Doc. No. 78-0943-07
August 23, 1993

CISCO SYSTEMS

Router Products Configuration and Reference Version 9.0 Errata

This document supplies corrections and additional information for the 9.0 version of the Cisco publication *Router Products Configuration and Reference* and replaces any prior versions. Keep this document with the *Router Products Configuration and Reference* manuals for future reference.

Corrections to Chapter 2, "First-Time Startup and Basic Configuration"

Add the following section to page 2-3, after the section "Using the Setup Command Facility."

The streamlined **setup** command facility enables you to continue to boot your system even though there may be problems with the configuration file when you are attempting to netboot an image.

The router enters the streamlined **setup** command facility under the following circumstances:

- You issued a **write erase** command, thereby deleting the configuration file in nonvolatile memory.
- Your configuration in NVRAM has been corrupted.
- If you set bit 15, the diagnostic bit, and answered "no" in response to the "set test addresses" prompt.

Corrections to Chapter 3, "Using Terminals"

In the section "Using the DEC MOP Server" on page 3-10, add the sections that follow.

Enabling MOP for an Interface

To control whether MOP is enabled for an interface, use the **mop enabled** interface subcommand.

mop enabled
no mop enabled

The default is enabled. Use the **no mop enabled** command to disable MOP.

Controlling the MOP System ID Messages

To control whether MOP periodic system ID messages are sent out to an interface, use the **mop sysid** interface subcommand.

mop sysid
no mop sysid

You can still run MOP without having the background system ID messages sent out. This lets you use the MOP remote console, but does not generate messages used by the configurator.

Use the **no mop sysid** to disable MOP from sending the system ID messages.

Corrections to Chapter 4, "Configuring the System"

Add the following information after the section "Obtaining the Boot File over the Network" on page 4-7.

Manually Booting from ROM

Use the **b** command at the ROM monitor prompt (>) to manually boot the system from the ROM software. The syntax is as follows:

b

The following is an example of the **b** command output:

```
>b
F3:
{ROM Monitor copyrights}
```

In the second example on page 4-5, in the section "Dynamic Buffer Sizing," the following example is incorrect:

```
buffers medium max-free 200
```

The correct example follows:

```
buffers middle max-free 200
```

Add the following subsection after the section "Obtaining the Boot File over the Network" on page 4-7.

Manually Netbooting

Use the **b** command at the ROM monitor prompt (>) to manually netboot the system, as in the following example. Check the appropriate hardware manual for the correct jumper or configuration register setting. The syntax for TFTP netbooting is as follows:

b *filename* [*address*]

The *filename* argument specifies the filename of the image you want loaded. It is case sensitive. The *address* argument is optional and defines the IP address of the host you want to boot from. The following is an example of the **b** command for manually netbooting:

```
>b testme4.test 131.108.15.112
F3:
{ROM Monitor copyrights}
```

Add the following section after the "Configuring Multiple Instances of the Boot Commands" on page 4-9.

Troubleshooting Information when Netbooting

Cisco routers support netbooting over both TFTP and MOP across all supported media types such as Ethernet, FDDI, serial, Token Ring, and HSSI. During a netbooting session, routers behave like hosts; they route via proxy ARP or a default gateway. However, when netbooting, a router ignores routing information, static IP routes, and bridging information. As a result, intermediate routers are responsible for handling ARP and TFTP requests correctly. For serial and HSSI media, ARP is not used.

If you need to netboot from a server, it is recommended that you ping the server from the ROM software. If you are unable to ping the server, there is a problem with the server configuration or hardware. Contact your technical support representative for assistance. See "Useful Information to Provide Technical Support" later in this section for details.

The list that follows contains solutions to common problems that occur when netbooting. Note that these solutions only apply if you were able to successfully ping the server.

Client ARP Request Times Out

When netbooting, the client you netboot from sends an ARP request to the server over every available appropriate network interface (such as an Ethernet port or a Token Ring port). The client expects the server or a router to return an ARP response. If the client does not receive an ARP response from the server or a router, a message similar to the following displays at the client console:

```
Booting gs3-bfx.................[timed out]
```

One possible cause of this message is that intermediate routers are not performing proxy arp. Look for **no ip proxy-arp** in the configuration of the intermediate router. Another possible reason for this message appearing at the client is that the client is using a broadcast address, and the intermediate router does not have an IP helper address defined to point to the TFTP server.

Timeouts and Out-of-Order Packets

When netbooting, it is not unusual for the client to send additional requests before receiving a response to the initial ARP request. This can result in timeouts, out-of-order packets, and multiple responses. Timeouts (shown as periods on a netbooting display) and out-of-order packets (shown as Os) do not necessarily prevent a successful boot. It is acceptable to have timeouts and out-of-order packets. The following examples show successful boots even though a timeout and out-of-order packets have occurred:

```
Booting gs3-bfx from 131.108.1.123: !.!!!!!!!!!!!!!!!!!!!!
```

```
Booting gs3-bfx from 131.108.1.123: !0.0!!!!!!!!!!!!!!!!!!!!
```

Note that intermittent timeouts and out-of-order packets may occur throughout a netbooting session without being cause for concern. Excessive timeouts and out-of-order packets can be caused by bad routing paths on the intermediate routers, an extremely slow server, problems caused by multiple paths, or noise on the line. If your netbooting session appears to have excessive timeouts and out-of-order packets, contact your technical support representative and report the problem. Before calling technical support, you will need to gather some information. See "Useful Information to Provide Technical Support" that follows for details.

Useful Information to Provide Technical Support

Collect the following information for the technical support representative:

- ROM images
- NVRAM configurations for client and adjacent routers
- Debugging output from the adjacent router using the following commands:

— **debug arp**

— **debug ip-udp**

Change the last sentence in the first note on page 4-13 to read, "A failure message, `buffer overflow - xxxx/xxxx`, will appear, where xxxx/xxxx is the number of bytes used/total number of bytes."

Add the following information to the section "Manually Booting from Flash Memory" on page 4-16.

boot flash [*filename*]

The optional *filename* argument specifies the filename of the image you want loaded. It is case sensitive.

Add the following information to the section "Storing and Booting System Software Using the Flash Memory Card" in Chapter 4. It should be placed after the subsection on page 4-12 titled "Copying the TFTP Image to Flash Memories."

Configuring Flash Memory as a TFTP Server

In the description that follows, one router with a Flash Memory card installed will be referred to as the Flash server; all other routers will be referred to as client routers. The configurations for the Flash server and client routers will be given through example configurations of each, with commands included as necessary.

Prerequisites

The Flash server and client router must be able to reach one another before the TFTP function can be implemented. Verify this connection by pinging between the Flash server and client router (in either direction) using the **ping** command.

The syntax for the **ping** command is as follows:

```
Router# ping 131.131.101.101 <Return>
```

In this example, the Internet Protocol (IP) address of 131.131.101.101 belongs to the client router. In response to this command, `!!!!!` indicates a connection, while `... [timed out]` or `[failed]` indicates none. If the connection fails, reconfigure the interface, check the physical connection between the Flash server and client router, and ping again.

After this connection is verified, ensure that a TFTP-bootable image is present in Flash memory. This is the system software image that the client router will boot. Note the name of this software image so that it can be verified after the first client boot (the filename gs3-bfx.90.1 will be used in this example).

Note: The filename used must represent a software image that is present in Flash memory. If no image resides in Flash memory, the client router will boot the server's ROM image as a default.

Caution: The type of software (bfx, and so forth) residing in the Flash memory *must* be of the same type as the ROM software installed on the client router. For example, if the client router has gs3-bfx.90.1 (capable of X.25 bridging) in ROM and gs3-bf.90.1 is booted from the Flash server, the client router will operate under the control of the gs3-bf.90.1 software and will not have X.25 bridging capability.

Once you have verified the presence of a bootable image in Flash memory, you can configure the Flash server.

Configuring the Flash Server

Configure the Flash server by adding both the **tftp-server system** command and the **access-list** command to the configuration memory. Use the **configure terminal** command to do so.

Sample output of these commands follows:

```
Server# configure terminal
   Enter configuration commands, one per line.
   Edit with DELETE, CRTL/W, and CRTL/U;end with CTRL/Z
   tftp-server system gs3-bfx.90.1 1
   access-list 1 permit 131.131.101.0 0.0.0.255
^Z
Server# write memory <Return>
[ok]
```

This example gives the filename of the software image in the Flash server and one access list (labeled 1). The access list must include the network within which the client router resides. Thus, in the example, the network 131.131.101.0 and any client routers on it are permitted access to the Flash server filenamed gs3-bfx.90.1. For this exercise, the IP address of the Flash server is 131.131.111.111.

For more information on access lists, refer to page 13-23.

Caution: Using the **no boot system** command in the following example will invalidate *all* other boot system commands currently in the client router system configuration. Before proceeding, determine whether the system configuration stored in the router you will use as the client should first be saved (uploaded) to a TFTP file server. Refer to Chapter 2 for instructions on uploading and downloading system configuration files.

Configuring the Client Router

Configure the client router using the **no boot system** command, the **boot system** command, and the **boot system rom** command. Use the **configure terminal** command to enter these commands into the client router's configuration memory. Using these commands requires changing the jumper on the processor's configuration register to the pattern 0-0-1-0 (Position 1).

Following is an example of the use of these commands:

```
Client# configure terminal
Enter configuration commands, one per line.
Edit with DELETE, CRTL/W, and CRTL/U;end with CTRL/Z
no boot system
boot system gs3-bfx.90.1 131.131.111.111
boot system rom
^Z
Client# write memory <Return>
[ok]
Server# reload
```

In this example, the **no boot system** command invalidates all other **boot system** commands currently in the configuration memory, and any **boot system** commands entered after this command will be executed first. The second command, **boot system** *filename address*, tells the client router to look for the file gs3-bfx.90.1 in the (Flash) server with an IP address of 131.131.111.111. Failing this, the client router will boot from its system ROM upon the **boot system rom** command, which is included as a backup in case of a network problem.

Caution: The system software (gs3-bfx.90.1) to be booted from the Flash server (131.131.111.111) *must* reside in Flash memory on the server. If it is not in Flash memory, the client router will boot the Flash server's system ROM.

Use the **show version** command on the client router to verify that the software image booted from the Flash server is the image present in Flash memory.

Following is sample output of the **show version** command:

```
Client# show version
GS Software (GS3-BFX), Version 9.0(1), CISCO SYSTEMS SOFTWARE
Copyright (c) 1986-1992 by cisco Systems, Inc.
Compiled Mon 30-Mar-92 17:16

System Bootstrap, Version 4.5(0.5), CISCO SYSTEMS SOFTWARE

Client uptime is 5 minutes
System restarted by reload
System image file is "gs3-bfx.90.1", booted via tftp from
131.131.111.111
```

The important information in this example is contained in the first line (GS Software...), and the last full line (System image file...). The two software types and versions shown indicate the software currently running in RAM in the client router (first line) and the software booted from the Flash server (last line). These two types and versions must be the same.

Note: If no bootable image was present in the Flash server memory when the client server was booted, the version currently running (first line of the above example) will be the system ROM version of the Flash server by default.

Verify that the software shown in the first line of the previous example is the software residing in the Flash server memory.

Additional TFTP Information

In the event of a Flash memory load failure, the following error message is displayed:

```
Error programming flash memory
```

If you see this message, contact customer service immediately and inform them of the situation.

Note: Images that run from ROM, which includes IGS images and images for the CSC-2 processor, cannot be loaded over the network.

Additionally, you can perform the following steps, which might recover from the error. These steps can be repeated.

Step 1: If possible, inspect the cable connections between the Flash card and the host card. Also, make sure that the host card is securely seated in its Multibus slot.

Step 2: Erase the Flash memory and try to download the file again. To erase the Flash memory, press Return or type y at the `Erase flash before writing? [confirm]` prompt.

On page 4-18, replace the last paragraph of the **enable password** description with the paragraphs that follow.

When you use the **enable** command at the console terminal, the EXEC does not prompt you for a password if the privileged mode password is not set. Additionally, if the enable password is not set and the line 0 (console line) password is not set, it is only possible to enter privileged mode on the console terminal. This feature allows you to use physical security rather than passwords to protect privileged mode if that is what you prefer to do.

If the enable password is not set and the line 0 (console) password is set, it is possible to enter privileged command mode either without entering a password at the console terminal or by entering the console line password when prompted while using any other line.

On page 4-18, the example at the bottom of the page should be changed to read as follows:

```
enable password secretword
```

On page 4-26, replace the paragraph that begins "The *encryptiontype* argument is a single-digit number..." with the paragraphs that follow.

The *encryptiontype* argument is a single-digit number that defines whether the text immediately following is encrypted and, if so, what type of encryption is used. Currently defined encryption types are 0, which means that the text immediately following is not encrypted, and 7, which means that the text is encrypted using a Cisco-defined encryption algorithm. A *password* can contain embedded spaces and must be the last option specified in the **username** command.

When you specify an encryption type of 0 to enter an unencrypted password, the system displays the encrypted version of the password. For example, suppose you enter the following command:

```
username bill password westward
```

The system would display this command like this:

```
username bill password 7 21398211
```

The encrypted version of the password is 21398211. The password was encrypted by the Cisco-defined encryption algorithm, as indicated by the "7."

If you were to enter the following command, the system would assume that the password is already encrypted and would do no encryption. It would display the command exactly as you typed it:

```
username bill password 7 21398211
username bill password 7 21398211
```

On page 4-27 add the sections that follow.

Setting the System Contact String

To set the system contact string (syscontact), use the **snmp-server contact** command. The command syntax follows:

snmp-server contact *text*

The *text* argument is a string that specifies the system contact information.

Setting the System Location String

To set the system location string, use the **snmp-server location** command. The command syntax follows:

snmp-server location *text*

The *text* argument is a string that specifies the system location information.

Add the note that follows to the example on page 4-38.

Note: Many UNIX systems require a tab character to be used as the "white space" separator in the /etc/syslog.conf file. Use of a space character rather than a tab may cause the entry in /etc/syslog.conf to be ignored.

On page 4-38, the following statements are incorrect:

"The auxiliary ports assert DTR only when a Telnet connection is established. The console port does not use RTS/CTS handshaking for flow control."

The correct statements follow.

"The auxiliary port asserts DTR when a Telnet connection is established or when an EXEC becomes active. Flow control is not supported on either the auxiliary port or the console port."

On page 4-38, before the section "Establishing Line Passwords," add the following section, "Disabling Automatic Connections":

Disabling Automatic Connections

To disable automatic connections, use the following line subcommand:

transport preferred none

This command prevents the system from assuming that any unrecognized command is a host name. No connection will be attempted if the command is not recognized (by misspelling a command, for example).

Corrections to Chapter 5, "Managing and Monitoring the System"

Add the following description of the CPU utilization field to Table 5-4, "Show Process Field Descriptions," on page 5-5:

The CPU utilization field provides a general idea of how busy the processor is. It is a ratio of the current idle time over the longest idle time. This information should be used as an estimate only.

Corrections to Chapter 6, "Configuring the Interfaces"

Throughout Chapter 6, the descriptions of the Five minute input rate and Five minute output rate fields incorrectly define the information as providing average *bytes* per second transmitted. Instead, the descriptions should state that the information displayed shows the average *bits* per second transmitted.

The following is a typical **show interfaces ethernet** display:

```
Ethernet 0 is up, line protocol is up
  Hardware is MCI Ethernet, address is aa00.0400.0134 (bia 0000.0c00.4369)
  Internet address is 131.108.1.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:01, output hang never
  Last clearing of "show interface" counters never
  Output queue 0/40, 27 drops; input queue 0/75, 0 drops
  Five minute input rate 4000 bits/sec, 3 packets/sec
  Five minute output rate 6000 bits/sec, 7 packets/sec
     8010711 packets input, 755061207 bytes, 0 no buffer
     Received 1825365 broadcasts, 1 runts, 0 giants
     9 input errors, 9 CRC, 1 frame, 0 overrun, 0 ignored, 0 abort
     25561602 packets output, 3064506850 bytes, 0 underruns
     12 output errors, 417136 collisions, 5 interface resets, 0 restarts
```

The following is the correct description for these fields.

Field	Description
Five minute input rate, Five minute output rate	Average number of bits and packets transmitted per second in the last five minutes.

This change affects descriptions in the following tables:

- Table 6-2 (page 6-11)
- Table 6-3 (page 6-16)
- Table 6-4 (page 6-22)
- Table 6-6 (page 6-31)
- Table 6-9 (page 6-41)
- Table 6-10 (page 6-45)

On page 6-8, the list of serial encapsulation methods should include Synchronous Data Link Control (SDLC) and serial tunnel (STUN). The following is the complete list of serial encapsulation. The complete list of *encapsulation-type* arguments for the **encapsulation** command follows the list.

- High-level Data Link Control (HDLC)
- HDLC Distant Host (HDH)
- Frame Relay
- Point-to-Point Protocol (PPP)
- Synchronous Data Link Control (SDLC)
- Switched Multimegabit Data Services (SMDS)
- Cisco Serial Tunnel (STUN)
- X.25-based encapsulations

The *encapsulation-type* argument of the **encapsulation** command is a keyword that identifies one of the following serial encapsulation types that the software supports:

- **bfex25**—Blacker Front End Encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation
- **frame-relay**—Frame Relay
- **hdh**—HDH protocol
- **hdlc**—HDLC protocol
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation
- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-Point Protocol (PPP)
- **sdlc-primary**—IBM serial SNA
- **smds**—SMDS service
- **stun**—STUN protocol function
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

On page 6-10, note that the sample output for **show interfaces serial** command is for an HDLC synchronous serial interface.

On page 6-12, in Table 6-2, the description of the collisions field is missing. The following is the description:

Collisions—Number of messages retransmitted due to an Ethernet collision. This usually is the result of an overextended LAN (Ethernet or transceiver cable too long, more than two repeaters between stations, or too many cascaded multiport transceivers). A packet that collides is counted only once in output packets.

In Table 6-6, on page 6-31, replace the second sentence in the description of frame errors with the following text:

On FDDI, frame errors might result from a failing fiber or from hardware malfunctions.

In the description of the **dialer fast-idle** interface subcommand on page 6-54, the third paragraph from the bottom of the page should have been removed. The paragraph to be ignored from this description follows.

- This timer is used when an interface is connected, and a packet is received for a different destination. If the duration specified for the **dialer fast-idle** command is exceeded and no traffic is detected on the connection, the link is disconnected and a dial-on-demand connection to the dial string/destination address associated with the queued packet is attempted (assuming the packet passes any applicable access list included in the system configuration). This allows less idle time on an interface with a great deal of contention.

The paragraph in the document that follows immediately after this paragraph provides the correct, amended information, and reads as follows:

- This timer is used when an interface is connected, and a packet is received for a different destination (phone number). If the duration specified for the **dialer fast-idle** command is exceeded and no traffic is detected on the connection, the link is disconnected. This makes the port available for making connections to an alternate destination (after the **dialer enable-timeout** period expires) and reduces idle time on an interface.

The **interface** command description on page 6-67 currently reads, “Specifies a serial interface.” In fact, this command can be used to configure any type of interface supported on your router platform. It should read as follows:

interface *type unit*

Specifies an interface.

Corrections to Chapter 7, "Adjusting Interface Characteristics"

Make the following corrections to the command syntax listed in the section "Assigning Priority by Protocol Type" on page 7-5:

priority-list *list* **protocol** *protocol-name queue-keyword* [*args*]
no priority-list *list* **protocol**

The statement below the note on page 7-6 is incorrect. The correct statement follows:

Use the **no priority-list** global configuration command followed by the appropriate list argument and the **protocol** keyword to remove a priority list entry assigned by protocol type.

On page 7-8 under "Assigning a Default Priority," change the last sentence to read as follows:

The **normal** queue is assumed if you use the **no** form of the command.

On page 7-9, the syntax of the **no priority-group** command is incorrect, as is the description following the command syntax. Replace all the text regarding this command with the following text:

priority-group *list*
no priority-group

The argument *list* is the priority list number assigned to the interface. Only one list can be assigned per interface. The **no** version of the command removes the specified **priority-group** assignment.

On page 7-10, add this section before "Controlling Interface Hold Queues":

Configuring Transmit Queue Limits on CSC-MCI and CSC-SCI Cards

The interface subcommand **tx-queue-limit** controls the size of the transmit queue available to a specified interface on the MCI and SCI cards themselves. The command syntax follows:

tx-queue-limit *number*

This command should be used only under the guidance of technical support representatives. *Number* is the number of packets that may be queued for transmission on the card for the specified interface.

Example

```
tx-queue-limit 5
```

Corrections to Chapter 8, "Configuring Packet-Switched Software"

On page 8-4, add the following information before the section, "Monitoring and Troubleshooting LAPB."

To define the number of packets to be held on an interface, use the **lapb hold-queue** interface subcommand:

lapb hold-queue *queue-size*
no lapb hold-queue [*queue-size*]

The argument *queue-size* defines the number of packets. By default, this number is 10. Use the **no lapb hold-queue** command without an argument to remove this command from the configuration file. Enter the command with a *queue-size* of zero to allow an unlimited number of packets.

Replace the example at the top of page 8-11 with the example that follows:

```
! Configuration for Cisco A
! -------------------------
int s 1
ip 131.108.170.1 255.255.255.0
x25 address 31370054068
x25 map ip 131.108.170.3 31370054065
x25 map ip 131.108.170.4 31370054065

int s 2
ip 131.108.170.2 255.255.255.0
x25 address 31370054069
x25 map ip 131.108.170.4 31370054067
x25 map ip 131.108.170.3 31370054067

! allow either source address
x25 route 31370054069 alias Serial1
x25 route 31370054068 alias Serial2


! Configuration for Cisco B
! -------------------------
int s 0
ip 131.108.170.3 255.255.255.0
x25 address 31370054065<<<<<<<
x25 map ip 131.108.170.1 31370054068
x25 map ip 131.108.170.2 31370054068

int s 3
ip 131.108.170.4 255.255.255.0
x25 address 31370054067<<<<<<<
x25 map ip 131.108.170.2 31370054069
x25 map ip 131.108.170.1 31370054069

! allow either source address
x25 route 31370054067 alias Serial0
x25 route 31370054065 alias Serial3
```

On page 8-12, the discussion of TCP header compression support is incorrect. TCP header compression over X.25 is not supported in the initial release of 9.0.

Table 8-3 on page 8-18 incorrectly describes the Class B address convention as follows:

Class B:Net.Net.Host.LH.PSN

The correct description follows.

Class B:Net.Net.Host.PSN

On pages 8-21 and 8-22, replace the section "Constructing the X.25 Routing Table" with the section that follows.

The X.25 routing table is consulted when an incoming call is received that should be forwarded to its destination. Two fields are used to determine the route: the called X.121 network interface address or the destination host address, and the X.25 packet's Called User Data (CUD) field. When the destination address and the CUD of the incoming packet fit the X.121 and CUD patterns in the routing table, the packet is forwarded.

An entry in the X.25 routing table is set up or removed with the **x25 route** global configuration commands. The full syntax and variations of these commands follow:

x25 route [# *position*] *x121-pattern* [**cud** *pattern*] **interface** *interface-type unit*
no x25 route [# *position*] *x121-pattern* [**cud** *pattern*] **interface** *interface-type unit*

x25 route [# *position*] *x121-pattern* [**cud** *pattern*] **ip** *ip-address*
no x25 route [# *position*] *x121-pattern* [**cud** *pattern*] **ip** *ip-address*

x25 route [# *position*] *x121-pattern* [**cud** *pattern*] **alias** *interface-type unit*
no x25 route [# *position*] *x121-pattern* [**cud** *pattern*] **alias** *interface-type unit*

The order in which X.25 routing table entries are specified is significant; the list is scanned for the first match. The optional argument *# position* (# followed by an integer) designates the line number of an existing entry. The new entry will be inserted after the existing entry indicated by the *position* argument. If no *position* parameter is given, the entry is appended to the end of the routing table.

The argument *x121-pattern* can be either an actual X.121 destination address or a regular expression representing a group of X.121 addresses (for example, **1111.***).

The optional Call User Data pattern can also be specified as a printable ASCII string. Both the X.121 address and Call User Data can be written using UNIX-style regular expressions. The Call User Data field is matched against the data that follows the protocol identification field, which is four bytes. The argument *interface-type* is the type of interface (for example, serial) and *unit* is the interface unit number. For connections routed through a LAN, the *ip-address* argument is the IP address of the network interface or DTE. Use the **show x25 route** command to display the X.25 routing table.

In the **alias** version of this command, the *type* argument is the type of interface and the *unit* argument is the unit number of the destination interface on the destination router. With the **alias** command, if a call comes in on the specified interface and the call's destination address fits the X.121 pattern, the call is received on the destination interface. In other words, an alias route is valid only for calls that come in on the named interface.

Enter the **no x25 route** command with the appropriate arguments and keywords to remove the entry from the table.

Examples

In the following example, if a call comes in on interface serial 0 and matches any x121-pattern, the call will be accepted for the type of connectivity configured for the interface and the CUD.

```
x25 route .* alias serial 0
```

In the following example, the call will be accepted because both this vax-x.121 address and the address given in the **x25 address** interface command will be treated as local addresses for interface serial 0.

```
x25 route vax-x121-address alias serial 0
```

Add the following text at the end of the section "Translating X.25 Called Addresses," which starts on page 8-22:

Note that address substitution is only performed on routes to an interface. When running X.25 over IP, address substitution can be performed on the destination IP system if the destination system is configured with the appropriate X.25 routing commands.

On page 8-34, the default value for the **x25 hold-queue** command should be 10, not zero. A hold-queue value of 0 allows an unlimited number of packets in the hold queue.

The **show x25 vc** command output on page 8-37 provides an additional line of information in the output display.

```
Window is closed
```

This statement shows when a full packet-level window has been transmitted and is waiting for acknowlegement from the other end of the virtual circuit in order to transmit the next full window. Flow control is being used on the virtual circuit.

The **show interface serial** command output on page 6-10 provides an additional line of information in the output display.

```
Window is closed
```

In this case, the statement shows when a full link-level frame has been transmitted and is waiting for acknowledgment from the other end of the virtual circuit in order to transmit the next full link-level frame. Flow control is being used on the serial interface.

In the first paragraph on page 8-49, the acronym LMI is incorrectly defined as Logical Management Interface. The correct definition is Local Management Interface.

On page 8-58, the **debug frame-relay-packets** command should be changed to the following syntax:

debug frame-relay-packet

Add the following explanation of the **packetsize** keyword to all X.25 commands that contain this keyword. This includes the following commands: **x25 map** (page 8-9), **x25 pvc** (pages 8-13 and 8-24), **x25 ips** (page 8-33), **x25 ops** (page 8-33), and **x25 facility** (page 8-36).

The only valid packet size values are 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096.

On page 8-76, the first sentence of the summary description of **x25 idle** *minutes* should read, "Clears an SVC after a set period of inactivity."

Corrections to Chapter 10, "Routing AppleTalk"

On page 10-9, Figure 10-3 incorrectly shows the routing table for Router R3. Change the last line of the R3 Routing Table in that figure to read:

514R_2

On page 10-16, add the **no** form of the interface subcommand **appletalk iptalk** *net.node zone.* The command description is as follows:

Use the **no appletalk iptalk** command to disable IPTalk encapsulation on the interface.

On page 10-17 add the following information:

The **appletalk iptalk-baseport** command has a **no** form. Use the **no** form of the **appletalk iptalk-baseport 768** command to return to the default setting.

On page 10-23, in the section "Configuring MacIP," add the following item to the bulleted list at the bottom of the page:

- If you are using MacIP to allow Macintoshes to communicate with IP hosts on the same LAN segment (that is, the Macintoshes are on the Cisco interface on which MacIP is configured) and the IP hosts have extended IP access lists, these access lists should include entries to permit IP traffic destined for these IP hosts (from the MacIP addresses). If these entries are not present, packets destined for IP hosts on the local segment will be blocked (that is, they will not be forwarded).

On page 10-28, in the section "Zone-Based AppleTalk Access Control," change the sentence "Cisco routers permit access and routing to be controlled using zone names—stated either explicitly or using generalized argument keywords" to the following:

Cisco routers permit routing to be controlled using zone names—stated either explicitly or using generalized argument keywords.

On page 10-31, in the section "Assigning an Access List to an Interface," change the first sentence to the following:

A *packet filter*, specified via the **appletalk access-group** interface subcommand, prevents any packets from being sent out an interface if the source network has access denied.

In the same section, add the following paragraph:

Access lists applied using the **access-group** interface subcommand must permit the network number of the outgoing interface.

On page 10-31, add the following note:

Note: A zone-specific ACL will have no effect when applied using the **appletalk access-group** interface subcommand.

On page 10-60, in the example for configuring the access lists on Router C, access list 620, which is referenced by the **appletalk distribute-list in** and **appletalk distribute-list out** commands, should not contain any zone names; it should contain only network numbers. This is because, as stated on page 10-32, a distribution list is a list of AppleTalk access list numbers that controls whether the network numbers specified by the access list are processed during the reception or transmission of routing updates.

On page 10-66, under "Displaying MacIP Status," change the sentence "MacIP traffic statistics are displayed under the **show appletalk traffic** command" to the following:

"MacIP traffic statistics are displayed via the **show apple traffic** and the **show apple macip-traffic** commands."

On page 10-69, in the last paragraph under "Monitoring MacIP Clients," replace the word "lase" with the word "last."

On page 10-69, under "Monitoring MacIP Traffic," change the command in the first sentence to **show apple traffic**. In this same section add the following command information:

Use the **show apple macip-traffic** command to obtain a detailed breakdown of MacIP traffic that is sent through a gateway from AppleTalk to IP through Cisco routers. The output from this command is different from the output of the **show apple traffic** command, which shows normal AppleTalk traffic generated, received, or routed by Cisco routers. The command syntax is as follows:

show apple macip-traffic

On page 10-73, in the section "Displaying the Network Routing Table," add the following paragraph:

When an AppleTalk route is poisoned by another router, its metric gets changed to poisoned (that is, 31 hops). The router will then age this route normally, during a hold-down period, when it will still be visible in the routing table with a distance of poisoned, or 31 hops.

On page 10-80, the **ping nbp help** command example display does not show all commands. The following is what the output looks like:

```
nbptest> help
Tests are:

lookup:     lookup an NVE.  prompt for name, type and zone
parms:      display/change lookup parms (ntimes, nsecs, interval)
zones:      display zones
poll:       for every zone, lookup all devices, using default parms
help|?:     print command list
confirm:    confirm an NVE.  prompt for name, type, zone, address
addclients: add a range of fake MACIP clients
delclients: delete a range of fake MACIP clients
register:   register an NVE.  prompt for name, type, zone
unregister: unregister an NVE.  prompt for name, type, zone
stats:      dump appletalk stats changed since last dump
debug:      set/unset debug switches
quit:       exit nbptest
```

On page 10-79, "Maintaining the AppleTalk Network," the following two commands are not supported:

clear appletalk neighbors
clear appletalk routes

On page 10-83, the entry for the **debug appletalk** command should be removed. The **debug appletalk** command does not exist.

Also on page 10-83, the command syntax is listed incorrectly. Following is the correct syntax:

debug apple-arp

Corrections to Chapter 12, "Routing DECnet"

On page 12-4 in the section "Configuring DECnet Routing," add the note that follows.

Note: If you plan to use DECnet and Novell routing concurrently on the same interface, you must enable DECnet routing first and then enable Novell routing without specifying the optional MAC address. If Novell routing is enabled before DECnet routing, Novell routing will be disrupted.

On page 12-13 replace the section "Configuring DECnet Access Lists" with the following:

Use the **access-list** global configuration command to create an access list.

access-list *list* {**permit**|**deny**} *source source-mask*
no access-list *list*

The argument *list* is an integer you choose between 300 and 399 that uniquely identifies the access list. The **permit** and **deny** keywords decide the access control action when a match occurs with the address arguments.

The standard form of the DECnet access list has a DECnet *source* followed by a *source-mask*, also in DECnet address format, with bits set wherever the corresponding bits in the address should be ignored. DECnet addresses are written in the form *area.node* (for example, 50.4 is area 50, node 4). All addresses and masks are in decimal form.

Note: In contrast with IP masks, a DECnet mask specification of "all ones" is entered as the decimal value 1023. In IP, the equivalent is 255.

Example

This example sets up access list 300 to deny packets coming from node 4.51 and permit packets coming from 2.31.

```
!
access-list 300 deny 4.51 0.0
access-list 300 permit 2.31 0.0
!
```

On page 12-15 in the section "DECnet Connect Initiate Filtering," add the table of DECnet object numbers that follows.

Table 12-1 Common DECnet Object Numbers

Name	Number	Description
FAL	17	File Access Listener
HLD	18	Host Loader
NML	19	Network Monitor Link/NICE
MIRROR	25	Loopback mirror
EVL	26	Event logger
MAIL	27	Mail
PHONE	29	Phone
NOTES	33	VAX Notes
CTERM	42	Terminal sessions
DTR	63	DECnet Test Sender/Receiver

On page 12-17, remove the last three paragraphs on the page, beginning with "When building a DECnet access list..."

On page 12-34 in the section "DECnet Global Configuration Command Summary," change the description of the default value for **decnet max-address** *value* from 255 to 1023.

Corrections to Chapter 13, "Configuring IP"

In Chapter 13, incorrect text appears on page 13-2 under Step 2. The correct text is as follows:

Step 3: Consider addressing options and broadcast packet handling, using commands described in the "Setting IP Interface Addresses" and "Broadcasting in the Internet" sections.

On page 13-12, in the section "Internet Broadcast Addresses," the explanation of the **no ip broadcast-address** command is incomplete. Change the last paragraph on the page to read as follows:

Use the **no ip broadcast-address** command to remove the broadcast address. To change from another broadcast address to the default broadcast address of 255.255.255.255, you must enter the following command:

```
ip broadcast-address 255.255.255.255
```

You cannot use a command in the format **no ip broadcast-address x.x.x.x** (where x.x.x.x is not 255.255.255.255) to return to the default broadcast address.

On page 13-18, in the section "Setting and Adjusting Packet Sizes," add the following paragraph:

The CTR card does not support the switching of frames larger than 4472 bytes. Interoperability problems may occur if CTR cards are intermixed with other Token Ring cards on the same network. These problems can be minimized by lowering the IP and CLNS maximum packet sizes (MTUs) to be the same on all devices on the network, using the **ip mtu** and **clns mtu** interface commands.

On page 13-28, the discussion "Controlling Interface Access" incorrectly states the following:

- After receiving and routing a packet to a controlled interface, the router checks the destination address of the packet against the access list.

Instead, this description should state that the router evaluates the *source* address. Replacement text follows:

- After receiving and routing a packet to a controlled interface, the router checks the source address of the packet against the access list.

On page 13-38, in the section "IP Processing on a Serial Interface," change the first bulleted restriction to read as follows:

- Only serial interfaces using HDLC or Frame Relay encapsulation can be unnumbered. It is not possible to use this subcommand with X.25 interfaces.

Add the following restriction to the same bulleted list on page 13-38:

- The interface defined by the *interface-name* argument must be enabled or not administratively down (as indicated in the **show interfaces** command display).

On page 13-43, the networks illustrated in Figure 13-9 were incorrectly referred to as "networks 132 and 196." The correct reference is as follows:

In the following example, networks 131 and 192 are separated by a backbone, as shown in Figure 13-9. The two networks are brought into the same logical network through the use of secondary addresses.

The example at the bottom of page 13-44, in the section "Helper Addresses," is incorrect. The correct example follows:

```
!
ip forward-protocol udp
!
interface ethernet 1
ip helper-address 110.44.23.7
interface ethernet 2
ip helper-address 191.24.1.19
```

Table 13-12 on page 13-73 incorrectly defines the trace character "U" as meaning the host is unreachable. Instead, the trace character "U" means that the port is unreachable. Also add the trace character "H" to the table, which means that host is unreachable.

Corrections to Chapter 14, "The IP Routing Protocols"

The following note regarding OSPF support is incorrect. Secondary address assignment is supported in OSPF.

Note: OSPF does not support secondary address assignment.

The note should read:

Note: OSPF supports secondary address assignment.

Add the following text and example to page 14-16, in the section "Defining OSPF on Networks and Assigning Area IDs":

When you configure secondary addresses with OSPF, make certain that the primary and secondary addresses of an interface fall into the same area. If they are in different areas, secondary addresses will be ignored. The following example illustrates the correct way to configure secondary addresses.

```
interface ethernet 0
ip address 131.108.10.10 255.255.255.0
ip address 10.0.0.10 255.0.0.0 secondary

router ospf 109
network 131.108.0.0 0.0.255.255 area 0
network 10.0.0.0. 0.255.255.255 area 0
```

On page 14-19 under "Defining a Stub Area" the initial paragraph should be changed to read as follows:

"There are two stub area router subcommands: the **stub** and **default-cost** options of the **area** router subcommand. In all routers attached to the stub area, the area should be configured as a stub area using the **stub** option of the **area** command. Use the **default-cost** only on an area border router attached to the stub area. This command provides the metric for the summary default route generated by the area border router into the stub area. The syntax for the **area** *area-id* **stub** router subcommand is as follows:"

On page 14-20 in the same section, the description of the **default-cost** keyword should be changed to read as follows:

"The **default-cost** *cost* keyword/argument pair assigns a specific cost for the default summary route used for the stub area. The accepted value is a 24-bit number."

On page 14-20, make the following edits:

Change the sentence "Different physical interfaces have different defaults, as follows:" to "Using the following formula, the default path costs were calculated as noted in the list that follows the formula. If these values do not suit your network, you can use your own method of calculating path costs."

On page 14-21, add the information that follows to the description of link state advertisements in the second paragraph under "Setting the Link State Retransmission Interval":

"The retransmit interval is the number of seconds a router should wait before retransmitting a link state advertisement (LSA) in the absence of an acknowledgment. If a change in the link occurs, an LSA is immediately transmitted."

In the third paragraph on page 14-22 under “Setting the Transmission Time for Link State Updates” that begins “The argument *number-of-seconds*,” replace the sentence that describes the default value.

The default value is 10 seconds.

Also add the paragraph that follows.

“If the delay is not added before transmission over a link, the time in which the LSA propagates over the link is not considered. This setting has more significance on very low-speed links.”

On page 14-22, under the section “Setting Router Priority,” the default router priority is incorrectly stated as being zero. The correct default value is 1.

On page 14-23, for the **ip ospf authentication-key** command, the default is null.

On page 14-25, for the **area** *area-id* **virtual-link** router-id [**authentication-key**] router subcommand, the default is null.

On page 14-27, the first paragraph incorrectly states that the update period is 90 seconds. The correct update period is 180 seconds. All other values described in the first paragraph are correct.

On page 14-30, in the section “Specifying the List of BGP Networks,” add the sentence that follows:

“The maximum number of network subcommands allowed under a router BGP process is 200.”

On page 14-32, in the section “Setting Route Weights,” add the sentence that follows.

“Negative values for the *weight* argument are not allowed.”

On page 14-36, add the following sections after the section “Clearing BGP Connections”:

Controlling BGP to Determine Which Neighbors Are Peers

To control how a BGP process determines which neighbors will be treated as peers, use the **neighbor any** router subcommand with the **router bgp 0** global subcommand. The syntax for this command is as follows:

neighbor any [*list*]
no neighbor any [*list*]

If the *list* argument is specified, the neighbor *must* be accepted by the access list number specified to be allowed to peer with the BGP process.

Continuing with the preceding sample configuration, the configuration would look like the following:

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.234.2  remote-as 109
neighbor 150.136.64.19  remote-as  99
neighbor any 1
access-list 1 permit 192.31.7.0 0.0.0.255
```

Using BGP without IGP Redistribution

Use the **no synchronization** subcommand of the **router bgp** global command when you want to disable the synchronization between BGP and your IGP.

Usually, a BGP speaker does not advertise a route to an external neighbor unless that route is local or exists in the IGP. The **no synchronization** command will allow a router to advertise a network route without waiting for the IGP. This feature allows routers within an AS to have the route before BGP makes it available to other ASs. Use the **synchronization** command if there are routers in the AS that do not speak BGP. The default is synchronization.

no synchronization
synchronization

In Figure 1, with synchronization on, Router B will not advertise network 10.0.0.0 to Router A until an IGRP route for network 10.0.0.0 exists. If you specify the **no synchronization** command, Router B will advertise network 10.0.0.0 as soon as possible.

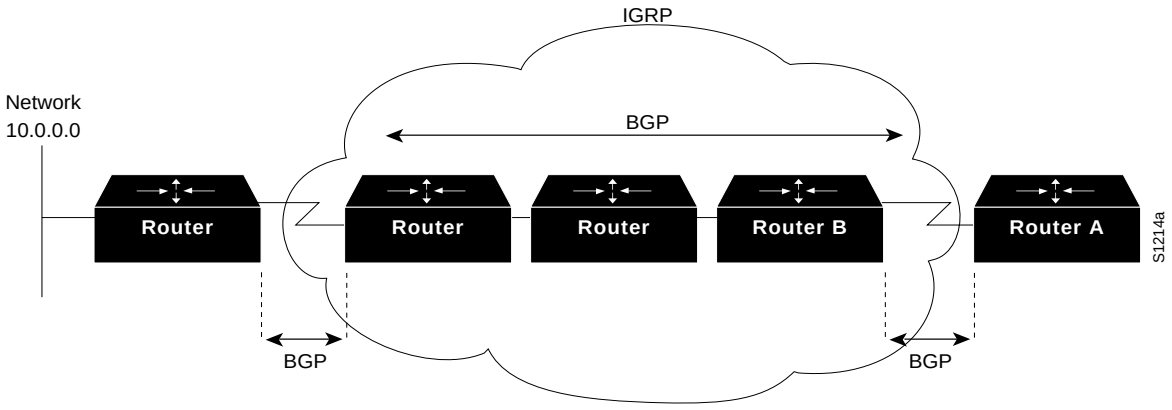

Figure 1 Synchronization Diagram

Displaying the BGP Routing Table

Use the **show ip bgp** EXEC command to display a particular network in the BGP routing table. Enter this command at the EXEC prompt:

show ip bgp [*network*]

The optional argument *network* is a network number and is entered to display a particular network in the BGP routing table.

Following is sample output of the command without specifying a network number:

```
puck> show ip bgp
BGP table version is 22855, local router ID is 192.54.222.5
Status codes: * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network       Next Hop        Metric Weight Path
*> 128.128.0.0   131.192.115.3                0 ?
*> 192.132.70.0  192.54.222.9               20 702 701 ?
*> 152.155.0.0   131.192.77.1                0 ?
*> 192.74.137.0  192.54.222.9               20 702 701 i
*> 192.52.247.0  131.192.2.1                 0 ?
*> 146.150.0.0   131.192.77.1                0 ?
*> 192.139.79.0  192.54.222.9               20 702 701 ?
*> 192.75.142.0  192.54.222.9               20 702 701 ?
*> 192.75.141.0  192.54.222.9               20 702 701 ?
*> 142.136.0.0   192.54.222.9               20 702 701 ?
*> 192.139.77.0  192.54.222.9               20 702 701 ?
*> 129.135.0.0   192.54.222.9               20 702 701 ?
*> 192.160.103.0 192.54.222.9               20 702 701 ?
*> 7.0.0.0       192.54.222.9               20 702 701 35 e
*> 139.140.0.0   131.192.34.2                0 ?
*> 8.0.0.0       131.192.2.1                 0 ?
*> 192.58.242.0  131.192.2.1                 0 ?
```

In the display:

- The Table Version is the internal version number for the table. This is incremented any time the table changes.
- The first three characters (Status codes) indicate the status. The asterisk (*) indicates that the table entry is valid. The > character indicates that the table entry is the best entry to use for that network. The lowercase i indicates that the table entry was learned via an internal BGP session.
- The Next Hop entry is the IP address of the next system to use when forwarding a packet to the destination network. An entry of 0.0.0.0 indicates that the local router has some non-BGP route to this network.
- The Metric field, if any, is the value of the interautonomous system metric. This is frequently not used.
- The Weight field is set through the use of AS filters.
- The Path field is the autonomous system path to the destination network. At the end of the path is the origin code for the path. The lowercase i indicates that the entry was originated with the local IGP and advertised with a **network** subcommand. A lowercase e indicates that the route originated with EGP. A question mark (?) indicates that the origin of the path is not clear. Usually this a path that is redistributed into BGP from an IGP.

Following is sample output of the command when used with a network number:

```
puck> show ip bgp 7.0.0.0
BGP routing table entry for 7.0.0.0, version 20760
Paths: (1 available, best #0)
  702 701 35
    192.54.222.9 from 192.54.222.9, metric 0, weight 20
      Origin EGP, valid, external, best
```

In the display:

- The first line indicates the network that the route is for and the version number of the table the last time the route changed.
- Paths indicates the number of paths, and the index to the best path.
- The third line is the AS path associated with the route.
- 192.54.222.9 from 192.54.222.9 indicates the next hop for the route and the router that it is learned from.
- The metric is the metric assigned to the route.
- The weight is the administrative weight assigned to the route.
- Origin EGP is the origin code for the route.
- Valid indicates whether or not the route is valid (usable).
- External indicates whether or not the route was learned externally or internally.
- Best indicates if the route is the best route or not.

Displaying Routes Learned from a Neighbor

Use the **show ip bgp neighbors** *network* routes command to show the routes learned from that particular neighbor. Enter this command at the EXEC prompt:

show ip bgp neighbors *network* **routes**

The optional argument *network* is the network number for the neighbor whose routes you have learned from.

The display is the same as the display for the **show ip bgp** command.

Displaying BGP Paths

Use the **show ip bgp paths** command to display all the BGP paths in the database. Enter this command at the EXEC prompt:

show ip bgp paths

Following is sample output:

```
Address    Hash Refcount Metric Path
0x297A9C      0        2      0 i
0x30BF84      1        0      0 702 701 ?
0x2F7BC8      2      235      0 ?
0x2FA1D8      3        0      0 702 701 i
```

In the display:

Address—Internal address where the path is stored

Hash—Hash bucket where path is stored

Refcount—Number of routes using that path

Metric—INTER_AS metric for the path

Path—AS_PATH for that route, followed by the origin code for that route

Displaying BGP Summaries

Use the **show ip bgp summary** command to display the status of all BGP connections. Enter the command that follows at the EXEC prompt.

show ip bgp summary

Following is sample output:

```
BGP table version is 3937, main routing table version 3937

Neighbor        V    AS MsgRcvd MsgSent TblVer InQ OutQ Uptime/State
192.54.222.6    2   690    7655     268   3937   0    0 2:39:51
192.54.222.9    3   702     682     364   3937   0    0 2:39:54
```

In the display:

- BGP table version—Internal version number of BGP database
- routing table version—Last version of BGP database that was injected into main routing table
- Neighbor—IP address of a neighbor
- V—BGP version number spoken to that neighbor
- AS—Autonomous system number of that neighbor
- MsgRcvd—BGP messages received from that neighbor
- MsgSent—BGP messages sent to that neighbor
- TblVer—Last version of the BGP database that was sent to that neighbor
- InQ—Number of messages from that neighbor waiting to be processed
- OutQ—Number of messages waiting to be sent to that neighbor
- Update/State—Length of time that the BGP session has been in state Established, or the current state if it is not Established

Debugging IP-BGP Updates

Use the following EXEC commands to debug BGP. For each **debug** command, there is a corresponding **undebug** command that runs the messages off.

debug ip-bgp-updates

The **debug ip-bgp-updates** command generates per-update messages.

Add the following bulleted list item to page 14-38, in the section "BGP Route Selection Rules":

- If IGP synchronization is enabled, then a path learned via BGP must be synchronized with IGP before it can be considered for selection.

In the same section, add the following as the next to the last paragraph in that section:

"You also can create routing loops if a BGP speaker injects a route into the IGP and then forwards packets back into the AS. For this reason, routes learned via internal BGP cannot be redistributed into other routing protocols."

On page 14-44, in the section "Example Core Gateway EGP Configuration," change the second paragraph to read as follows:

With the following configuration (on the router designated Core), C1, C2, and C3 cannot route traffic directly to each other via the X.25 network:

On the same page, replace the fourth and fifth paragraphs (those that begin, “This command specifies...” and “In contrast with this general form...”) with the following paragraphs:

This configuration specifies that an EGP process on any router on network 10.0.0.0 can act as a peer with the Core router. All traffic in this configuration will flow through the Core router.

Third-party advertisements allow traffic to bypass the Core router and go directly to the router that advertised reachability to Core.

The command syntax displayed in the section “Adjusting Metrics” on page 14-49 is incorrect. The correct syntax follows:

offset-list {**in**|**out**} *offset* [*accesslist*]
no offset-list {**in**|**out**} *offset* [*accesslist*]

The examples in this section should also be changed, as follows:

```
offset-list out 10 0

offset-list out 10 121
```

On page 14-64, in the section “Configuring Static Routes,” add the following paragraph:

Static routes that point to an interface (using the argument *interface*) are advertised via RIP and IGRP regardless of whether **redistribute static** commands were specified for those routing protocols. This is because static routes that point to an interface are considered in the routing table to be connected and hence lose their static nature. However, if you define a static route to an interface that is not one of the networks defined in a **network** command, neither RIP nor IGRP will advertise the route.

On page 14-69, add the *sleeptime* option to the **timers basic** router subcommand:

timers basic *update invalid holddown flush sleeptime*
no timers basic

On page 14-70, add this paragraph at the end of the bulleted list:

- The optional argument *sleeptime* is used to postpone IGRP routing updates for the specified number of milliseconds. Note that other timing values are specified in seconds. The *sleeptime* value should be less than the *update* time. If the *sleeptime* is greater than the *update* time, routing tables will become unsynchronized.

On page 14-67, in the section "IGRP Metric Adjustments," the paragraphs that discuss the formula for computing the composite IGRP metric should read as follows:

If K5 equals 0, the composite IGRP metric is computed according to the following formula:

metric = [K1*bandwidth + (K2*bandwidth)/(256 - load) + K3*delay]

If K5 does not equal 0, an additional operation is done:

metric = metric * [K5/(reliability + K4)]

The default version of IGRP has K1 == K3 == 1, K2 == K4 == K5 == 0.

Delay is in units of 10 microseconds. This gives a range of 10 microseconds to 168 seconds. A delay of all ones indicates that the network is unreachable.

Bandwidth is inverse bandwidth of the path in bits per second scaled by a factor of 1e10. The range is from a 1200-bps line to 10 Gbps.

On page 14-78, in the section "Example 1: Basic OSPF Configuration," change the first paragraph to read as follows:

The following example illustrates a simple OSPF configuration that enables OSPF routing process 9000, attaches Ethernet 0 to area 0.0.0.0, and redistributes RIP into OSPF:

Corrections to Chapters 15 and 16, "Switching ISO CLNS" and "ISO CLNS Routing Protocols"

In Chapters 15 and 16, "Switching ISO CLNS" and "ISO CLNS Routing Protocols," various CLNS configuration commands allow the use of a name to represent a full NSAP or NET. When names are used in this way, there are certain system behaviors that are not documented in the *Router Products Configuration and Reference* manual. The notes that follow outline the effects and requirements associated with using names to represent NETs and NSAPs.

- Although using names as proxies for addresses is allowed with CLNS commands, the name is never written out to NVRAM.
- The **clns host** command is generated after the **router** subcommands when the configuration file is parsed. As a result, the NVRAM version of the configuration cannot be edited to specifically change the address defined in the original **clns host** command. This affects all commands that accept names.
  - **net** (router subcommand)
  - **clns is-neighbor** (interface subcommand)

— **clns es-neighbor** (interface subcommand)

— **clns route** (global configuration command)

- In the following command example, the **net** command specification assumes that a **clns host** command already has been entered (with the name cisco representing a valid NET).

```
router iso-igrp
net cisco
```

Correction to Chapter 15, "Switching ISO CLNS"

In Chapter 15 the definitions of the ISO-IGRP and IS-IS NSAP addressing structures are in error. The corrected definitions follow.

Corrected ISO-IGRP version, page 15-5, (third bulleted item):

- The DSP is at least a 10-byte structure that contains a one-byte domain address, two-byte Area identifier, six-byte station ID, and one-byte n-selector (S) field.

Corrected IS-IS version, page 15-6 (third bulleted item)):

- The DSP is a eight-byte structure that contains a one-byte area ID, a six-byte station ID, and a one-byte n-selector (S) field.

On page 15-16, in the section "NSAP Shortcut Command," add the following sentence to the end of the second paragraph (the paragraph that starts, "The argument *name*..."):

The first character of the *name* argument must be a letter; it cannot be a digit.

On page 15-17, add the following section at the bottom of the page:

DEC-Compatible Mode

The **clns dec-compatible** command is useful in old DEC implementations of ES-IS in which the NSAP address advertised in an ISH does not have the N-selector byte present. (This is the last byte of the NSAP address.) This command has the following syntax:

clns dec-compatible
no clns dec-compatible

When the **clns dec-compatible** command is set, ISHes sent and received ignore the N-selector byte.

Corrections to Chapter 16, "ISO CLNS Routing Protocols"

Under the command syntax description for the following form of the **clns route** command on page 16-4, replace the description that follows.

clns route *nsap-prefix interface-type unit* [*SNPA-address*]

The optional argument *SNPA-address* is required for multiaccess networks.

The new description should read:

clns route *nsap-prefix interface-type unit* [*SNPA-address*]

Note: You must include the argument *SNPA-address* when the *interface-type unit* specifies a multiaccess network. Otherwise you receive an error message that complains about a bad SNPA.

The command syntax displayed in the section "Enabling IS-IS Routing" on page 16-9 is incorrect. The correct syntax follows:

router isis [*tag*]
no router isis [*tag*]

The example configuration provided on page 16-14 for CLNS Static Intradomain Routing included an erroneous **net** command specification. The example that follows provides a corrected command listing. In the revised version, the **net** command uses the host name chicago as its argument. Replacement text:

The following is one way to configure the router in Chicago:

```
clns host chicago 47.0004.0050.0002.0000.0c00.243b.00
clns host detroit 47.0004.0050.0001.0000.0c00.1e12.00
clns routing
router iso-igrp sales
net chicago
interface ethernet 0
clns router iso-igrp sales
interface serial 0
encapsulation x25
x25 address 31342174523156
x25 nvc 4
clns router iso-igrp sales
clns is-neighbor detroit 31343136931281 broadcast
```

The two example configurations provided on page 16-16 contain erroneous command specifications. Corrections are described and corrected example specifications are provided in the sections that follow.

The address specified in the first example's last line is changed from 47.0006.0200.0200 to 39.001 to reduce any ambiguity about the use of static routes. In this case, static routes are being used to route to different domains. The correct specification for this example follows:

Example Configuration for Castor

```
router iso-igrp orion
net 47.0006.0200.0100.0102.0304.0506.00
!
clns host pollux 47.0006.0200.0200.1112.1314.1516.00
!
interface ethernet 0
clns router iso-igrp orion
!
interface serial 1
clns enable
!
clns route 39.0001 pollux
```

The second example incorrectly specifies the domain name orion in the **clns host** and **clns route** global configuration commands. Instead, the host name castor should be specified in both cases. The correct specification for this example follows:

Example Configuration for Pollux

```
router iso-igrp pleiades
net 47.0006.0200.0200.1112.1314.1516.00
!
clns host castor 47.0006.0200.0100.0001.0102.0304.0506.00
!
interface ethernet 0
clns router iso-igrp pleiades
!
interface serial 0
clns enable
!
clns route 47.0006.0200.0100 castor
```

Corrections to Chapter 17, "Routing Novell IPX"

On page 17-2 in the section "Configuring Novell Routing," add the note that follows.

Note: If you plan to use DECnet and Novell routing concurrently on the same interface, you must enable DECnet routing first and then enable Novell routing without specifying the optional MAC address. If Novell routing is enabled before DECnet routing, Novell routing will be disrupted.

On page 17-4 in the section "Novell Encapsulation," the description of the default keyword **novell-ether** in the first sentence of that paragraph is wrong. The correct sentence follows.

The default keyword argument is **novell-ether**, which specifies Novell IPX over Ethernet using Novell's variant of IEEE 802.3 encapsulation.

On page 17-5, in the section "Setting Maximum Paths," add the following sentence to the second paragraph:

The cost of a path is determined by the hop count.

On page 17-25, add the following description for the command display:

The numbers in brackets, for example [1/1], represent the metric and the delay. The first part is the metric used to make routing decisions. The second part is delay, expressed in IBM clock ticks.

On page 17-25, the **show novell route** command description in the manual should document the fact that the command can be used to display just one route. For example, **show novell route A100FF** would show just the route table entry for network A100FF.

On page 17-27, add the following **debug** commands:

debug novell-routing-events

The **debug novell-routing-events** command displays a reduced subset of the output of the **debug novell-routing** command.

debug novell-sap-events

The **debug novell-sap-events** command displays a reduced subset of the output of the **debug novell-sap** command.

Correction to Chapter 19, "Routing VINES"

On page 19-3, the portion of the VINES network address space assigned to Cisco Systems by Banyan is stated incorrectly. The hexadecimal number begins with 0x300 or 0x301, not 0x600 or 0x601.

On page 19-3, under "Configuring VINES Routing," add the following note:

Note: The Token Ring interface must be fully initialized prior to enabling VINES routing if you wish to use the Token Ring interface.

On page 19-4, the command syntax is listed incorrectly. Following is the correct syntax:

vines redirect [*number*]

Correction to Chapter 20, "Routing XNS"

On page 20-17, Example 4 at the top and Example 2 toward the bottom of the page have incorrect addresses. The correct examples follow.

Example 4

In the following example, access is denied from source address 0011.1622.0015 on network 21 to destination address 01D3.020C.0022 on network 31. All other packets are permitted.

```
access-list 500 deny 1 21.0011.1622.0015 0000.0000.0000 31.01D3.020C.0022
0000.0000.0000
```

Example 2

An example of applying a mask for both the source and destination networks is as follows:

```
access-list 500 permit 1 21.0011.1622.0015 0000.0000.0000 1234
31.01D3.020C.0022 0000.00ff.ffff 1234
```

Corrections to Chapter 21, "Configuring Transparent Bridging"

The example on page 21-14, in the section "Assigning Path Costs" is incorrect. The correct example follows:

```
!
interface ethernet 0
bridge-group 1 path-cost 250
!
```

The command syntax displayed in the section "Filtering on MAC Address" on page 21-15 is incorrect. The correct syntax follows:

bridge *group* **address** *MAC-address* {**forward**|**discard**} [*interface*]
no bridge *group* **address** *MAC-address*

Add the following no versions to the command syntax displays on pages 21-18 through 21-20. The no versions remove the filter.

bridge-group *group* **input-type-list** *list*
no bridge-group *group* **input-type-list** *list*

bridge-group *group* **output-type-list** *list*
no bridge-group *group* **output-type-list** *list*

bridge-group *group* **input-lsap-list** *list*
no bridge-group *group* **input-lsap-list** *list*

bridge-group *group* **output-lsap-list** *list*
no bridge-group *group* **output-lsap-list** *list*

Add the following no versions to the command syntax displays on pages 21-26 through 21-27. The no versions remove the access conditions.

bridge-group *group* **input-lat-service-deny** *grouplist*
no bridge-group *group* **input-lat-service-deny** *grouplist*

bridge-group *group* **input-lat-service-permit** *grouplist*
no bridge-group *group* **input-lat-service-permit** *grouplist*

On page 21-20, in the top example, the corrected example is as follows:

```
!
interface ethernet 0
bridge-group 1
bridge-group 1 output-lat-service-deny 12-20
!
```

Add the following note to page 21-27, in the section "Establishing Load Balancing":

Note: Load balancing only works on directly connected links such as HDLC. It is not supported for packet-switched networks like X.25 and Frame Relay.

On page 21-36, in the section "Maintaining the Transparent Bridge," change the first sentence in the first paragraph to the following:

Use the **clear bridge** command to remove any learned entries from the forwarding database and to clear the transmit and receive counts for any statically configured or system-configured entries.

Correction to Chapter 22, "Configuring Source-Route Bridging"

For better command integration with future software releases, the Source-Route/Translational Bridging (SR/TLB) **source-bridge old-oui** command has been changed to **bridge old-oui**. The functionality of the new command is identical to that of the **source-bridge old-oui** command; only the name has been changed. Therefore, wherever the documentation mentions **source-bridge old-oui**, use **bridge old-oui** instead.

bridge old-oui
no bridge old-oui

Refer to pages 22-34 and 22-35 for a description of how to use the command. For compatibility with the manual, the format as described there, **source-bridge old-oui**, is still accepted when entered by hand in the configuration, but the **bridge old-oui** form will be printed out and used with both the **write** and **show** commands.

On page 22-42, under "Administratve Filtering by Vendor Code or Address," make the following changes:

Change the heading "Filtering Source Addresses" to "Filtering Incomming Frames."

Change the heading "Filtering Destination Addresses" to "Filtering Outgoing Frames."

Change the first sentence under this heading to read, "To configure filtering on IEEE 802 source addresses, assign an access list to a particluar interface for filtering the Token Ring or IEEE 802 source address."

Add the following note:

Note: The **source-bridge output-address-list** command uses the source address for filtering, which differs from other MAC level access lists that use the destination address for output filtering.

The examples on page 22-51 under "Designing and Configuring the Access Lists Used by the Expression," on page 22-54 under "Configuring the Access Expression into the Router," and on page 22-55 under "Optimizing Access Expressions" are missing the hexadecimal notation for the access list permited value. Replace all instances of

```
0404
f0f0
```

with the hexadecimal values that follow.

```
0x0404
0xf0f0
```

On page 22-89, the description of the **source-bridge enable-805d** command refers incorrectly to a **source sap-805d** command. The correct reference is to the **source-bridge sap-805d** command.

Corrections to Chapter 23, "Configuring Serial Tunneling in SDLC and HDLC Environments"

Figure 23-5 on page 23-13 has an incorrect IP address. The address 131.408.1.2 should be 131.108.1.2.

On page 23-16, the example found in the section "Prioritizing STUN Traffic" should be as follows:

```
priority-list 4 protocol stun high
priority-list 4 protocol ip high tcp 1994
```

In the example on pages 23-16 and 23-17, reference to "network 1" should be "network 1.0.0.0."

On page 23-17 in the first paragraph, "network 1.0.0.1" should be "host 1.0.0.1" and "network 1.0.0.2" should be "host 1.0.0.2."

On page 23-18, Figure 23-7, interface E0 on ciscoA needs an IP address of 1.0.0.1 and interface E1 on ciscoB needs an IP address of 2.0.0.2 to support the configuration that follows.

On page 23-22, change the default value for the **stun poll-interval** command to 100 milliseconds, and change the minimum value to 50 milliseconds.

Corrections to Chapter 25, "SDLLC: SDLC to LLC2 Media Translation"

On page 25-1, add the following information to the section "The Cisco SDLLC Function":

SDLLC supports two modes of connections:

- Host-initiated connection—In this type of connection, the Token Ring host is responsible for initiating the connection with the remote serial device. Someone always needs to be present at the host site to initiate connection whenever the remote serial device wants to communicate with the host. "Configuring SDLLC" describes a host-initiated connection.

- Device-initiated connection—In this type of connection, the Token Ring host is always kept in a state ready to accept a connection from the remote serial device. The remote serial device is responsible for initiating connections. The advantage of this scheme is that the serial device can communicate with the Token Ring host whenever it chooses without requiring personnel to be on the host site.

Add the following section to page 25-3:

SDLLC Device-Initated Connections

To support device-initiated connections for SDLLC, you must specify the **sdllc partner** command. This is an interface subcommand that must be specified for the serial line that is attached to the serial line device.

sdllc partner *mac-address sdlc-address*
no sdllc partner

Both the MAC address of the Token Ring host and the SDLC serial line address are required to initiate connections with the Token Ring host. The argument *mac-address* is the 48-bit MAC address of the Token Ring host. It is written as a dotted triple of four-digit hexadecimal numbers. The *sdlc-address* argument is the SDLC address of the serial device that will communicate with the Token Ring host. These two devices talk to each other through the Cisco router.

Although the device is said to initiate connections, the router actually initiates connections with the Token Ring host on behalf of the serial device. As part of the SDLLC implementation, the serial device "thinks" that it is communicating with a host also on a serial line. It is actually the router that does all the frame and protocol conversions between serial and Token Ring devices.

There are two conditions under which a router will attempt to initiate a connection to a host on behalf of a serial device:

- When the serial device attached to the router is powered on. In this case, the router attached to the serial line detects a change in interface signals and initiates a connection with the Token Ring hosts by exchanging explorer and XID packets.
- When a previously shut down serial interface is brought back online. When the **no shutdown** command is issued, the router will detect a change in the serial line state from down to up and initiate a session with the Token Ring host by exchanging explorer and XID packets.

The router will continue trying once a minute to initiate a connection whenever one of these two conditions is met, until the host responds to its requests. When you no longer want the router to initiate connections with a host, use the **no sdllc partner** command.

Note: For device-initiated sessions, the host will check the IDBLKNUM of the serial device it receives in the XID packet against the information configured on the host. If the information in the XID packet does not match with what is configured on the host, the host will drop the session. Therefore, for device-initiated connections, always specify the correct IDBLKNUM on the router serial interfaces with the **sdllc xid** command (documented on page 25-9.)

This document is to be used in conjunction with the *Router Products Configuration and Reference* publication.