Official Cert Guide

Advance your IT career with hands-on learning

CCNA 200-301

Volume 2

Hands-On Labs

Review Exercises

Config Checklists

Study Planner

WENDELL ODOM,
CCIE® NO. 1624 EMERITUS

ciscopress.com

CCNA 200-301, Volume 2
Official Cert Guide

In addition to the wealth of updated content, this new edition includes a series of free hands-on exercises to help you master several real-world configuration activities. These exercises can be performed on the CCNA 200-301 Network Simulator Lite, Volume 2 software included for free on the companion website that accompanies this book. This software, which simulates the experience of working on actual Cisco routers and switches, contains the following 13 free lab exercises, covering ACL topics in Part I:

1. ACL I
2. ACL II
3. ACL III
4. ACL IV
5. ACL V
6. ACL VI
7. ACL Analysis I
8. Named ACL I
9. Named ACL II
10. Named ACL III
11. Standard ACL Configuration Scenario
12. Extended ACL I Configuration Scenario
13. Extended ACL II Configuration Scenario

If you are interested in exploring more hands-on labs and practice configuration and troubleshooting with more router and switch commands, go to **www.pearsonitcertification.com/networksimulator** for demos and to review the latest products for sale.

CCNA 200-301 Network Simulator Lite, Volume 2 system requirements:

Windows system requirements (minimum):

- Windows 10 (32/64-bit), Windows 8.1 (32/64-bit), or Windows 7 (32/64-bit)
- 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor
- 1 GB RAM (32-bit) or 2 GB RAM (64-bit)
- 16 GB available hard disk space (32-bit) or 20 GB (64-bit)
- DirectX 9 graphics device with WDDM 1.0 or higher driver
- Adobe Acrobat Reader version 8 and above

Mac system requirements (minimum):

- macOS 10.15, 10.14, 10.13, 10.12, or 10.11
- Intel core Duo 1.83 GHz
- 512 MB RAM (1 GB recommended)
- 1.5 GB hard disk space
- 32-bit color depth at 1024 x 768 resolution
- Adobe Acrobat Reader version 8 and above

Key Topic

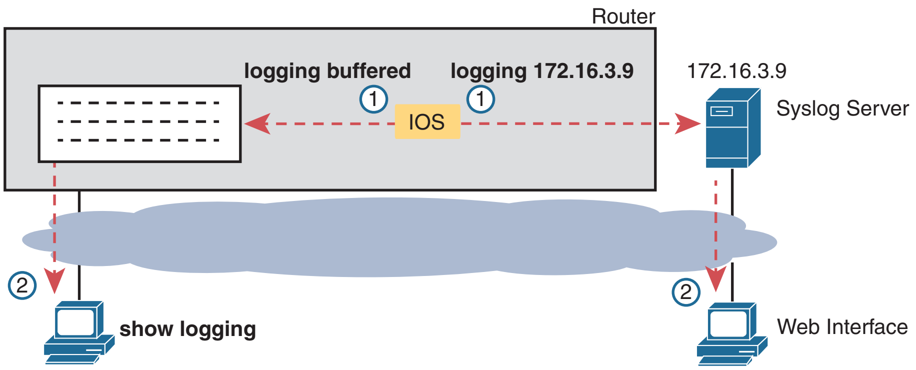

Figure 9-2 *IOS Storing Log Messages for Later View: Buffered and Syslog Server*

Log Message Format

IOS defines the format of log messages. The message begins with some data fields about the message, followed by some text more easily read by humans. For example, take a close look at this sample message:

```
*Dec 18 17:10:15.079: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state to down
```

Notice that by default on this particular device, we see the following:

A timestamp: *Dec 18 17:10:15.079

The facility on the router that generated the message: %LINEPROTO

The severity level: 5

A mnemonic for the message: UPDOWN

The description of the message: Line protocol on Interface FastEthernet0/0, changed state to down

IOS dictates most of the contents of the messages, but you can at least toggle on and off the use of the timestamp (which is included by default) and a log message sequence number (which is not enabled by default). Example 9-1 reverses those defaults by turning off timestamps and turning on sequence numbers.

Example 9-1 *Disabling Timestamps and Enabling Sequence Numbers in Log Messages*

```
R1(config)# no service timestamps
R1(config)# service sequence-numbers
R1(config)# end
R1#
000011: %SYS-5-CONFIG_I: Configured from console by console
```

To see the change in format, look at the log message at the end of the example. As usual, when you exit configuration mode, the device issues yet another log message. Comparing

Answers to the "Do I Know This Already?" quiz:

1 D **2** C **3** A **4** C **5** E, F **6** E, F this message to the previous example, you can see it now no longer lists the time of day but does list a sequence number.

Log Message Severity Levels

Log messages may just tell you about some mundane event, or they may tell you of some critical event. To help you make sense of the importance of each message, IOS assigns each message a severity level (as noted in the same messages in the preceding page or so). Figure 9-3 shows the severity levels: the lower the number, the more severe the event that caused the message. (Note that the values on the left and center are used in IOS commands.)

Keyword	Numeral	Description	
Emergency	0	System unusable	Severe
Alert	1	Immediate action required	
Critical	2	Critical Event (Highest of 3)	Impactful
Error	3	Error Event (Middle of 3)	
Warning	4	Warning Event (Lowest of 3)	
Notification	5	Normal, More Important	Normal
Informational	6	Normal, Less Important	
Debug	7	Requested by User Debug	Debug

Figure 9-3 *Syslog Message Severity Levels by Keyword and Numeral*

Figure 9-3 breaks the eight severity levels into four sections just to make a little more sense of the meaning. The two top levels in the figure are the most severe. Messages from this level mean a serious and immediate issue exists. The next three levels, called Critical, Error, and Warning, also tell about events that impact the device, but they are not as immediate and severe. For instance, one common log message about an interface failing to a physically down state shows as a severity level 3 message.

Continuing down the figure, IOS uses the next two levels (5 and 6) for messages that are more about notifying the user rather than identifying errors. Finally, the last level in the figure is used for messages requested by the **debug** command, as shown in an example later in this chapter.

Table 9-2 summarizes the configuration commands used to enable logging and to set the severity level for each type. When the severity level is set, IOS will send messages of that severity level and more severe ones (lower severity numbers) to the service identified in the command. For example, the command **logging console 4** causes IOS to send severity level 0–4 messages to the console. Also, note that the command to disable each service is the **no** version of the command, with *no* in front of the command (**no logging console**, **no logging monitor**, and so on).

Table 9-2 How to Configure Logging Message Levels for Each Log Service

Service	To Enable Logging	To Set Message Levels
Console	**logging console**	**logging console** *level-name* | *level-number*
Monitor	**logging monitor**	**logging monitor** *level-name* | *level-number*
Buffered	**logging buffered**	**logging buffered** *level-name* | *level-number*
Syslog	**logging host** *address* | *hostname*	**logging trap** *level-name* | *level-number*

Configuring and Verifying System Logging

With the information in Table 9-2, configuring syslog in a Cisco IOS router or switch should be relatively straightforward. Example 9-2 shows a sample, based on Figure 9-4. The figure shows a syslog server at IP address 172.16.3.9. Both switches and both routers will use the same configuration shown in Example 9-2, although the example shows the configuration process on a single device, router R1.

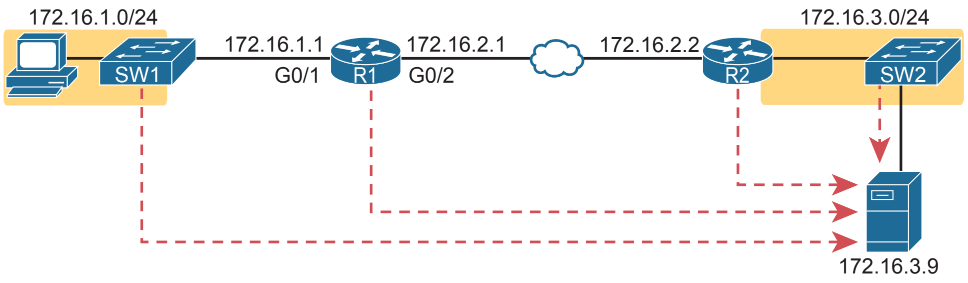

Figure 9-4 *Sample Network Used in Logging Examples*

Example 9-2 *Syslog Configuration on R1*

```
logging console 7
logging monitor debug
logging buffered 4
logging host 172.16.3.9
logging trap warning
```

First, note that the example configures the same message level at the console and for terminal monitoring (level 7, or debug), and the same level for both buffered and logging to the syslog server (level 4, or warning). The levels may be set using the numeric severity level or the name as shown earlier in Figure 9-3.

The **show logging** command confirms those same configuration settings and also lists the log messages per the logging buffered configuration. Example 9-3 shows a sample, with the configuration settings to match Example 9-2 highlighted in gray.

Example 9-3 *Viewing the Configured Log Settings per the Earlier Example*

```
R1# show logging
Syslog logging: enabled (0 messages dropped, 3 messages rate-limited, 0 flushes, 0
overruns, xml disabled, filtering disabled)


No Active Message Discriminator.


No Inactive Message Discriminator.


    Console logging: level debugging, 45 messages logged, xml disabled,
                     filtering disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level warnings, 0 messages logged, xml disabled,
                    filtering disabled
    Exception Logging: size (8192 bytes)
    Count and timestamp logging messages: disabled
    Persistent logging: disabled

No active filter modules.

    Trap logging: level warnings, 0 message lines logged
       Logging to 172.16.3.9 (udp port 514, audit disabled,
           link up),
           0 message lines logged,
           0 message lines rate-limited,
           0 message lines dropped-by-MD,
          xml disabled, sequence number disabled
          filtering disabled
       Logging Source-Interface: VRF Name:

Log Buffer (8192 bytes):
```

You might notice by now that knowing the names of all eight log message levels can be handy if you want to understand the output of the commands. Most of the **show** commands list the log message levels by name, not by number. As you can see in the gray highlights in this example, two levels list "debug," and two list "warning," even though some of the configuration commands referred to those levels by number.

Also, you cannot know this from the output, but in Example 9-3, router R1 has no buffered log messages. (Note the counter value of 0 for buffered logging messages.) If any log messages had been buffered, the actual log messages would be listed at the end of the command. In this case, I had just booted the router, and no messages had been buffered yet. (You could also clear out the old messages from the log with the **clear logging** EXEC command.)

The next example shows the difference between the current severity levels. This example shows the user disabling interface G0/1 on R1 with the **shutdown** command and then re-enabling it with the **no shutdown** command. If you look closely at the highlighted messages, you will see several severity 5 messages and one severity 3 message. The **logging buffered 4** global configuration command on R1 (see Example 9-2) means that R1 will not buffer the severity level 5 log messages, but it will buffer the severity level 3 message. Example 9-4 ends by showing that log message at the end of the output of the **show logging** command.

Example 9-4 *Seeing Severity 3 and 5 Messages at the Console, and Severity 3 Only in the Buffer*

```
R1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# interface g0/1
R1(config-if)# shutdown
R1(config-if)#
*Oct 21 20:07:07.244: %LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to
administratively down
*Oct 21 20:07:08.244: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEther-
```

```
net0/1, changed state to down
R1(config-if)# no shutdown
R1(config-if)#
*Oct 21 20:07:24.312: %LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to
up
*Oct 21 20:07:25.312: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEther-
net0/1, changed state to up
R1(config-if)# ^Z
R1#
*Oct 21 20:07:36.546: %SYS-5-CONFIG_I: Configured from console by console
R1# show logging
! Skipping about 20 lines, the same lines in Example 9-3, until the last few lines

Log Buffer (8192 bytes):

*Oct 21 20:07:24.312: %LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to
up
```

The debug Command and Log Messages

Of the eight log message severity levels, one level, debug level (7), has a special purpose: for messages generated as a result of a user logged in to the router or switch who issues a **debug** command.

The **debug** EXEC command gives the network engineer a way to ask IOS to monitor for certain internal events, with that monitoring process continuing over time, so that IOS can issue log messages when those events occur. The engineer can log in, issue the **debug** command, and move on to other work. The user can even log out of the device, and the debug remains enabled. IOS continues to monitor the request in that **debug** command and generate log messages about any related events. The debug remains active until some user issues the **no debug** command with the same parameters, disabling the debug.

NOTE While the **debug** command is just one command, it has a huge number of options, much like the **show** command may be one command, but it also has many, many options.

The best way to see how the **debug** command works, and how it uses log messages, is to see an example. Example 9-5 shows a sample debug of OSPF Hello messages for router R1 in Figure 9-4. The router (R1) enables OSPF on two interfaces and has established one OSPF neighbor relationship with router R2 (RID 2.2.2.2). The debug output shows one log message for the sent Hello, as well as log messages for received Hello messages.

Example 9-5 *Using* **debug ip ospf hello** *from R1's Console*

```
R1# debug ip ospf hello
OSPF hello debugging is on
R1#
*Aug 10 13:38:19.863: OSPF-1 HELLO Gi0/1: Send hello to 224.0.0.5 area 0 from
172.16.1.1
*Aug 10 13:38:21.199: OSPF-1 HELLO Gi0/2: Rcv hello from 2.2.2.2 area 0 172.16.2.2
```

```
*Aug 10 13:38:22.843: OSPF-1 HELLO Gi0/2: Send hello to 224.0.0.5 area 0 from
172.16.2.1
R1#
```

The console user sees the log messages created on behalf of that **debug** command after the debug command completes. Per the earlier configuration in Example 9-2, R1's **logging console 7** command tells us that the console user will receive severity levels 0–7, which includes level 7 debug messages. Note that with the current settings, these debug messages would not be in the local log message buffer (because of the level in the **logging buffered warning** command), nor would they be sent to the syslog server (because of the level in the **logging trap 4** command).

Note that the console user automatically sees the log messages as shown in Example 9-4. However, as noted in the text describing Figure 9-1, a user who connects to R1 would need to also issue the **terminal monitor** command to see those debug messages. For instance, anyone logged in with SSH at the time Example 9-4's output was gathered would not have seen the output, even with the **logging monitor debug** command configured on router R1, without first issuing a **terminal monitor** command.

Note that all enabled debug options use router CPU, which can cause problems for the router. You can monitor CPU use with the **show process cpu** command, but you should use caution when using **debug** commands carefully on production devices. Also, note the more CLI users that receive debug messages, the more CPU that is consumed. So, some installations choose to not include debug-level log messages for console and terminal logging, requiring users to look at the logging buffer or syslog for those messages, just to reduce router CPU load.

Network Time Protocol (NTP)

Each networking device has some concept of a date and a time-of-day clock. For instance, the log messages discussed in the first major section of this chapter had a timestamp with the date and time of day listed. Now imagine looking at all the log messages from all routers and switches stored at a syslog server. All those messages have a date and timestamp, but how do you make sure the timestamps are consistent? How do you make sure that all devices synchronize their time-of-day clocks so that you can make sense of all the log messages at the syslog server? How could you make sense of the messages for an event that impacted devices in three different time zones?

For example, consider the messages on two routers, R1 and R2, as shown in Example 9-6. Routers R1 and R2 do not synchronize their clocks. A problem keeps happening on the serial link between the two routers. A network engineer looks at all the log messages as stored on the syslog server. However, when the engineer sees some messages from R1, at 13:38:39 (around 1:40 p.m.), he does not think to look for messages from R2 that have a timestamp of around 9:45 a.m.

Example 9-6 *Log Messages from Routers R1 and R2, Compared*

```
*Oct 19 13:38:37.568: %OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on Serial0/0/0 from FULL
to DOWN, Neighbor Down: Interface down or detached
*Oct 19 13:38:40.568: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0/0,
changed state to down
```

```
! These messages happened on router R2
Oct 19 09:44:09.027: %LINK-3-UPDOWN: Interface Serial0/0/1, changed state to down
Oct 19 09:44:09.027: %OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on Serial0/0/1 from FULL
to DOWN, Neighbor Down: Interface down or detached
```