

CISCO IOS TCL and RCMD testing and troubleshooting scripting

The following paper does not claim to be a complete treatise on TCL. Instead, it describes how the TCL shell on a Cisco router can be used to automate some testing tasks during a typical lab exam. If you require more information about TCL, you can make use of a number of tutorials on the web, or one of the many books on the subject.

Somewhere during any successful CCIE lab attempt, it will be necessary to test the reachability of addresses from each device in your testbed. Some students use cut and paste techniques coupled with Notepad to ping the addresses of interest. Unfortunately, there are numerous drawbacks to this technique. Often, students have to manually change terminal windows to issue the pings on each device. In addition, the student has to tweak the terminal program settings in order to avoid filling the console buffer on the router. In short, Notepad cut and paste isn't the most carefree way to look for routing problems in a lab environment.

Coupling TCL (tool command language) with the RCMD facility on a Cisco router allows a single router to rapidly send an entire string of ping commands to a list of routers without overloading the input buffer. Furthermore, the script can automatically assemble a list of addresses used in the topology, and will send the pings to the correct router without manual intervention from the student.

Checking reachability on a router requires the following steps:

1. Select a router that can actually execute the TCL script. TCL runs on the 2600 series routers, 3600 series, and 7500 series units, but seems to be unavailable on the 2500 series, 4000 / 4500 series and Catalyst 3500s.

Note that TCL can send ping commands to routers even if the receiving routers cannot themselves run TCL. As long as you have one device that supports the TCL Shell software, these techniques will work.

2. Select an address on each router that will allow the TCL script to contact that particular device. Insure that the router that will be running the TCL script can successfully ping these addresses. Select an interface on the router running the TCL script that will be used as a TCL source address. Make sure that all other routers can reach this address.
3. Configure the necessary RCMD commands on the router running the TCL script as well as on the rest of the routers in the topology.
4. Using TCL, assemble a list of all of the IP addresses used on all links in your topology. Manually edit the output so that it can be used in the subsequent TCL ping script.
5. Create, debug and run the script to ping the collected addresses.

Here are the steps that I use to accomplish the aforementioned tasks.

The test topology contains routers R1, R2, and R3. All routers contain ip addresses that are fully reachable from all of the other routers. Each router contains a loopback address configured along the lines of 172.16.10N.1 /24 where N is the router number. R1 is a 2621XM router, and contains the necessary IOS code to support TCL commands.

1. First off, configure R1 with the following RCMD commands:

```
R1#
Config ter
Int loop101
Ip address 172.16.101.1 255.255.255.0
No shut

ip rcmd rsh-enable
ip rcmd remote-host R1 172.16.101.1 R1 enable
ip rcmd source-interface Loopback101
end
```

Configure R2 and R3 in a similar fashion. The RCMD source-interface command is not necessary on these two.

```
R2#
config ter
int loop102
ip address 172.16.102.1 255.255.255.0
!
ip rcmd rsh-enable
ip rcmd remote-host r1 172.16.101.1 r1 enable
end
```

2. Collect the IP addresses that need to be tested for reachability. Use the following script to accomplish this task. Type in the script exactly as shown. Moving { } or spaces will cause the script to fail.

Also, note the space after the “^” in the following script. Examining the output from the Show Run command demonstrates that IOS always indents the line “ip address x.x.x.x” by a single space when it sits below the interface command. This space does not exist when searching for the hostname keyword in the same running config.

```
R1#
tclsh
foreach router {
172.16.101.1
172.16.102.1
172.16.103.1
} {
rsh $router show run | include ^ ip address \[0-9]
}
```

This script will produce output that looks something like this:

```
ip address 172.16.101.1 255.255.255.0
ip address 172.16.10.129 255.255.255.128
ip address 172.16.123.1 255.255.255.128
ip address 172.16.21.1 255.255.255.128
ip address 172.16.16.1 255.255.255.128
ip address 172.16.10.1 255.255.255.128

ip address 172.16.20.2 255.255.255.128
ip address 172.16.123.2 255.255.255.128
ip address 172.16.21.2 255.255.255.128

ip address 172.16.200.126 255.255.255.128
ip address 172.16.30.3 255.255.255.128
ip address 172.16.123.3 255.255.255.128
ip address 172.16.34.3 255.255.255.128
ip address 172.16.35.3 255.255.255.128
```

(The spaces between the above sets of addresses shows where the \$router variable changed from one entry in the foreach list to the next. As a result, the first set of ip addresses comes from R1, the second from R2 and the third from R3).

3. Pasting this output into Notepad and using the Search / Replace capability, remove everything except the IP addresses. Following editing, the above output will look like this:

```
172.16.101.1
172.16.10.129
172.16.123.1
172.16.21.1
172.16.16.1
172.16.10.1
172.16.20.2
172.16.123.2
172.16.21.2
172.16.200.126
172.16.30.3
172.16.123.3
172.16.34.3
172.16.35.3
```

4. Create a TCL script to ping each of these addresses from each of the routers in our topology as follows:

```
R1#
tclsh
foreach router {
172.16.101.1
172.16.102.1
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max =  
1/2/4 ms
```

```
hostname R1
```

```
172.16.123.1
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 172.16.123.1, timeout is  
2 seconds:
```

```
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max =  
4/4/4 ms
```

This script makes it very easy to identify reachability problems in the topology. Issuing a Ping via RSH achieves a rapid response from the router actually executing the ping. This means that when pings are successfully reaching their target, each set of pings for individual addresses will occur with machine gun rapidity. It's only when a ping fails, that the speed of the command execution slows to a halt. When watching the execution of the script, successful pings will almost occur too rapidly to read the actual script output. When pings fail however, it will be easy to identify the failing address and from where the ping actually originated.

The script only uses the "Puts" command, and Set Hostname command to format the output into something that is easy to read. It also allows immediate identification of which router and which address is pinging at any given time.

These tools are very helpful to test the reachability in the DOiT scenarios:

<http://www.netmasterclass.com/site/articles/DOiT-Workbook-FAQ.pdf>

[http://www.netmasterclass.com/site/articles/flash/DOiT Workbook Presentation.swf](http://www.netmasterclass.com/site/articles/flash/DOiT%20Workbook%20Presentation.swf)

Look at the other articles in the NetMasterClass Technical Library:

<http://www.netmasterclass.net/site/lib.php>

If you have any question about this article please send them to <http://bbs.netmasterclass.com>