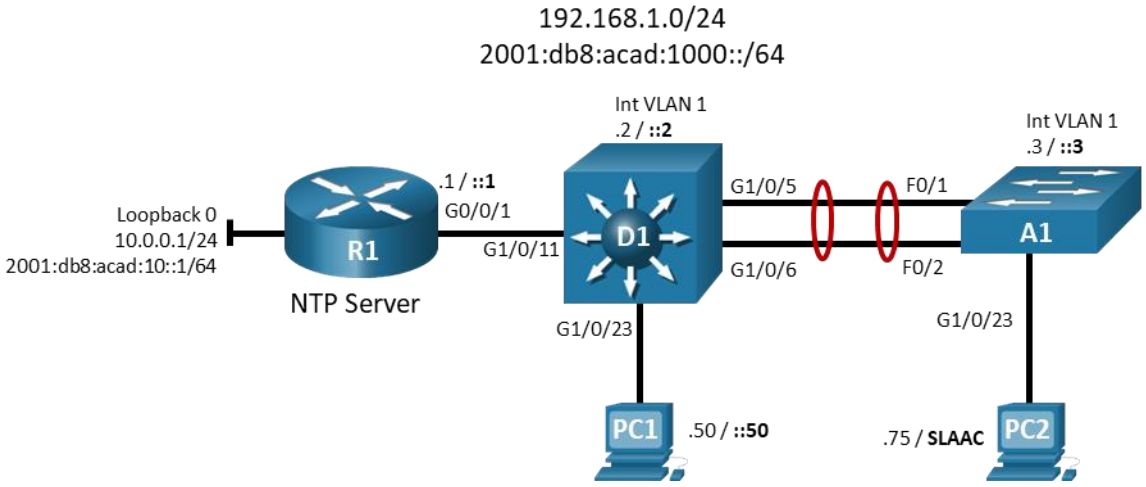
Lab - Implement Flexible Netflow

Topology



Addressing Table

Device	Interface	IP Address	IPv6 Address	IPv6 Link Local
R1	G0/0/1	192.168.1.1/24	2001:db8:acad:1000::1/64	fe80::1:1
	Loopback0	10.0.0.1/24	2001:db8:acad:10::1/64	fe80::1:2
D1	VLAN 1	192.168.1.2/24	2001:db8:acad:1000::2/64	fe80::d1:1
A1	VLAN 1	192.168.1.3/24	2001:db8:acad:1000::3/64	fe80::a1:1
PC1	NIC	192.168.1.50/24	2001:db8:acad:1000::50/64	EUI-64
PC2	NIC	192.168.1.75/24	Assigned by SLAAC	EUI-64

Objectives

Part 1: Build the Network and Configure Basic Device Settings and Interface Addressing

Part 2: Configure and Verify Flexible Netflow

Part 3: (Optional) Configure and Verify Netflow

Background / Scenario

NetFlow is a Cisco IOS technology that provides statistics on packets flowing through the router. NetFlow is the standard for acquiring IP operational data from IP networks. NetFlow provides data to enable network and security monitoring, network planning, traffic analysis, and IP accounting.

Flexible NetFlow improves on original NetFlow by adding the capability to customize the traffic analysis parameters for your specific requirements. Flexible NetFlow facilitates the creation of more complex configurations for traffic analysis and data export through the use of reusable configuration components.

Note: This lab is an exercise in configuring options available for Flexible Netflow and does not necessarily reflect network troubleshooting best practices.

Note: The routers used with CCNP hands-on labs are Cisco 4221 with Cisco IOS XE Release 16.9.4 (universalk9 image). The switches used in the labs are Cisco Catalyst 3650s with Cisco IOS XE Release 16.9.4 (universalk9 image) and Cisco Catalyst 2960s with Cisco IOS Release 15.2(2) (lanbasek9 image). Other routers, switches, and Cisco IOS versions can be used. Depending on the model and Cisco IOS version, the commands available and the output produced might vary from what is shown in the labs. Refer to the Router Interface Summary Table at the end of the lab for the correct interface identifiers.

Note: IOS XE does not support classic Netflow. If your lab has ISR G2 series routers, skip Part 2 of this lab and do Part 3, which covers classic Netflow.

Note: Make sure that the switches have been erased and have no startup configurations. If you are unsure, contact your instructor.

Note: The default Switch Database Manager (SDM) template on a Catalyst 2960 does not support IPv6. You must change the default SDM template to the dual-ipv4-and-ipv6 default template using the **sdm prefer dual-ipv4-and-ipv6 default** global configuration command. Changing the template will require a reboot.

Required Resources

- 1 Router (Cisco 4221 with Cisco IOS XE Release 16.9.4 universal image or comparable)
- 1 Switch (Cisco 3650 with Cisco IOS XE Release 16.9.4 universal image or comparable)
- 1 Switch (Cisco 2960 with Cisco IOS Release 15.2(2) lanbasek9 image or comparable)
- 2 PCs (Choice of operating system with terminal emulation program and a packet capture utility installed, such as Wireshark)
- Console cables to configure the Cisco IOS devices via the console ports
- Ethernet cables as shown in the topology

Instructions

Part 1: Build the Network and Configure Basic Device Settings and Interface Addressing

In Part 1, you will set up the network topology and configure basic settings and interface addressing on routers.

Step 1: Cable the network as shown in the topology.

Attach the devices as shown in the topology diagram, and cable as necessary.

Step 2: Configure basic settings for each device.

a. Console into each device, enter global configuration mode, and apply the basic settings. The startup configurations for each device are provided below.

Router R1

```
hostname R1
no ip domain lookup
ipv6 unicast-routing
banner motd # R1, Implement Flexible Netflow #
line con 0
 exec-timeout 0 0
```

```
 logging synchronous
 exit
line vty 0 4
 privilege level 15
 exec-timeout 0 0
 password cisco123
 login
 exit
interface g0/0/1
 ip address 192.168.1.1 255.255.255.0
 ipv6 address fe80::1:1 link-local
 ipv6 address 2001:db8:acad:1000::1/64
 no shutdown
 exit
interface loopback 0
 ip address 10.0.0.1 255.255.255.0
 ipv6 address fe80::1:2 link-local
 ipv6 address 2001:db8:acadd:10::1/64
 no shutdown
 exit
ntp master 3
end
```

Switch D1

```
hostname D1
no ip domain lookup
ipv6 unicast-routing
banner motd # D1, Implement Flexible Netflow #
line con 0
 exec-timeout 0 0
 logging synchronous
 exit
line vty 0 4
 privilege level 15
 exec-timeout 0 0
 password cisco123
 login
 exit
interface vlan 1
 ip address 192.168.1.2 255.255.255.0
 ipv6 address fe80::d1:1 link-local
 ipv6 address 2001:db8:acad:1000::2/64
 no shutdown
 exit
ip default-gateway 192.168.1.1
interface g1/0/23
```

```
 spanning-tree portfast
 switchport mode access
 no shutdown
 exit
interface g1/0/11
 spanning-tree portfast
 switchport mode access
 no shutdown
 exit
interface range g1/0/5-6
 switchport mode trunk
 channel-group 1 mode active
 no shutdown
 exit
interface range g1/0/1-4, g1/0/7-10, g1/0/12-22, g1/0/24, g1/1/1-4
 shutdown
 exit
ntp server 192.168.1.1
end
```

Switch A1

```
hostname A1
no ip domain lookup
ipv6 unicast-routing
banner motd # A1, Implement Flexible Netflow #
line con 0
 exec-timeout 0 0
 logging synchronous
 exit
line vty 0 4
 privilege level 15
 exec-timeout 0 0
 password cisco123
 login
 exit
interface vlan 1
 ip address 192.168.1.3 255.255.255.0
 ipv6 address fe80::a1:1 link-local
 ipv6 address 2001:db8:acad:1000::3/64
 no shutdown
 exit
ip default-gateway 192.168.1.1
interface range f0/1-2
 switchport mode trunk
 channel-group 1 mode active
 no shutdown
```

```
 exit
interface f0/23
 switchport mode access
 spanning-tree portfast
 no shutdown
 exit
interface range f0/3-22, f0/24, g0/1-2
 shutdown
 exit
ntp server 192.168.1.1
end
```

b. Set the clock on each device to UTC time.

c. Save the running configuration to startup-config.

d. Configure IPv4 and IPv6 addresses on hosts PC1 and PC2 as shown in the addressing table.

e. Verify that R1, D1, A1, and PC2 can successfully ping PC1 at 192.168.1.50.

Part 2: Configure and Verify Flexible Netflow

As previously stated, Flexible Netflow provides the ability to customize traffic analysis parameters. The workflow for Flexible Netflow consists of four steps:

Step 1. Create Flow Records. Flow records define the information to be collected. There are predefined flow records that match the flow caching done by Classic Netflow, or you can configure your own custom flow record to suit your needs.

Step 2. Create Flow Exporter. This defines where compiled statistic information is sent.

Step 3. Create Flow Monitor and associate Flow Records and Flow Exporters with it.

Step 4. Configure the appropriate interface for input or output caching associated with the appropriate Flow Monitor.

In this part of the lab, you will configure Flexible Netflow to send statistical information about R1 interface g0/0/1 to PC1.

Step 1: Create flow records.

a. For our first flow record, we will use the predefined ipv4 original-input flow record. Because it is predefined, there is no configuration necessary.

b. For our second flow record, we will create a custom flow record. Because the first flow record is focused on input traffic, the second will focus on output traffic. Create a flow record named CCNP8-CUSTOM-OUT.

```
R1(config)# flow record CCNP8-CUSTOM-OUT
```

1) Give the flow record a description.

```
R1(config-flow-record)# description Custom Flow Record for outbound traffic
```

2) Set up the flow record to match ipv4 destination address and transport destination.

```
R1(config-flow-record)# match ipv4 destination address
R1(config-flow-record)# match transport destination-port
```

3) Set up the flow record to collect bytes and packets.

```
R1(config-flow-record)# collect counter bytes
```

```
R1(config-flow-record)# collect counter packets
```

4) Use the **show flow record CCNP8-CUSTOM-OUT** command to examine the results.

```
R1# show flow record CCNP8-CUSTOM-OUT
flow record CCNP8-CUSTOM-OUT:
  Description:        Custom Flow Record for outbound traffic
  No. of users:       0
  Total field space:  14 bytes
  Fields:
    match ipv4 destination address
    match transport destination-port
    collect counter bytes
    collect counter packets
```

Step 2: Create a flow exporter.

a. The flow exporter configuration defines where the cached information will be sent. Create a flow exporter named CCNP8-COLLECTOR-HOST. Further specify that the exporter should use Netflow version 9, and point to 192.168.1.50 udp port 9999.

```
R1(config)# flow exporter CCNP8-COLLECTOR-HOST
R1(config-flow-exporter)# destination 192.168.1.50
R1(config-flow-exporter)# export-protocol netflow-v9
R1(config-flow-exporter)# transport UDP 9999
R1(config-flow-exporter)# exit
```

b. Use the **show flow exporter CCNP8-COLLECTOR-HOST** command to examine the results.

```
R1# show flow exporter CCNP8-COLLECTOR-HOST
Flow Exporter CCNP8-COLLECTOR-HOST:
  Description:              User defined
  Export protocol:          NetFlow Version 9
  Transport Configuration:
    Destination IP address: 192.168.1.50
    Source IP address:      192.168.1.1
    Transport Protocol:     UDP
    Destination Port:       9999
    Source Port:            63275
    DSCP:                   0x0
    TTL:                    255
    Output Features:        Used
```

Step 3: Create flow monitors.

The flow monitor associates a flow record with the flow exporter. For our exercise, we need to create two flow monitors, one for each flow record.

a. Create the first flow monitor and name it CCNP8-INBOUND-MONITOR using the **flow monitor CCNP8-INBOUND-MONITOR** command. As part of the flow monitor, specify that it will record the netflow ipv4 original-input flow record, export the cache to the exporter every 30 seconds, and identify CCNP8-COLLECTOR-HOST as the exporter.

```
R1(config)# flow monitor CCNP8-INBOUND-MONITOR
R1(config-flow-monitor)# record netflow ipv4 original-input
R1(config-flow-monitor)# cache timeout active 30
```

```
R1(config-flow-monitor)# exporter CCNP8-COLLECTOR-HOST
R1(config-flow-monitor)# exit
```

b. Create the second flow monitor and name it CCNP8-OUTBOUND-MONITOR using the **flow monitor CCNP8-OUTBOUND-MONITOR** command. As part of the flow monitor, specify that it will record the CCNP8-CUSTOM-OUT flow record, export the cache to the exporter every 30 seconds, and identify CCNP8-COLLECTOR-HOST as the exporter.

```
R1(config)# flow monitor CCNP8-OUTBOUND-MONITOR
R1(config-flow-monitor)# record CCNP8-CUSTOM-OUT
R1(config-flow-monitor)# cache timeout active 30
R1(config-flow-monitor)# exporter CCNP8-COLLECTOR-HOST
R1(config-flow-monitor)# exit
```

c. Use the **show flow monitor** command to examine the results.

```
R1# show flow monitor
Flow Monitor CCNP8-INBOUND-MONITOR:
  Description:       User defined
  Flow Record:       netflow ipv4 original-input
  Flow Exporter:     CCNP8-COLLECTOR-HOST
  Cache:
    Type:                normal (Platform cache)
    Status:              not allocated
    Size:                200000 entries
    Inactive Timeout:    15 secs
    Active Timeout:      30 secs
    Trans end aging:   off

Flow Monitor CCNP8-OUTBOUND-MONITOR:
  Description:       User defined
  Flow Record:       CCNP8-CUSTOM-OUT
  Flow Exporter:     CCNP8-COLLECTOR-HOST
  Cache:
    Type:                normal (Platform cache)
    Status:              not allocated
    Size:                200000 entries
    Inactive Timeout:    15 secs
    Active Timeout:      30 secs
    Trans end aging:   off
```

Step 4: Configure the interface for flow caching.

The last step is to configure the appropriate interface(s) so that they will cache information. In our lab, we will focus on the input and output from interface g0/0/0 on R1. Use the **ip flow monitor <name> <direction>** command on g0/0/1 to specify the inbound and outbound flow monitors you have created.

```
R1(config)# interface g0/0/1
R1(config-if)# ip flow monitor CCNP8-INBOUND-MONITOR input
R1(config-if)# ip flow monitor CCNP8-OUTBOUND-MONITOR output
R1(config-if)# exit
```

Step 5: Create some traffic.

To gather statistics, we will need some traffic.

a. From PC2, start a continuous ping to R1 using IPv4 and IPv6. As a part of each set of ping parameters, set the size of the packets to 1475 bytes.

The windows commands are as follows:

```
C:\> ping 10.0.0.1 -t -l 1475
C:\> ping 2001:db8:acad:1000::1 -t -l 1475
```

b. From switch A1, telnet to R1. Login and leave the session running.

c. From switch D1, use the extended ping utility to send pings to R1 Loopback 0 using a sweep range of 36 bytes to 18024 bytes. Set the repeat count to 1,000,000 and the sweep interval to 1.

```
D1# ping
Protocol [ip]:
Target IP address: 10.0.0.1
Repeat count [5]: 100000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Ingress ping [n]:
Source address or interface:
DSCP Value [0]:
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0x0000ABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]: y
Sweep min size [36]:
Sweep max size [18024]:
Sweep interval [1]:
Type escape sequence to abort.
Sending 89945, [36..18024]-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

d. On PC1, start Wireshark and apply the filter **ip.src == 192.168.1.1 && udp.dstport == 9999 && ! icmp**. This filters the display to show packets from 192.168.1.1 destined for UDP port 9999 and not ICMP packets.

Step 6: Wait 60 seconds then examine the results.

a. On PC1, observe the Wireshark display window. There should be traffic matching the filter being displayed.

b. On R1, issue the command **show flow monitor CCNP8-INBOUND-MONITOR statistics**.

```
R1# show flow monitor CCNP8-INBOUND-MONITOR statistics
  Cache type:                               Normal (Platform cache)
  Cache size:                               200000
  Current entries:                               2
```



```
  High Watermark:                                 12

  Flows added:                                   103
  Flows aged:                                    101
    - Active timeout      (    30 secs)          38
    - Inactive timeout    (    15 secs)          63
```

c. On R1, issue the command **show flow monitor CCNP8-INBOUND-MONITOR cache**. **Note**: Output will vary depending upon how long within the 30-second window traffic has been caching.

```
R1# show flow monitor CCNP8-INBOUND-MONITOR cache
  Cache type:                               Normal (Platform cache)
  Cache size:                               200000
  Current entries:                               1
  High Watermark:                               12

  Flows added:                                 112
  Flows aged:                                  111
    - Active timeout      (    30 secs)          43
    - Inactive timeout    (    15 secs)          68

IPV4 SOURCE ADDRESS:        192.168.1.75
IPV4 DESTINATION ADDRESS:   10.0.0.1
TRNS SOURCE PORT:           0
TRNS DESTINATION PORT:      2048
INTERFACE INPUT:            Gi0/0/1
FLOW SAMPLER ID:            0
IP TOS:                     0x00
IP PROTOCOL:                1
ip source as:               0
ip destination as:          0
ipv4 next hop address:      0.0.0.0
ipv4 source mask:           /0
ipv4 destination mask:      /0
tcp flags:                  0x00
interface output:           Null
counter bytes:              12024
counter packets:            8
timestamp first:            20:43:34.189
timestamp last:             20:43:41.263
```

d. Stop all the pings and exit the telnet session.

Part 3: (Optional) Configure and Verify Netflow

IOS-XE, which is the baseline operating system version for the routers in CCNPv8, only supports Flexible Netflow. However, the CCNP ENCOR blueprint says you must also be able to configure and verify classic Netflow. So the configuration and verification steps are presented here for your reference, or if your school only has ISR G1 or ISR G2 series routers.

Step 1: Configure Netflow.

a. Set the Netflow export version to version 9.

```
R1(config)# ip flow-export version 9
```

b. Set the Netflow export destination to 192.168.1.50 port 9999.

```
R1(config)# ip flow-export destination 192.168.1.50 9999
```

c. On R1 interface G0/1, configure Netflow to monitor ingress and egress traffic.

```
R1(config)# interface g0/1
R1(config-if)# ip flow ingress
R1(config-if)# ip flow egress
R1(config-if)# exit
```

Step 2: Create some traffic.

To gather statistics, we will need some traffic.

a. From PC 2, start a continuous ping to R1 using IPv4 and IPv6. As a part of each set of ping parameters, set the size of the packets to 1475 bytes.

The windows commands are as follows:

```
C:\> ping 10.0.0.1 -t -l 1475
C:\> ping 2001:db8:acad:1000::1 -t -l 1475
```

b. From switch A1, telnet to R1. Login and leave the session running.

c. From switch D1, use the extended ping utility to send pings to R1 Loopback 0 with the df-bit set and using a sweep range of 36 bytes to 18024 bytes. Set the repeat count to 1,000,000 and the sweep interval to 1.

```
D1# ping
Protocol [ip]:
Target IP address: 10.0.0.1
Repeat count [5]: 100000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Ingress ping [n]:
Source address or interface:
DSCP Value [0]:
Type of service [0]:
Set DF bit in IP header? [no]: y
Validate reply data? [no]:
Data pattern [0x0000ABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]: y
Sweep min size [36]:
Sweep max size [18024]:
Sweep interval [1]:
Type escape sequence to abort.
Sending 89945, [36..18024]-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

d. On PC1, start Wireshark and apply the filter **ip.src == 192.168.1.1 && udp.dstport == 9999 && ! icmp**. This filters the display to show packets from 192.168.1.1 destined for UDP port 9999 and not ICMP packets.

Step 3: Verify Netflow.

a. Issue the command **show ip flow interface** to verify the interface(s) involved in flow capture.

```
R1# show ip flow interface
GigabitEthernet0/1
  ip flow ingress
  ip flow egress
```

b. Issue the command **show ip flow export** to show the collection host IP address and how many flows have been exported.

```
R1# show ip flow export
Flow export v9 is enabled for main cache
  Export source and destination details :
  VRF ID : Default
    Destination(1)  192.168.1.50 (9999)
  Version 9 flow records
  117 flows exported in 55 udp datagrams
  0 flows failed due to lack of export packet
  0 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues
  0 export packets were dropped due to fragmentation failures
  0 export packets were dropped due to encapsulation fixup failures
```

c. Issue the command **show ip cache flow** to see flow information.

```
R1# show ip cache flow
IP packet size distribution (2597 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .172 .023 .070 .016 .012 .016 .016 .017 .016 .012 .012 .012 .012 .012 .012

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .012 .012 .012 .172 .355 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
  3 active, 4093 inactive, 97 added
  2551 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 34056 bytes
  3 active, 1021 inactive, 93 added, 93 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow     /Flow
TCP-Telnet           3      0.0         7    42      0.0       0.8      15.5
UDP-NTP             36      0.0         1    76      0.0       0.6      15.7
```

```
UDP-other            19      0.0          6   106      0.0       5.1      15.4

SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP  Pkts
ICMP                 36      0.0          41   750      0.0       1.3      15.0
Total:               94      0.0          18   675      0.0       1.8      15.4

SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP  Pkts
Gi0/1         192.168.1.50    Local         192.168.1.1     01 0000 0303     1
Gi0/1         192.168.1.75    Local         10.0.0.1        01 0000 0800   447
Gi0/1         192.168.1.75    Local         10.0.0.1        01 0000 0000   447
```

d. You should be seeing packets collected in Wireshark.

e. Stop all the pings and exit the telnet session

Router Interface Summary Table

Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
1800	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
1900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2801	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
2811	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
4221	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
4300	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)

Note: To find out how the router is configured, look at the interfaces to identify the type of router and how many interfaces the router has. There is no way to effectively list all the combinations of configurations for each router class. This table includes identifiers for the possible combinations of Ethernet and Serial interfaces in the device. The table does not include any other type of interface, even though a specific router may contain one. An example of this might be an ISDN BRI interface. The string in parenthesis is the legal abbreviation that can be used in Cisco IOS commands to represent the interface.