based Management Architecture using XML", Technical Report, 2002. http://oopsla.snu.ac.kr/~bjlee/document/x-cli/x-cli-techreport-2002.doc

[3]TS Choi, SH Yoon, HS Chung, CH Kim, JS Park, BJ Lee, TS Jeong, "Wise<TE>: Traffic Engineering Server for a Large-Scale MPLS-based IP Network", NOMS 2002