CUSTOMIZING THE ConfD CLI: OPERATIONAL MODE

tail-f

Table of Contents

1. Executive Summary 3

2. tailf:cli-full-show-path 4

3. tailf:cli-incomplete-show-path 5

4. tailf:cli-show-template value 6

5. tailf:cli-show-template-legend 8

6. tailf:cli-show-template-footer value 9

7. Summary 10

Introduction

One of ConfD's data model driven features is the ability to automatically render a CLI from a YANG data model. This powerful feature provides both development velocity and productivity benefits. The user can easily write a YANG data model, start ConfD, and log into a fully functional, industry-standard style CLI. However, the automatically rendered CLI maybe not be exactly what is desired. Therefore, ConfD provides a very extensive set of YANG extension statements which can be used to annotate a YANG data model in order to customize the auto-rendering.

This application note focuses on the most important operational mode CLI extensions for modifying the auto-rendered show commands that are commonly used when developing a CLI interface. For more information regarding these annotations please refer to the ConfD User Guide.

This application note is a companion to the previously published application note "Customizing the ConfD CLI: Configuration Mode".

Background

YANG

YANG is a data modeling language for the definition of data sent over the NETCONF network configuration protocol. The data modeling language can be used to model both the configuration data as well as the state data and administrative actions of network elements. Furthermore, YANG can be used to define the format of event notifications which can be generated by network elements and also allows data modelers to define the signature of extension remote procedure calls (RPCs) which can be invoked on network elements via the NETCONF protocol.

ConfD

ConfD is an embedded management software framework that enables network element providers to quickly and inexpensively deliver world-class management functionality and programmability for their products. The network elements can be physical devices or virtual devices (VNFs, Virtual routers, etc.). ConfD is data model-driven and provides automatic rendering of all northbound interfaces including NETCONF, RESTCONF, CLI, JSON-RPC, and SNMP as well as C, Python, JAVA, and Erlang APIs for application integration.

CLI Extensions

The CLI operational mode extensions are predefined annotations that are inserted into a YANG data model. The purpose of these annotations is to make the development of customized CLI show commands easier and reduce the time to market for a given product. The CLI operational mode extensions can change the look and feel of the auto-rendered show commands.

These YANG extension statements were written by Tail-f and are defined in a YANG data model which is included which the ConfD distribution. The CLI extension statements are defined in tailf-cli-extension.yang which is a submodule of tailf-common.yang. To make use of the CLI extensions in a YANG data model, simply import tailf-common.yang:

```
import tailf-common {
  prefix tailf;
}
```

tailf:cli-full-show-path

Definition: Specifies that a path to the show command is considered complete, i.e., no more elements can be added to the path.

Usage: Used in J-, I- and C-style CLIs. The cli-full-show-path statement can be used in: leaf, leaf-list, list, container, tailf:symlink, and refine.

Example:

Case 1: (no CLI extension)

YANG model:

```
container routing {
  config false;
  container bgp {
    list neighbor {
      key "neighbor-id remote-as";
      leaf neighbor-id {
        type inet:ipv4-address;
      }
      leaf remote-as {
        type uint16;
      }
    }
  }
}
```

CLI output:

```
# show routing bgp ?
Possible completions:
  neighbor |  <CR>
Router# show routing bgp
```

Case 2: (with CLI extension)

YANG model:

```
container routing {
  config false;
  container bgp {
    tailf:cli-full-show-path;
    list neighbor {
      key "neighbor-id remote-as";
      leaf neighbor-id {
        type inet:ipv4-address;
      }
      leaf remote-as {
        type uint16;
      }
    }
  }
}
```

CLI output:

```
# show routing bgp ?
Possible completions:
  | <CR>
Router# show routing bgp
```

Tip: It can also be used to specify a maximum number of keys to be given for lists.

tailf:cli-incomplete-show-path

Definition: Specifies that a path to the show command is considered incomplete, i.e., it needs more elements added to the path

Usage: Used in J-, I- and C-style CLIs.

The cli-incomplete-show-path statement can be used in: leaf, tailf:symlink, leaf-list, list, container, and refine.

Example:
Case 1: (no CLI extension)

YANG model:

```
container routing {
  config false;
  container bgp {
    list neighbor {
      key "neighbor-id remote-as";
      leaf neighbor-id {
        type inet:ipv4-address;
      }
      leaf remote-as {
        type uint16;
      }
    }
  }
}
```

CLI output:

```
Router # show routing bgp <CR>
% No entries found.
Router #
```

Case 2: (with CLI extension)

YANG model:

```
container routing {
  config false;
  container bgp {
    tailf:cli-incomplete-show-path;
    list neighbor {
      key "neighbor-id remote-as";
      leaf neighbor-id {
        type inet:ipv4-address;
```

continued on next page...

```
      }
      leaf remote-as {
        type uint16;
      }
    }
  }
}
```

CLI output:

```
Router# show routing bgp <CR>
-------------------------^
syntax error: incomplete path
Router #
```

Tip: It can also be used to specify a minimum number of keys to be given for lists.

tailf:cli-show-template value

Definition: Specifies a template string to be used by the 'show' command in operational mode. It is primarily intended for displaying non-configuration data but configuration data may be included in the template as well. See the definition of cli-template-string in the tailf_yang_cli_extensions(5) man page for more info.

Usage: Used in J-, I- and C-style CLIs.

Example:
Case 1: (no CLI extension)

YANG model:
Note: The following YANG model snippet is also used in the next two sections.

```
container routing {
  config false;
  container ospf {
    list neighbor {
      tailf:cli-show-template "\n ABC";
      key "neighbor-address process-id";
      leaf neighbor-address {
        type inet:ipv4-address;
      }
      leaf process-id {
        type uint32;
        }
      }
      leaf router-id {
        type inet:ipv4-address;
      }
      leaf if-name {
        type string;
      }
      leaf operational-status {
        type enumeration {
          enum up;
          enum down;
          enum going-up;
          enum going-down;
          enum failed;
```

continued on next page...

```
      }
    }
    <<< snip >>>
```

CLI output:

```
Router# show routing ospf neighbor?
router-id          2.2.2.2
 if-name            eth1
 operational-status up
 priority           1
 state              full
Router#
```

Case 2: (with CLI extension)

YANG model:

```
container routing {
  config false;
  container ospf {
    list neighbor {
      tailf:cli-show-template "\nParameters for OSPF neighbor\n
        RID\: $(router-id? $(router-id== ?n/a:$(router-id)):n/a)
        PRIO\: $(priority? $(priority==? n/a: $(priority)):n/a)
        STATE\: $(state? $(state==? n/a: $(state)):n/a)
        OPER\: $(operational-status? $(operational-status==?
          n/a:$(operational-status)):n/a)
        ADDRESS\: $(neighbor-address? $(neighbor-address==?
          n/a:$(neighbor-address)):n/a)
        IF-NAME\: $(if-name? $(if-name==?n/a:$(if-name)):n/a)";
      key "neighbor-address process-id";
      <<< snip >>>
```

CLI output:

```
Router# show routing ospf neighbor?
Parameters for OSPF neighbor
RID:  4.4.4.4
PRIO:   1
STATE:   full
OPER:  up
ADDRESS:  192.168.100.2
IF-NAME:  eth1
Router#
```

Tip: The template applied to a specific list provides a mechanism to manipulate the data that comes from data provider.

tailf:cli-show-template-legend

Definition: Specifies a template string to be printed before all list entries are printed. See the definition of cli-template-string in the tailf_yang_cli_extensions(5) man page for more info.

Usage: Used in J-, I- and C-style CLIs.

Example:

Case 1: (no CLI extension)

YANG model:
See Case 1 (no CLI extension) i the section “tailf:cli-show-template” above.

CLI output:
See Case 1 (no CLI extension) in the section “tailf:cli-show-template” above.

Case 2: (with CLI extension)

YANG model:
Use the template defined in the previous example and the legend.

```
container routing {
  config false;
  container ospf {
    list neighbor {
      tailf:cli-show-template-legend "\n-------------------------\n";
      tailf:cli-show-template "\nParameters for OSPF neighbor\n
        RID\: $(router-id? $(router-id== ?n/a:$(router-id)):n/a)
        PRIO\: $(priority? $(priority==? n/a: $(priority)):n/a)
        STATE\: $(state? $(state==? n/a: $(state)):n/a)
        OPER\: $(operational-status? $(operational-status==?
          n/a:$(operational-status)):n/a)
        ADDRESS\: $(neighbor-address? $(neighbor-address==?
          n/a:$(neighbor-address)):n/a)
        IF-NAME\: $(if-name? $(if-name==?n/a:$(if-name)):n/a)";
      key "neighbor-address process-id";
      <<< snip >>>
```

CLI output:

```
Router# show routing ospf neighbor <CR>
-------------------------
Parameters for OSPF neighbor
RID:  4.4.4.4
PRIO:  1
STATE:  full
OPER:  up
ADDRESS:  192.168.100.2
IF-NAME:  eth1
Router#
```

Tip: The legend annotation lets you specify a legend for the show command. Most of the simple opera to a specific list provides a mechanism to manipulate the data that comes from data provider. The legend in this case outputs a delimiter formed of ‘-’ characters.

tailf:cli-show-template-footer value

Definition: Specifies a template string to be printed after all list entries are printed. See the definition of cli-template-string in the tailf_yang_cli_extensions(5) man page for more info.

Usage: Used in J-, I- and C-style CLIs.

Example:

Case 1: (no CLI extension)

YANG model:
See Case 1 (no CLI extension) i the section “tailf:cli-show-template” above.

CLI output:
See Case 1 (no CLI extension) in the section “tailf:cli-show-template” above.

Case 2: (with CLI extension)

YANG model:

```
container routing {
  config false;
  container ospf {
    list neighbor {
      tailf:cli-show-template-footer "\n##########################\n";
      tailf:cli-show-template "\nParameters for OSPF neighbor\n
        RID\: $(router-id? $(router-id== ?n/a:$(router-id)):n/a)
        PRIO\: $(priority? $(priority==? n/a: $(priority)):n/a)
        STATE\: $(state? $(state==? n/a: $(state)):n/a)
        OPER\: $(operational-status? $(operational-status==?
          n/a:$(operational-status)):n/a)
        ADDRESS\: $(neighbor-address? $(neighbor-address==?
          n/a:$(neighbor-address)):n/a)
        IF-NAME\: $(if-name? $(if-name==?n/a:$(if-name)):n/a)";
      key "neighbor-address process-id";
      <<< snip >>>
```

CLI output:

```
Router# show routing ospf neighbor <CR>
Parameters for OSPF neighbor
RID:  4.4.4.4
PRIO:  1
STATE:  full
OPER:  up
ADDRESS:  192.168.100.2
IF-NAME:  eth1
##########################
Router#
```

Tip: Show template footer provides the final lines of output for a show command. In this case, the char ‘#’ was printed many times at the end of the show command output.

Conclusion

This application note has discussed some of the most commonly used YANG extension statements which are used to customize the default automatic rendering of the ConfD CLI. While this application note has focused on the most commonly used CLI operational mode customizations, there are many more customization statements available. There are also YANG extension statements for customizing the CLI configuration mode. Additionally, the existing commands can be modified or added to using definitions called "clispec" and implemented using scripts or executable programs.

For more information about ConfD:
visit http://www.tail-f.com

App Note: "Customizing the ConfD CLI: Configuration Mode":
http://info.tail-f.com/customizing-confd-cli

For more information about YANG:
visit https://tools.ietf.org/html/rfc7950

For more information about the ConfD CLI:
see the "CLI Agent" chapter of the ConfD User Guide

For more information about the ConfD CLI YANG extensions:
see the tailf_yang_cli_extensions(5) man page

To see more CLI customization options in action:
see examples.confd/cli in your ConfD distribution

tail-f

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40