



Global Knowledge™
Experts Teaching Experts

Expert Reference Series

Cisco IOS Tips and Tricks

Cisco IOS Tips and Tricks

By Denise Donohue

In this paper, you will learn how to use tools and commands built into Cisco's IOS to make configuring and monitoring your Cisco devices quicker and easier. We'll start with some basic, commonly used commands and settings. Then we'll move into more advanced ways to make your Cisco life easier, including some commands that are new with the most recent versions of the IOS. Topics covered include:

- Auto-completing commands
- Disabling DNS server lookup
- Limiting EXEC interruptions
- Setting the console speed
- Deprecated commands
- Time-stamping messages
- Setting the clock and time zone
- Displaying an interface config
- Filtering the output of commands
- Erasing an interface config
- The Do command
- Alias commands
- Privilege levels
- Changing the TCP timeout
- Stopping runaway debugs
- Editing access lists
- Router autoconfiguration

Auto-Completion of Commands

When typing a command, it is only necessary to type enough letters to make the command unique at your current configuration mode. Once you have enough letters that the IOS can understand what command you mean, you can stop typing. Type a few letters, then press the **TAB** key – if the router understands the command, it will complete it for you. If the router doesn't understand the command, type another letter or two, press **TAB** again, and see if that's enough. Typing fewer letters per command speeds up your configuration, and creates fewer changes for typos!

Disable DNS Server Lookup

When you type something (at the user or privileged prompt) that the router doesn't recognize as a command, it assumes that you want to telnet there. So it telnets there if it's an IP address, or looks in the config for a host table if you typed a word. If the word can't be resolved to an IP address via a host table, the router broadcasts for a DNS server. This is good if what you typed was an IP address or host name that you want to telnet to – it means you don't have to type the word "telnet." This is bad if you just mistyped a command. It ties up the router until the DNS request times out. You can turn off the DNS request with the command **no ip domain-lookup** at the global config mode.

Example:

```
Router#debug
Translating "debug"...domain server (255.255.255.255)
(255.255.255.255)
```

```

Translating "debug"...domain server (255.255.255.255)
% Unknown command or computer name, or unable to find computer
address

Router(config)#no ip domain-lookup
Router(config)#^Z
Router#debug
Translating "debug"
Translating "debug"
% Unknown command or computer name, or unable to find computer
address

```

Limit EXEC Interruptions

If the router has a message for you, it will display the message, even if that means interrupting something you were typing. Then you're stuck having to finish your command at the end of the router's message. The command **logging synchronous** under the console line configuration mode corrects this. The router still displays its message but will redisplay the command you were typing on the line when it's done. To turn off logging messages to the console altogether, give the command **no logging console** under global configuration mode. *Note:* This may cause you to miss some important messages from the router. You can modify this command by specifying the severity level of the message you want to turn off.

Example:

```

Router(config)#^Z
Router#clo
00:07:31: %SYS-5-CONFIG_I: Configured from console by console
% Incomplete command.

Router(config)#line con 0
Router(config-line)#logging synchronous
Router(config-line)#^Z
Router#clo
00:08:39: %SYS-5-CONFIG_I: Configured from console by console
Router#clo ! router retyped this

```

Setting the Console Speed

To connect via the console to a Cisco device, your terminal emulation program needs to be set to the following specifications. (Later in this paper, we'll tell you how to change the console speed.) Baud rate: 9600 bps, Data bits: 8, Parity: None, Stop bits: 1, Flow control: None. These are the default settings for both the console and the auxiliary ports.

Although the console port defaults to a blazing 9600 bps, some router models will allow you to increase the console speed to as high as 115200. Increasing the console speed has its advantages. At face value, it will enhance the response time when typing at the console and will allow console messages to be sent to the console faster. But more importantly, it is useful for those times when you have to upgrade the IOS through the console port.

To change the console speed, you simply use the **speed** command in line configuration mode for console 0. The syntax for the speed command is **speed <new speed>**. Options may vary on different platforms. Once you change the console speed, you will no longer have console access through your existing terminal setting. You will have to change the console setting on your terminal emulation program to match.

Example:

```
!  
line console 0  
    speed 115200  
!
```

Deprecated Commands

There are some commands that are officially no longer supported by Cisco, but that still work on most devices. A few of these commands are shorter than their modern-day replacements, so using them can speed up your configuration. Some commonly used ones include:

Old Command

wr (write memory)
wr er (write erase)
who
whe (where)

New Command

copy run start (copy running-config startup-config)
erase start
show users
show sessions

Timestamping Messages

When the router displays log or debug messages, it defaults to showing you how long the router had been up when the event occurred. Many times, it is more helpful instead to know the date and time when the event occurred. The command for this is **service timestamps [log | debug] datetime localtime** in global configuration mode. If you use this mode of logging, you should set the clock and date as shown in the next section.

Example:

```
Router(config)#^Z  
Router#  
00:39:40: %SYS-5-CONFIG_I: Configured from console by console  
  
Router(config)#service timestamps log datetime localtime  
Router(config)#^Z  
Mar  2 15:47:04: %SYS-5-CONFIG_I: Configured from console by console
```

Setting the Clock and Time Zone

You can set the clock on the router with the privileged mode command: **clock set <hh:mm:ss> <date> <MONTH> <year>**. When issuing this command, you must use the actual name of the month, not its number.

You may also want to set the time zone and daylight savings time adjustments. This is done in global configuration mode. The commands are: **clock timezone <NAME_OF_ZONE> <hours-offset-from-GMT>**, and **clock summer-time <name-of-timezone> recurring**.

Example:

```
Router#clock set 3:42:00 2 MARCH 2004
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#clock timezone EST -5
Router(config)#clock summer-time EDT recurring
Router#show clock
16:02:18.331 EST Tue Mar 2 2004
```

Displaying an Interface Config

Do you ever have times when all you want to see is the configuration for just one interface? But if you display the whole running configuration, you have to press the spacebar several times to get down to the one specific interface you want to see. Well, the “show run” command can be modified in several different ways (see later section). To quickly display only the configuration for a specific interface, use that interface designation as the modifier.

Example:

```
Router#show run interface s8
Building configuration...

Current configuration : 96 bytes
!
interface Serial8
 bandwidth 115
 ip address 10.0.0.9 255.255.255.252
 clock rate 115200
end
```

Erasing an Interface Config

When you need to change the configuration of an interface, you typically have to remove the unwanted parts of the existing configuration line by line by putting a “no” in front of each line. If you have a lot of changes, or if you need to completely redo the configuration, it’s faster to just reset the interface to its original, default configuration. This removes all existing commands and shuts down the interface. Use the command **default interface <interface_number>**.

Example:

```
interface Serial1/0
  no ip address
  encapsulation frame-relay
  clock rate 800000
  frame-relay intf-type dce
  frame-relay route 102 interface Serial2 201
  frame-relay route 103 interface Serial3 301

Router(config)#default interface s1
Building configuration...

Interface Serial1 set to default configuration
*Feb 28 19:04:56: %LINK-3-UPDOWN: Interface Serial1, changed state
to down

Router#show run interface s1/0
interface Serial1
  no ip address
end
```

Filtering the Output of Commands

What if you want to see the configuration for your routing protocol? That's listed near the end of the router's configuration, after all the interfaces. Typing the command "show run" would likely result in your needing to page through several screens of configuration before getting to the part you need. The "show run" and "show start" command have several modifiers. We looked at the "interface" modifier above. In this section, we look at using the pipe command to filter what is displayed on the console.

The pipe character is typically found above the backslash on the keyboard, and looks like this: |. Many of the show commands can be modified this way. There are three options when using the pipe – begin, include, and exclude. You then specify a value the router must search for. *Note:* Any words used as values in this command are case-sensitive.

| **Begin** causes the router to search through the output for whatever keyword you specify and begins displaying from there. This is especially useful when you are looking for something that falls near the end of the configuration. For example:

```
Router#show run | begin ospf
router ospf 1
  router-id 100.100.100.100
  log-adjacency-changes
  redistribute connected metric 50 subnets
  network 172.31.0.0 0.0.255.255 area 0
  neighbor 172.31.8.2
  [further output omitted]
```

However, be careful because the "begin" option may start the display sooner than you wanted. For instance, in the above example, we wanted to see only the commands under OSPF router configuration. But if the word "ospf" appeared earlier in the config, such as under an interface,

then the router would have displayed the configuration beginning there. You might still end up paging through several screens to get where you wanted. In cases like that, you may want to use a combination of keywords that gets you to the unique location you desire. For example:

```
Router#show run | begin ospf
ip ospf network non-broadcast
cdp enable
```

That command resulted in the router beginning the display at an interface's configuration – the first place it found the word “ospf”. To make it skip down to the router ospf section, you can use the following command. You could also have used “show run | begin ospf 1”.

```
Router#show run | begin router ospf
router ospf 1
log-adjacency-changes
redistribute connected metric 50 subnets
network 172.31.0.0 0.0.255.255 area 0
[further output omitted]
```

| **Include** displays any line that includes the value specified. This can be useful in displaying parts of the configuration but is probably most useful when displaying the output from more dynamic show commands. Following are two examples of possible uses for this command.

In the first example, you want to see all of your frame-relay map statements. Without using the pipe, your options would be to look at the configuration for each interface, or look at the output from the “show frame-relay map” command. Neither would give you the information as succinctly as the following command:

```
Router#show run | include frame-relay map
frame-relay map ip 172.31.1.1 111 broadcast
frame-relay map ip 172.31.1.2 112 broadcast
frame-relay map ip 172.31.2.1 121 broadcast
```

As another example, suppose you need to see how much memory has been allocated to all the various OSPF processes. To avoid scrolling through pages of output, modify the command so that the router displays only the OSPF information (note that the word OSPF is in capitals in the output of the show command, and so it must be in capitals following the pipe.)

```
BBR1#show memory allocating-process totals | include OSPF
0x034C7652      65580      1  OSPF lsdB
0x034C76B8      20044      1  OSPF path
0x034D28AA      10368     24  OSPF Router
0x034D406A       3324     29  OSPF Router
[some output omitted]
```

| **Exclude** is also useful when displaying dynamic output. For example, when you want to see how loaded the router's CPU is and what processes are causing that load, you use the command “show process cpu”. But the router can have dozens of processes running on it, some of which are placing no load at all on the CPU. Using the pipe to exclude inactive processes from the output will give you more appropriate information, as shown below:

```

Router#show process cpu | excl 0.00%__0.00%__0.00%
CPU utilization for five seconds: 6%/0%; one minute: 2%; five minutes: 2%
  PID Runtime(ms)   Invoked    uSecs   5Sec   1Min   5Min TTY Process
    3     39176       2295     17070   4.58%  0.63%  0.49%  0 Check heaps
    7     4188        264     15863   0.00%  0.04%  0.03%  0 Serial Backgroun
   14     2848       2734     1041    0.08%  0.04%  0.02%  0 Net Background
   20    12880       1612     7990    0.16%  0.14%  0.16%  0 Compute load avg
   21     9728        136    71529   0.00%  0.10%  0.11%  0 Per-minute Jobs
   31     3288        446     7372    0.00%  0.04%  0.00%  0 IP Background
   45     1080        136     7941    0.00%  0.01%  0.00%  0 Adj Manager
   47     2012        135    14903   1.80%  1.61%  0.52%  0 Exec
   51      620       8057      76    0.00%  0.01%  0.00%  0 Crypto SM

```

Do Command

A wonderful command introduced in version 12.2(8)T is the **do** command. The “do” command allows you to execute EXEC commands (such as **show**, **clear**, and **debug** commands) while configuring your router. You can do EXEC-level commands from global configuration mode or any configuration submode. After the EXEC command is executed, the system will return to the configuration mode you were using.

The syntax for the “do” command is **do <command>**.

Example:

The following shows us using a derivative of the **show running-config** command in interface configuration mode. This command is normally done in privileged mode without the “do” command.

```

Router(config-if)# do show run interface atm2/0
Building configuration...

Current configuration : 133 bytes
!
interface ATM2/0
 ip address 192.1.3.254 255.255.255.0
 no atm ilmi-keepalive
 pvc 0/50
  protocol ip 192.1.3.1 broadcast
!
end
Router(config-if)#

```

Alias Commands

A neat little IOS feature that is worth writing about is the alias command. The alias command allows us to convert those long, frequently used commands to just a few keystrokes.

To create a command alias, use the **alias** command in global configuration mode. The syntax is as follows: **alias <mode> <command-alias> <original-command>**.

Example:

In the following example, we have created an alias that executes the command “show ip interface brief” by simply typing **sb**<enter>.

```
Router(config)# alias exec sb show ip interface brief
Router(config)# exit
Router# sb
```

FastEthernet0/0	10.1.1.1	YES	NVRAM	up	up
Serial0/0	unassigned	YES	NVRAM	up	up
Serial0/0.1	10.1.2.1	YES	NVRAM	up	up
FastEthernet0/1	10.2.2.1	YES	NVRAM	up	up
Serial0/1	unassigned	YES	manual	down	down
Ethernet1/0	10.3.3.1	YES	NVRAM	up	up

Privilege Levels

A useful management tool available in IOS is the one that gives you the ability to assign levels of privilege. Privilege levels are assigned to both users and commands. The privilege levels range from 0 to 15. By default, commands are assigned either level 1 or level 15. Those commands that need to be executed in privileged EXEC mode are level 15 commands. With a few exceptions, those commands that can be executed in user EXEC mode are level 1 command. A small number of commands are level 0 commands. These commands include **enable**, **disable**, **exit**, **logout**, and **help**. Level 0 commands can be executed at any level.

A user operating in privileged EXEC mode is a level 15 user. A user operating in user EXEC mode is a level 1 user. Commands and users can be assigned a privilege level different from their default. The way the privileges work is a higher level has the same rights as the lower levels beneath it. For instance, a level 10 user (if you set one up) can do everything users at levels 9 through 0 can do. Level 15 users can execute all commands.

Commands can be reassigned a different level of privilege as well. You can raise or lower the level of privilege on any command. Privilege levels on commands are assigned using the **Privilege** command; the command syntax is as follows. Use global configuration mode for this command.

privilege <mode> level <0 – 15> <command>

This feature is quite useful as it allows us to create various levels of users with custom rights to IOS commands. Imagine if you had a number of administrators with limited knowledge of the workings of IOS. But you need their help with certain specific tasks such as shutting down and re-enabling an interface or adding users to an access server. You can configure the router or access server so that depending upon the password provided, the user will be assigned a specific level of privilege and will only be allowed to use commands assigned to that level and below.

Privilege levels for users can be set in a number of ways via the IOS. They can be set permanently on a line using the **privilege level** command; at the command prompt using the **enable** command; or when logging in using the **username** command.

To set the default privilege level for a line, use the **privilege level** command in line configuration mode. The syntax is as follows: **privilege level <0 – 15>**.

To interactively reset the level of privilege from the command line, use the **enable** command. The command syntax is **enable <0 – 15>**. The command can be executed in any EXEC mode. If you leave off the level number, the router assumes you mean 15. It is advisable to set up an “enable secret” password for each level of privilege. Be sure each level’s password is different and that the passwords are only known to users within the appropriate level. The syntax for this command is **enable secret level <1-15> <password>**.

The most common way to assign levels of privilege is to do so based on the user’s username. The IOS allows you to create and use username/password pairs in your router configuration for authentication purposes. Along with the authentication process, the user can be assigned as level of privilege.

Examples:

In the example below, we have created three custom user levels using level numbers 2, 3, and 4. Level 2 users can do show commands as well as all commands in levels below. By default, level 1 users can do most show commands. But after issuing the command **privilege exec level 2 show** in our configuration, it will no longer be possible. Level 3 users are permitted to issue the command **show ip route**, but level 2 users cannot. Level 4 and above users can issue the show access-list command as well as any command in levels below.

```
Router(config)# privilege exec level 2 show
Router(config)# privilege exec level 3 show ip route
Router(config)# privilege exec level 4 show access-list
```

The example below shows us assigning level 2 to any user that enters the router via telnet. It assigns level 3 to any user that enters via the auxiliary port. And it assigns level 4 to anyone that enters via the console port.

```
Router(config)# line vty 0 4
Router(config-line)# privilege level 2
Router(config-line)# line aux 0
Router(config-line)# privilege level 3
Router(config-line)# line console 0
Router(config-line)# privilege level 4
```

The example below creates a separate password for each custom level of privilege. If a user issues the command **enable 2**, that user will be prompted for the password “twopass”. A level three request will require the level three password and so on.

```
Router(config)# enable secret level 2 twopass
Router(config)# enable secret level 3 threepass
Router(config)# enable secret level 4 fourpass
```

The example below creates three users: bob, fred, and sam. Bob is a level 2 user. Fred is a level 3 user. Sam is a level 4 user. To have the router prompt for username and password when logging in, use the command **login local** in your line configuration mode.

```
Router(config)# username bob privilege 2 password bobpass
Router(config)# username fred privilege 3 password fredpass
Router(config)# username sam privilege 4 password sampass
Router(config)# line vty 0 4
Router(config-line)# login local
```

Changing the TCP Timeout

You are telnetting from a router or switch to another device. As soon as you press Enter, you notice that you've mistyped the IP address. But there's no way to stop the telnet – you just have to wait until it times out. Make it time out quicker with the global command **ip tcp synwait-time <seconds>**. You can set the TCP session to time out anywhere from 5 to 300 seconds. One warning about this – if you lower the time the router waits when making new connections, make sure you allow enough time for your message to reach the other host and get back. Note also that this is a global command, and so it applies to all TCP sessions, not just telnet.

Stopping Runaway Debugs, Part 1

The debug command displays the actions of the router, on the console, in real time. Some debug commands produce a lot of output, and that output can appear continuously, too quickly for you to turn off the debugging. If that happens, your main recourse is to power-cycle the router – which could disrupt your network.

The typical way to turn off debugging is to put a “no” in front of the debugging command (e.g., “no debug all”). The quickest way to turn off debugging is the command **u all**, short for **undebug all**. If you need to debug a process that you think might overwhelm the router, first type in **u all**. This command will have no effect, since you aren't debugging anything yet. Then type your debug command. If the worst happens, you will only need to press the Up arrow twice and hit Enter to turn off all debugging. That's a lot quicker than typing the command, so you are more likely to be able to sneak it in between lines of output.

Stopping Runaway Debugs, Part 2

The best way to stop a runaway debug is to prevent it in the first place. The router gives a high priority to displaying the debugging output, so a debug command that results in excessive output could monopolize the router's resources. One way to prevent this is to filter what the router shows you. If the traffic you are interested in can be matched by an access list, then link an access list to your debug. This will limit the display to only the relevant items.

First, create an access list that permits only the traffic you want to see. This could be traffic from a specific host, routing updates from a particular neighbor, or traffic belonging to a certain

application. Associate that access list with the debug command as in the following example. This will filter the output through the access list and only display items matching a permit statement in the list.

```
Router#debug ip packet ?
    <1-199>      Access list
    <1300-2699>  Access list (expanded range)
```

Editing Access Lists

Traditionally, one of the biggest problems with access lists has been the inability to edit them on the fly. Any commands added to an access list are added at the bottom. You can remove a statement from a named access list, but not add one in the middle of the list. To edit an access list, you had to copy it from the config into a text editor, make the changes, remove the old list, and paste the new one in (or change the new list's name/number and paste it in.) That problem with access lists has been addressed in the newest IOS releases.

Beginning with IOS version 12.2(15)T, sequence numbers can be applied to the statements in named IP access lists. The sequence number is inserted before each permit/deny condition. By numbering your access list statements, you can edit a specific line, or remove and replace it, or add lines into the middle of the access list. If you create a named access list, the router automatically numbers the lines in increments of 10. Entries with no sequence number are still added at the bottom of the list. Line numbers are shown when you display the access list.

Example:

```
Router(config)#ip access-list extended TEST
Router(config-ext-nacl)#10 permit ip host 10.1.1.1 host 10.6.5.4
Router(config-ext-nacl)#20 permit icmp any any
Router(config-ext-nacl)#30 permit tcp any host 10.1.1.1
Router(config-std-nacl)#15 permit ip 10.5.5.0 0.0.0.255 any

Router#show access-list TEST
10 permit ip host 10.1.1.1 host 10.6.5.4
15 permit ip 10.5.5.0 0.0.0.255 any
20 permit icmp any any
30 permit tcp any host 10.1.1.1
```

Router Autoconfiguration

Autoconfiguration is a highly valuable feature for any company utilizing a WAN with one central location and many remote locations. It is a built-in feature on Cisco routers that allows for easy initial configuration of new routers in remote sites.

Using Figure 1 below as an example, you will see how the feature works. Assume you have a Central location in Richmond VA and a brand-new remote location in Arlington VA. In the Arlington location, you have a small staff but no one who is network savvy. You can configure the new Arlington router without having to travel OR without having to pre-configure and ship

the router. Instead, you can perform all of the configuration work in the comfort of your office (perhaps in Richmond).

With some prep work in Richmond, autoconfiguration can give you the flexibility to have a new (un-configured) router shipped directly to Arlington. Once received, you can simply walk anyone through the process of plugging the router in and within a few minutes it will be operational.

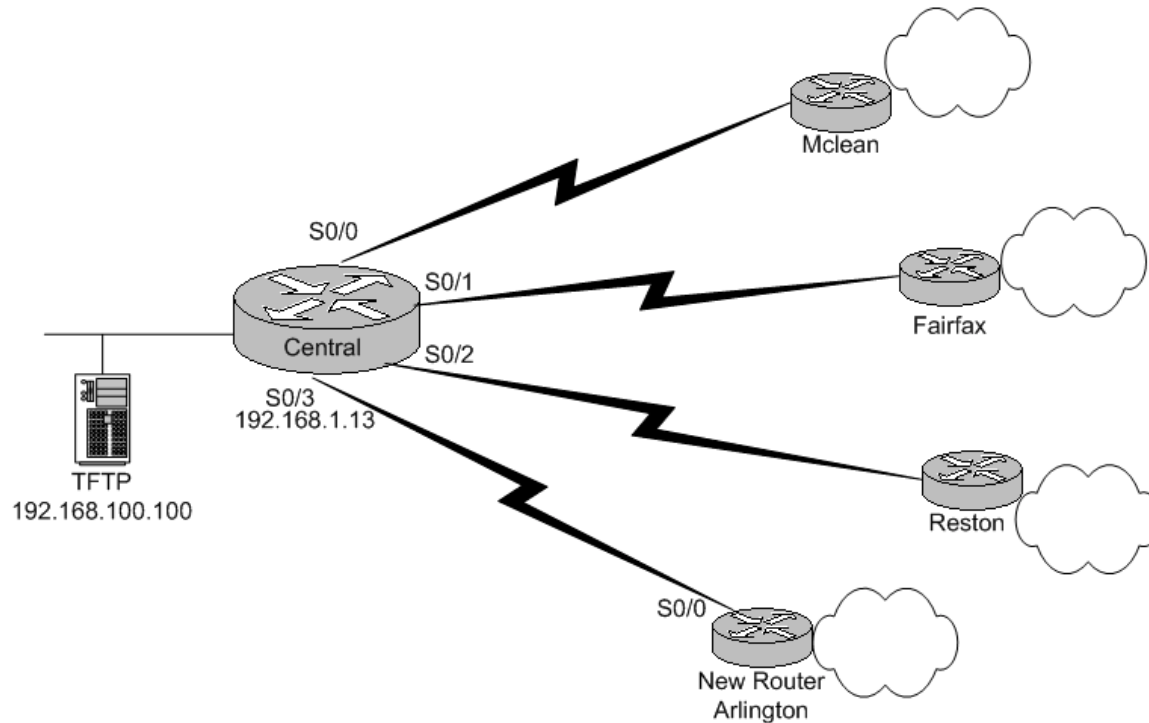
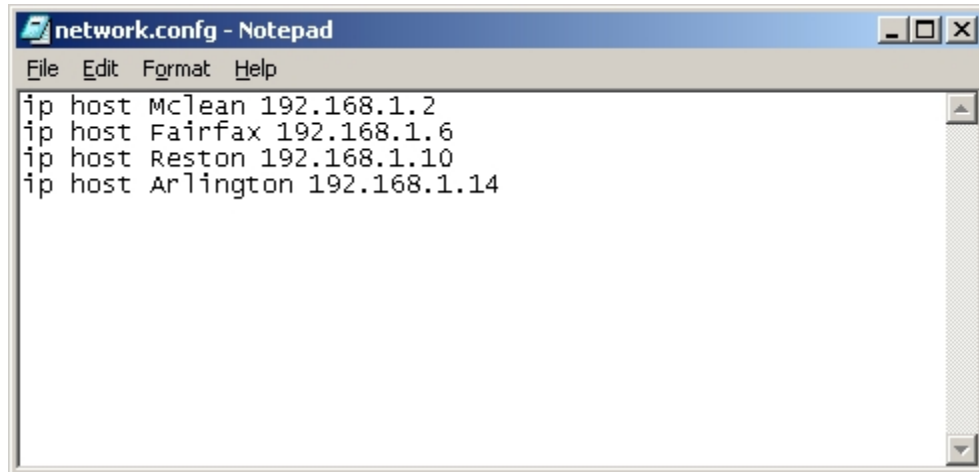


Figure 1. Wide Area Network

The process requires a couple of components, including some basic router configuration and an operational TFTP server. In Figure 1, we see a TFTP server in Richmond with the IP address 192.168.100.100. The server houses a minimum of two files: one is called **network.config** and the other is called **Arlington.config**. Network.config is an ASCII text file with nothing but Cisco IOS **ip host** commands. The job of the network.config file is much like DNS in that it allows a router to perform a reverse lookup for a router's hostname. In actuality, it is used to build a local host's table on the router. An example of one follows in Figure 2.



```
network.config - Notepad
File Edit Format Help
ip host Mclean 192.168.1.2
ip host Fairfax 192.168.1.6
ip host Reston 192.168.1.10
ip host Arlington 192.168.1.14
```

Figure 2. Example Network.config File

The other file (Arlington.config) is a full working configuration for the new router. Assuming everything works, the autoconfiguration process will ultimately deliver this file to the new router, and the new router will immediately use the configuration. As long as the configuration is valid, the Arlington site should consequently become operational.

The Arlington.config file need only be an ASCII text file with valid IOS commands. It's common practice to use the same format in your *router.config* files that would normally result from the output of a **copy running-config** command. The filename format has significance. It must correspond to one of the entries in the *network.config* file. You will notice, in figure 2, there is an entry for a router named Arlington that resolves to the IP address 192.168.1.14. This entry was put in place with the intention of auto-configuring a router with the hostname Arlington. To configure a router with the hostname Arlington, auto-configure needs to make use of a file called Arlington.config. So in summary, the filename format is **<hostname>.config**. It is advisable to create one of these for every router in your network and store them on your TFTP server.

How It Works

1. The new router's serial interface is addressed and enabled.

Shortly after booting up, a new router will administratively enable its interfaces (if possible) in an attempt to acquire IP addressing. On LAN interfaces, it will use BootP. On serial interfaces, it will use a Cisco protocol called SLARP (serial line address resolution protocol). Our example uses serial interfaces; therefore, autoconfiguration on our network will use SLARP.

SLARP is simple. You configure the Central side serial interface with either the first or second address in a block of IP addresses. If the central site router is configured with the first address in the range, the remote router will take on the second address. If the central router interface has been configured with the second address in a block, the remote will take on the first.

In our example, the new router's Serial0/0 interface will be enabled and acquire the IP address 192.168.1.14 with a mask of 255.255.255.252.

2. The new router learns its own hostname.

Once the new router has an interface enabled and addressed, it will attempt to perform a reverse lookup. The reverse lookup process can be achieved using DNS or the combination of TFTP and the network.config file. In our example, the reverse lookup will use the network.config file.

The request is sent by the new router as an IP broadcast out of its serial interface. When the Central router receives the broadcast, it is responsible for forwarding the broadcast to the TFTP server. The command used to make such a forward is the **'ip helper-address <ip-address>** command (in our example, we would use "ip helper-address 192.168.100.100"). This command changes the broadcast to a routable unicast, and the packet is routed and ultimately received by the TFTP server. The TFTP server will consequently respond with a file copy operation. It will send the network.config file to the router.

The router will merge the contents of the network.config file into the running-configuration. Since the network.config file contains "ip host" commands, the process will build a local host's table on the router. Now that the router has a local host table, it has the ability to do name resolution (and reverse resolution). The end result will be that the new router learns its own hostname, Arlington.

3. The new router copies its configuration from the TFTP server.

Now that the new router (Arlington) knows its hostname, it has the ability to formulate what file contains its configuration. It formulates the filename by taking its own hostname and concatenates it with **.config**. In the case of router Arlington, the filename formulated will be **Arlington.config**.

Arlington will send a TFTP get request in broadcast format for filename Arlington.config. As with the previous TFTP operation, the broadcast will be converted to a unicast and routed to the TFTP server. The server will respond by copying the Arlington.config file to the router. The router will merge this new configuration into its running-config and use it. Provided it was a good configuration file, the Arlington location will be operational.

Configuration

All router configuration is done on the Central router in Richmond. The remote router simply needs to have no configuration at all. The minimum configuration commands required are as follows:

```
!  
hostname Central  
!  
interface Serail0/3  
    ip address 192.168.1.1 255.255.255.252  
    ip helper-address 192.168.100.100  
!  
end  
!
```

And, as mentioned before, you will also need a TFTP server with the appropriate files saved on it.

Note: The value of this feature is in the fact that a new router can be installed by a layman and reduces the overall need for the network administrator to travel during an install. It also serves as a very good problem resolution tool. Let's assume you have a network.config file that lists an IP host entry for all of your routers. And you also have a <router>.config file containing the most current configuration for each of your routers. If one of your remote routers should fail and a new router is needed, you simply order a similar router. Have it shipped to the remote location. When it arrives, you can instruct a coworker to disconnect the old router and plug in the new one. Within a few minutes, the network will be back up and operational with little effort on your part.

Hopefully, you have learned some techniques that will make configuring, monitoring, and maintaining your Cisco equipment quicker and more efficient. Using these techniques will help make your life as a network administrator easier, which is always a good thing.