IOS/IOS-XR
大量に検証コンフィグを作成する時の手法サンプル

Shishio Tsuchiya

shtsuchi@cisco.com

はじめに

- 今回お話する内容は筆者が普段検証環境にて、使用してる設定手法の共有です。
- 実施前に検証機などで十分確認し、稼働中の実機に投入する際には必ず自身で確認する様にして下さい。

Agenda

- IOS/IOS-XEの場合
- IOS-XRの場合

大量設定をやる前に

```
archive
 path flash:/config-enog-
 maximum 14
 write-memory
```

```
IOS#show archive
The maximum archive configurations allowed is 14.
There are currently 1 archive configurations saved.
The next archive file will be named flash:/config-enog--<timestamp>-1
 Archive #  Name
   1        flash:/config-enog-Jun-19-09-12-02.287-0 <- Most Recent
   2
```

- IOS/IOS-XEではデフォルトだとrollback出来ません。
- たった14ですが、無いより大分マシです。
- *configure replace* でrolback実施

http://tools.bgp4.jp/index.php?cmd=read&page=tools%20team%2Ftools%2FRouter%2FCisco%2FRollback

これを沢山作りたい

```
!
ip vrf 1
 rd 6500:1
!
interface GigabitEthernet0/0.2
 encapsulation dot1Q 2
 ip vrf forwarding 1
 ip address 10.10.10.1 255.255.255.0
!
```

- 変数は2つだけ

Tclで書くならこう

```
for {set x 1} {$x<=100} {incr x} {
set y [expr $x+1]
puts "ip vrf $x"
puts "rd 6500:$x”
puts "interface GigabitEthernet0/0.$y"
puts "encapsulation dot1Q $y"
puts "ip vrf forwarding $x"
puts "ip address 10.10.10.1 255.255.255.0"
}
```

- xを1から始めて、1個ずつ増やし、100まで
- yはx に1足す

IOS/IOS-XEではTclが使える

```
IOS#tclsh
IOS(tcl)#info tclversion
8.3
IOS(tcl)#set var "Hello! World"
Hello! World
```

- また拡張コマンドとして、*ios_config* がありIOS CLコンフィグが実行可能

初期状態

```
IOS#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         10.11.12.7      YES DHCP   up                    up
GigabitEthernet0/1         unassigned      YES NVRAM  administratively down down
IOS#show ip interface brief | count 0/0
Number of lines which match regexp = 1
```

- また拡張コマンドとして、*ios_config* がありIOS CLコンフィグが実行可能

初期状態

```
IOS#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         10.11.12.7      YES DHCP   up                    up
GigabitEthernet0/1         unassigned      YES NVRAM  administratively down down
IOS#show ip interface brief | count 0/0
Number of lines which match regexp = 1
```

- また拡張コマンドとして、*ios_config* がありIOS CLコンフィグが実行可能

Tcl ios_config

```
 for {set x 1} {$x<=100} {incr x} {
 set y [expr $x+1]
 ios_config "ip vrf $x" "rd 6500:$x" "interface GigabitEthernet0/0.$y" "encapsulation dot1Q
$y" "ip vrf forwarding $x" "ip address 10.10.10.1 255.255.255.0"
 }
```

- コマンド毎に””で区切る。

実施後

```
IOS(tcl)# for {set x 1} {$x<=100} {incr x} {
+> set y [expr $x+1]
+>$ $y" "ip vrf forwarding $x" "ip address 10.10.10.1 255.255.255.0"
+> }

Jun 19 11:05:22.959: %SYS-5-CONFIG_I: Configured from console by vty0
-------
IOS(tcl)#
IOS#show ip interface brief | count 0/0
Number of lines which match regexp = 101
```

- 簡単で取りこぼしも少ないが、コンフィグ->Tclのイメージがちょっと難しい？

Perl（コンフィグファイル作成）

```
#!/usr/local/bin/perl
for ($x =1; $x <=100; $x++){
$y=$x+1;
print
"!
ip vrf $x
 rd 6500:$x
!
interface GigabitEthernet0/0.$y
 encapsulation dot1Q $y
 ip vrf forwarding $x
 ip address 10.10.10.1 255.255.255.0
!
";}
exit;
```

- シンプルコンフィグを大量生産
- コピペすると取りこぼすかも
- ネットワーク経由のコピーならまぁ大丈夫
- ファイルはでかくなる

Perl + Treraterm macro

```
#!/usr/local/bin/perl
for ($x =1; $x <=100; $x++){
$y=$x+1;
print "sendln 'ip vrf $x'\n";
print "wait '#'\n";
print "senln 'rd 6500:$x'\n";
print "wait '#'\n";
print "interface GigabitEthernet0/0.$y'\n";
print "wait '#'\n";
print "encapsulation dot1Q $y'\n";
print "wait '#'\n";
print "ip vrf forwarding $x'\n";
print "wait '#'\n";
print "ip address 10.10.10.1 255.255.255.0'\n";
print "wait '#'\n";
;}
exit;
```

- Teraterm macroと組み合わせちゃう
- sendln 文字列と改行を送る
- wait ‘’の文字を待つ
- Teraterm macroファイルとして実施
- コンフィグの実施が目で確認出来る
- ファイルがでかくなる

Teratermマクロでコンフィグを読み出す

```
fileopen fhandle 'filename.txt' 0

:loop
filereadln fhandle line
if result goto fclose



sendln line
wait '#'

goto loop

:fclose
fileclose fhandle
```

- コンフィグファイルを読み出し
- 一行ずつ送る
- #を待つ
- 応用は一番効くかも？（単純なコンフィグじゃなくて、実際に使用してるコンフィグの再現試験時にも使える）
- ファイルが大きいのは一緒

Teratermマクロでコンフィグを書いてしまう。

```
 x = 0
do while x < 100

x  = x + 1
y  = x + 1
int2str countx x
int2str county y

sendln 'ip vrf ' countx
wait '#'
sendln 'rd 6500:'countx
wait '#'
sendln 'interface GigabitEthernet0/0.'county
wait '#'
sendln 'encapsulation dot1Q 'county
wait '#'
sendln 'ip vrf forwarding ' countx
wait '#'
sendln 'ip address 10.10.10.1 255.255.255.0'
wait '#'

loop
```

- do/loopの繰り返し whileで抜ける
- int2strで整数値を文字列に変更
- sendln/wait
- 一番軽い。でも･･･Teratermに依存しすぎ？

Agenda

- IOS/IOS-XEの場合
- IOS-XRの場合

IOS-XR

- デフォルトでrollbackは可能
- configが間違ってたらshow configuration failedで教えてもらえる
- 階層式なコンフィグ表示
- IOS Tclが無い

IOS-XRとIOSの違い

```
 !
 vrf 1
 address-family ipv4 unicast
!
interface GigabitEthernet0/0/0/0.2
 vrf 1
 ipv4 address 10.10.10.1 255.255.255.0
 encapsulation dot1q 2
!
```

```
!
ip vrf 1
 rd 6500:1
!
interface GigabitEthernet0/0.2
 encapsulation dot1Q 2
 ip vrf forwarding 1
 ip address 10.10.10.1 255.255.255.0
!
```

- 設定パラメーターがグローバルユニークでは無い
- 実はスクリプトを書くとき、階層を意識する必要があって、めんどくさい。
- 実際のコマンドを投入する時は1ラインでの実施も可能

show running-config formal

```
 RP/0/0/CPU0:IOS-XR#show running-config formal
hostname IOS-XR
vrf 1
vrf 1 address-family ipv4 unicast
interface GigabitEthernet0/0/0/0.2
interface GigabitEthernet0/0/0/0.2 vrf 1
interface GigabitEthernet0/0/0/0.2 ipv4 address 10.10.10.1 255.255.255.0
interface GigabitEthernet0/0/0/0.2 encapsulation dot1q 2
```

- formal ?な表示をする事ができる
- スクリプトで設定を入れるのであれば、こちらを使用すると良い

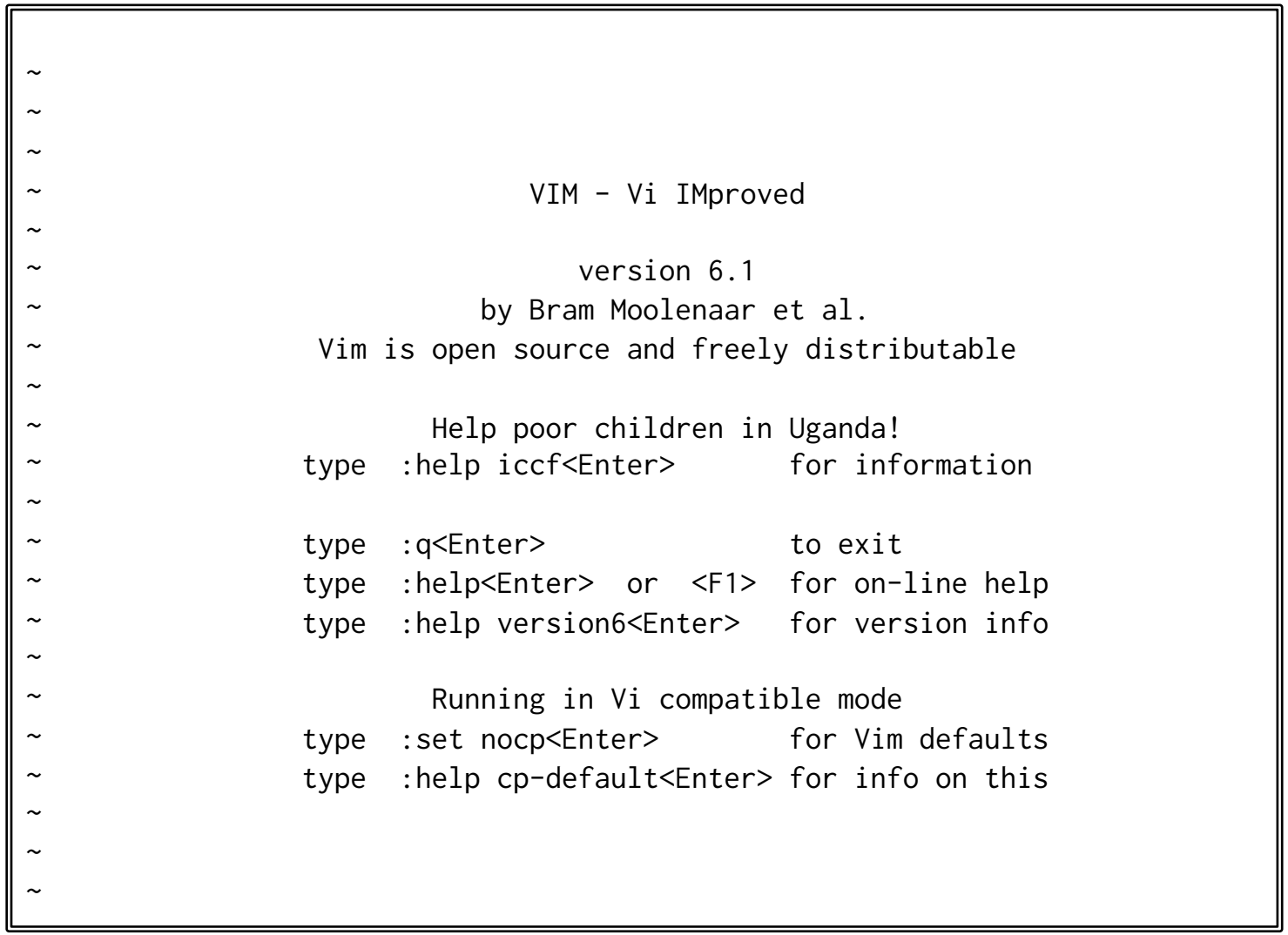
Tclは無いが・・・

```
RP/0/0/CPU0:IOS-XR#run
Thu Jun 19 13:14:39.231 UTC
# perl -v

This is perl, v5.6.0 built for 4k-

Copyright 1987-2000, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5.0 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using `man perl' or `perldoc perl'.  If you have access to the
Internet, point your browser at http://www.perl.com/, the Perl Home Page.
```

Tclは無いが・・・

```
RP/0/0/CPU0:IOS-XR#run
Thu Jun 19 13:14:39.231 UTC
# perl -v

This is perl, v5.6.0 built for 4k-

Copyright 1987-2000, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5.0 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using `man perl' or `perldoc perl'.  If you have access to the
Internet, point your browser at http://www.perl.com/, the Perl Home Page.
```

vimも使えちゃう



なので直接編集

```
# less xr.pl
#!/usr/local/bin/perl
for ($x =1; $x <=100; $x++){
$y=$x+1;
print
 “vrf $x address-family ipv4 unicast
interface GigabitEthernet0/0/0/0.$y vrf $x
interface GigabitEthernet0/0/0/0.$y ipv4 address 10.10.10.1 255.255.255.0
interface GigabitEthernet0/0/0/0.$y encapsulation dot1q $y
";}
exit;
```

load

```
# perl xr.pl > xr.txt
#
# exit
RP/0/0/CPU0:IOS-XR#conf t
Thu Jun 19 14:38:54.574 UTC
RP/0/0/CPU0:IOS-XR(config)#lo
load  locale  logging
RP/0/0/CPU0:IOS-XR(config)#load usr/xr.txt
Loading.
24754 bytes parsed in 1 sec (24268)bytes/sec
RP/0/0/CPU0:IOS-XR(config)#commit
Thu Jun 19 14:39:19.113 UTC
RP/0/0/CPU0:IOS-XR(config)#
```

まとめ

- スクリプトと設定ファイルの見た目が同じ（似てる）方がいいかも。
- 投入内容を確認出来る手法の方が望ましい
- 元に戻せる事も重要

Thank you.