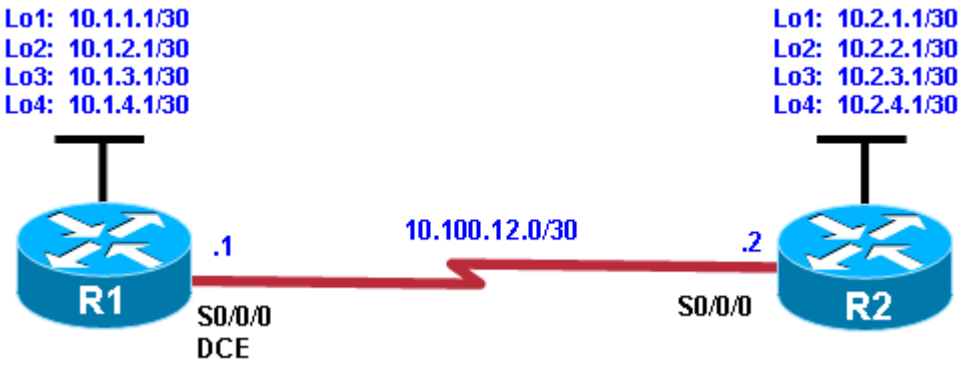
Chapter 1 Lab 1-1, Tcl Script Reference and Demonstration

Topology



Objectives

- Use Tcl scripts to verify full connectivity.
- Identify causes of failures.

Background

The Cisco IOS Scripting feature provides the ability to run Tool Command Language (Tcl) commands from the Cisco IOS command-line interface (CLI). Tcl scripts can be created to accomplish routine and repetitive functions with Cisco IOS-based networking devices. In this lab, you create and execute a Tcl script that sends pings to multiple IP addresses in the network to test overall network connectivity.

Note: Cisco IOS Release 12.3(2)T and later supports Tcl scripting.

Required Resources

- 2 routers (Cisco 1841 with Cisco IOS Release 12.4(24)T1 Advanced IP Service or comparable)
- Serial and console cables

Note: This lab uses Cisco 1841 routers with Cisco IOS Release 12.4(24)T1 and the advanced IP image c1841-advipservicesk9-mz.124-24.T1.bin. Other routers (such as a 2801 or 2811) and Cisco IOS Software versions can be used if they have comparable capabilities and features. Depending on the router model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Step 1: Configure initial settings.

Copy and paste the following initial configurations for R1 and R2.

Router R1

```
hostname R1
!
interface loopback 1
 ip address 10.1.1.1 255.255.255.252
!
interface loopback 2
 ip address 10.1.2.1 255.255.255.252
!
interface loopback 3
 ip address 10.1.3.1 255.255.255.252
!
interface loopback 4
 ip address 10.1.4.1 255.255.255.252
!
interface serial 0/0/0
 ip address 10.100.12.1 255.255.255.252
 clock rate 64000
 bandwidth 64
 no shutdown
!
router rip
 version 2
 network 10.0.0.0
 no auto-summary
!
end
```

Note: A 30-bit subnet mask (255.255.255.252) is used for the serial links in this lab. However, starting with IOS 12.2(4)T, the 31-bit subnet mask (255.255.255.254) is supported on IPv4 point-to-point interfaces (per RFC 3021), requiring only 2 IP addresses per point-to-point link (.0 and .1). The IP Unnumbered feature can also be used to conserve IP addresses.

Router R2

```
hostname R2
!
interface loopback 1
 ip address 10.2.1.1 255.255.255.252
!
interface loopback 2
 ip address 10.2.2.1 255.255.255.252
!
interface loopback 3
 ip address 10.2.3.1 255.255.255.252
!
interface loopback 4
 ip address 10.2.4.1 255.255.255.252
!
interface serial 0/0/0
 bandwidth 64
```

```
 no shutdown
!
router rip
 version 2
 network 10.0.0.0
 no auto-summary
!
end
```

Do you think that these configurations will achieve full connectivity between R1 and R2? Explain.

Step 2: Verify connectivity.

The simplest way to verify OSI Layer 3 connectivity between two routers is to use ICMP. ICMP defines a number of message types in RFC 792 for IPv4 and RFC 4443 for IPv6. (See www.ietf.org and http://tools.ietf.org for more information.)

ICMP defines procedures for echo (ping), traceroute, and source notification of unreachable networks. Pinging an IP address can result in a variety of ICMP messages, but the only message indicating that a ping is successful is the ICMP echo reply message indicated by an exclamation point (!) in the output of the **ping** command. The following command on R1 pings its Lo1 interface. Loopback interfaces always have a status of UP/UP.

```
R1# ping 10.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

In Step 1, you might have noticed that the R2 configuration omits an IP address on serial 0/0/0. R2 does not exchange IP packets with R1 because the IP protocol is not running on the R2 serial interface until the IP address has been configured.

Without this IP address, for which addresses in the topology diagram do you expect the ping to fail?

Step 3: Create and execute a Tcl script.

Tcl scripts can be created to accomplish routine and repetitive functions with Cisco IOS-based networking devices. To construct a simple connectivity verification script, do the following.

a. Open a text editor and create a new text file. Using a text file saves time, especially if you are pasting the Tcl script into multiple devices.

b. Start with the **tclsh** command to enter Tcl shell mode in which you can use native Tcl instructions like **foreach** or issue EXEC mode commands. You can also access configuration mode from within the Tcl shell and issue configuration commands from their respective menus, although these features are not explored in this lab.

```
R1# tclsh
R1(tcl)#
```

c. Begin a loop using the **foreach** instruction. The loop iterates over a sequence of values, executing a defined sequence of instructions once *for each* value. Think of it as "for each *value* in *Values*, do each *instruction* in *Instructions*." For each iteration of the loop, *$identifier* reflects the current *value* in *Values*. The **foreach** instruction uses the following model.

```
foreach identifier {
value1
value2
.
.
.
valueX
} {
instruction1
instruction2
.
.
.
instructionY
}
```

d. To create a Tcl script that pings every IP address in the topology, enter each IP address in the value list. Issue the **ping $address** command as the only instruction in the instruction list.

```
foreach address {
 10.1.1.1
 10.1.2.1
 10.1.3.1
 10.1.4.1
 10.100.12.1
 10.2.1.1
 10.2.2.1
 10.2.3.1
 10.2.4.1
 10.100.12.2
} {
 ping $address
}
```

e. Enter Tcl mode with the **tclsh** command, and copy the Tcl script from the text file and paste it into R1.

```
R1# tclsh

R1(tcl)#foreach address {
```



```
+>(tcl)#10.1.1.1
+>(tcl)#10.1.2.1
+>(tcl)#10.1.3.1
+>(tcl)#10.1.4.1
+>(tcl)#10.100.12.1
+>(tcl)#10.2.1.1
+>(tcl)#10.2.2.1
+>(tcl)#10.2.3.1
+>(tcl)#10.2.4.1
+>(tcl)#10.100.12.2
+>(tcl)#} {
+>(tcl)#ping $address
+>(tcl)#}
```

Note: You might need to press Enter to execute the script.

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.4.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.3.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.4.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

f. Enter Tcl mode with the **tclsh** command, and copy the Tcl script from the text file and paste it into R2.

```
R2# tclsh

R2(tcl)#foreach address {
+>(tcl)#10.1.1.1
+>(tcl)#10.1.2.1
+>(tcl)#10.1.3.1
+>(tcl)#10.1.4.1
+>(tcl)#10.100.12.1
+>(tcl)#10.2.1.1
+>(tcl)#10.2.2.1
+>(tcl)#10.2.3.1
+>(tcl)#10.2.4.1
+>(tcl)#10.100.12.2
+>(tcl)#} {
+>(tcl)#ping $address
+>(tcl)#}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.3.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.4.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.4.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

g. Exit Tcl mode using the **tclquit** command on each device.

```
R1(tcl)#tclquit
```

Note: You can also use the **exit** command to exit Tcl mode.

Notice that in the previous output, R1 and R2 could not route pings to the remote loopback networks for which they did not have routes installed in their routing tables.

You might have also noticed that R1 could not ping its local address on serial 0/0/0. This is because with PPP, HDLC, Frame Relay, and ATM serial technologies, all packets, including pings to the local interface, must be forwarded across the link.

For instance, R1 attempts to ping 10.100.12.1 and routes the packet out serial 0/0/0, even though the address is a local interface. Assume that an IP address of 10.100.12.2/30 is assigned to the serial 0/0/0 interface on R2. When a ping from R1 to 10.100.12.1 reaches R2, R2 evaluates that this is not its address on the 10.100.12.0/30 subnet and routes the packet back to R1 using its serial 0/0/0 interface. R1 receives the packet and evaluates that 10.100.12.1 is the address of the local interface. R1 opens this packet using ICMP, and responds to the ICMP echo request (ping) with an echo reply destined for 10.100.12.1. R1 encapsulates the echo reply at serial 0/0/0 and routes the packet to R2. R2 receives the packet and routes it back to R1, the originator of the ICMP echo. The ICMP protocol on R1 receives the echo reply, associates it with the ICMP echo that it sent, and displays the output in the form of an exclamation point.

Note: To understand this behavior, you can observe the output of the **debug ip icmp** and **debug ip packet** commands on R1 and R2 while pinging with the configurations provided in Step 1.

Step 4: Resolve connectivity issues.

a. On R2, assign the IP address 10.100.12.2/30 to serial 0/0/0.

```
R2# conf t
R2(config)# interface serial 0/0/0
R2(config-if)# ip address 10.100.12.2 255.255.255.252
```

b. On each router, verify the receipt of RIPv2 routing information with the **show ip protocols** command.

```
R1# show ip protocols
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 28 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial0/0/0           2     2
    Loopback1             2     2
    Loopback2             2     2
    Loopback3             2     2
    Loopback4             2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.100.12.2           120      00:00:13
  Distance: (default is 120)
```

```
R2# show ip protocols
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 26 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial0/0/0           2     2
    Loopback1             2     2
    Loopback2             2     2
    Loopback3             2     2
    Loopback4             2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.100.12.1           120      00:00:14
  Distance: (default is 120)
```

c. On each router, verify full connectivity to all subnets in the diagram by issuing the **tclsh** command and pasting the Tcl script on the command line in privileged EXEC mode.

```
R1# tclsh

R1(tcl)#foreach address {
+>(tcl)#10.1.1.1
+>(tcl)#10.1.2.1
+>(tcl)#10.1.3.1
+>(tcl)#10.1.4.1
+>(tcl)#10.100.12.1
+>(tcl)#10.2.1.1
+>(tcl)#10.2.2.1
+>(tcl)#10.2.3.1
+>(tcl)#10.2.4.1
+>(tcl)#10.100.12.2
+>(tcl)#} {
+>(tcl)#ping $address
+>(tcl)#}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.4.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/64 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.4.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
R1(tcl)#tclquit
```

Notice that the average round-trip time for an ICMP packet from R1 to 10.100.12.1 is approximately twice that of a packet from R1 to loopback1 on R2. This verifies the conclusion reached in Step 3 that the ICMP echo request to 10.100.12.1 and the ICMP echo reply from 10.100.12.1 each traverse the link *twice* to verify full connectivity across the link.

```
R2# tclsh

R2(tcl)#foreach address {
+>(tcl)#10.1.1.1
+>(tcl)#10.1.2.1
+>(tcl)#10.1.3.1
+>(tcl)#10.1.4.1
+>(tcl)#10.100.12.1
+>(tcl)#10.2.1.1
+>(tcl)#10.2.2.1
+>(tcl)#10.2.3.1
+>(tcl)#10.2.4.1
+>(tcl)#10.100.12.2
+>(tcl)#} {
+>(tcl)#ping $address
+>(tcl)#}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.3.1, timeout is 2 seconds:
```

```
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.4.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.4.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.100.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/68 ms
R2(tcl)#tclquit
```

Notice also that the average round-trip time for an ICMP packet from R2 to 10.100.12.2 is approximately twice that of a packet from R2 to loopback1 on R1.

Conclusion

The creation of Tcl scripts takes a little extra time initially but can save considerable time during testing each time the script is executed. Use Tcl scripts to verify all your configurations in this course. If you verify your work, both academically and in production networks, you will gain knowledge and save time in troubleshooting.

Router Interface Summary Table

Router Interface Summary				
Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
1700	Fast Ethernet 0 (FA0)	Fast Ethernet 1 (FA1)	Serial 0 (S0)	Serial 1 (S1)
1800	Fast Ethernet 0/0 (FA0/0)	Fast Ethernet 0/1 (FA0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2600	Fast Ethernet 0/0 (FA0/0)	Fast Ethernet 0/1 (FA0/1)	Serial 0/0 (S0/0)	Serial 0/1 (S0/1)
2800	Fast Ethernet 0/0 (FA0/0)	Fast Ethernet 0/1 (FA0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)

Note: To find out how the router is configured, look at the interfaces to identify the type of router and how many interfaces the router has. Rather than list all the combinations of configurations for each router class, this table includes identifiers for the possible combinations of Ethernet and serial interfaces in the device. The table does not include any other type of interface, even though a specific router might contain one. An example of this is an ISDN BRI interface. The string in parenthesis is the legal abbreviation that can be used in Cisco IOS commands to represent the interface.