360RSW05-WKB-TA07

Cisco 360 CCIE R&S Exercise Workbook Introduction

The Cisco 360 CCIE® R&S Exercise Workbook contains 20 challenging scenarios at the CCIE level that can be used for rigorous self-paced practice.

Each lab provides an extensive answer key, Mentor Guide support, and verification tables and is designed to maximize learning by providing practical experience. Also, self-paced learning resources such as the Cisco 360 CCIE R&S Reference Library and Cisco 360 CCIE R&S lessons supplement the Exercise Workbook scenarios.

Cisco 360 CCIE R&S Exercise Workbook Lab 7 Troubleshooting Section Answer Key

Table of Contents

Answer Key Structure

Section One

The answer key PDF document is downloadable from the web portal.

Section Two

To obtain a comprehensive view of the configuration for a specific section, access the Mentor Guide engine in the web portal.

Exercise Workbook Lab 7 Troubleshooting Section Answer Key

Note	Regardless of any configuration you perform in this lab, it is very important that you conform to the general guidelines that are provided in the "Restrictions and Goals" section. If you do not conform to the guidelines, you could have a significant deduction of points in your final score.

Grading and Duration

- Troubleshooting lab duration: 2 hours
- Troubleshooting lab maximum score: 24 points

Note	You can assess your progress on the self-paced labs in this workbook by adding up the points that are assigned to sections and tasks. Consider taking the full Assessment Labs to assess your readiness level.

Difficulty Level

- Difficulty: Intermediate

Restrictions and Goals

Note	Read this section carefully.

- To receive credit for a subsection, you must fully complete the subsection per requirements. You will *not* receive partial credit for partially completed subsections.
- IPv4 subnets that are displayed in the Lab IPv4 IGP diagram are /24 subnets of 158.10.0.0/16, except for CustomerA VRF, which is 172.16.0.0/16.
- *Points will be deducted from multiple sections for failing to assign correct IPv4 addresses.*
- Advertise loopback interfaces with their original masks.
- All IP addresses involved in this scenario must be reachable, unless explicitly specified otherwise.
- Unless explicitly specified otherwise, addresses and networks that are advertised in the Border Gateway Protocol (BGP) section need to be reachable by all BGP routers but do not have to be reachable by routers that use only interior gateway protocol (IGP).
- Use conventional routing algorithms only, unless the instructions specify otherwise.

- Do not create new interfaces to fulfill IGP requirements, and do not summarize unless you are explicitly asked to do so.
- Do not modify the hostname, console, or vty configuration unless you are specifically asked to do so.
- Do not modify the initial interface or IP address numbering.

Explanation of Each of the Restrictions and Goals

IPv4 subnets that are displayed in the scenario IPv4 IGP diagram belong to network 158.10.0.0/16.

All IP addresses in this exam belong to the 158.10.0.0/16 address space with the exception of prefixes that are explicitly specified as being part of a different IP space. The exception to this are the prefixes associated with the MPLS Layer 3 VPN section in this lab.

Advertise loopback interfaces with their original masks.

The original mask is the mask configured on the loopback interface. OSPF treats loopback interfaces as host routes by default and advertises them as /32 prefixes. The requirement to advertise loopback interfaces with their original masks precludes using the default OSPF network type for the loopback interface. You need to provide a solution such as changing the OSPF network type or summarizations.

All IP addresses that are involved in this scenario must be reachable.

This is a key goal to observe. It requires that all your IGPs and your routing policy tasks be configured properly. The key elements of your routing policy include route redistribution and the controlling of routing updates using the **distribute-lists**, **route-maps**, and **distance** commands. A key point to remember about this lab is that the term "redistribution" is not explicitly used. However, you must perform redistribution to ensure that all IP addresses are reachable without the use of static routes or 0.0.0.0/0 routes.

Addresses and networks that are advertised in the BGP section need to be reachable by all BGP routers but do not have to be reachable by IGP-only routers.

This statement relaxes the requirement that all IP addresses must be reachable. The BGP prefixes need only be reachable only among the routers specified in the BGP section. They can be used in other unicast tables. However, BGP routers need to have the prefixes in the routing tables and to be able to forward traffic to the addresses that are known via BGP.

Use conventional routing algorithms.

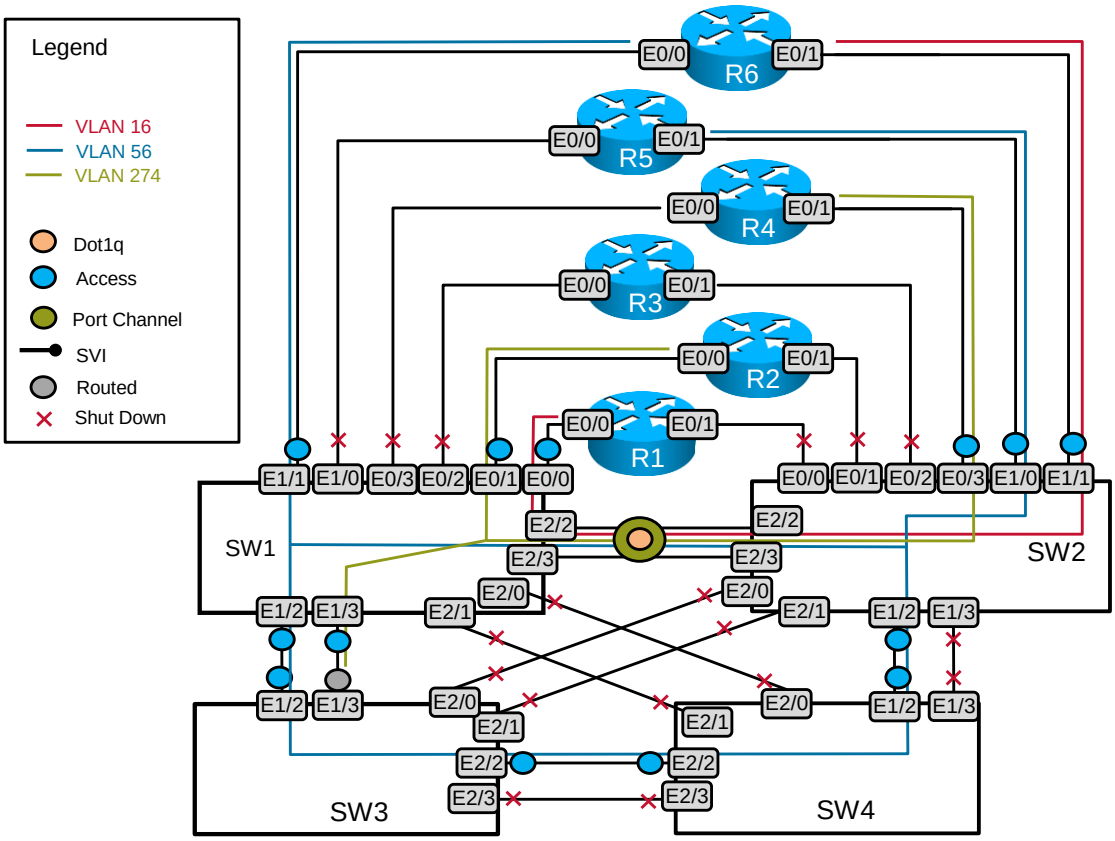
This restriction prevents you from solving any problems by configuring policy routing. At the heart of this restriction is the interpretation of "conventional routing algorithms." Although this phrase can be interpreted in different ways, this interpretation is applied in this workbook:

Conventional routing algorithms are routing algorithms that apply destination-based prefix lookups in a routing table. Conventional routing algorithms do not use any other type of information other than the destination address to make a packet-forwarding decision.

1. Switched Network Troubleshooting Section

In this troubleshooting lab, a Layer 2 diagram is supplied with the scenario itself.

VLAN Propagation Diagram



Consequently, there are no tables that list VLANs or device-to-VLAN assignments in the scenario itself. Begin the troubleshooting process by comparing the initial switch configurations with the supplied Layer 2 diagram.

1.1. Symptom: The link connecting SW1 and SW3 should be access, not trunk.

Analysis and testing:

The supplied diagram shows the link connecting SW1 and SW3 to be an access link in VLAN 56. When you compare the output of the command **show vlan** on SW1 or SW3 to the supplied VLAN diagram, you notice that E1/2 is not associated with VLAN 56. The output of **show interfaces trunk** shows this link to be an ISL trunk.

```
SW1#show vlan brie

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et0/2, Et0/3, Et1/0, Et2/0
                                                Et2/1
16   VLAN0016                         active    Et0/0
```



```
56   VLAN0056                         active    Et1/1
274  VLAN0274                         active    Et0/1, Et1/3
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
SW1#
SW1#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Et1/2       desirable        n-isl          trunking      1
Po1         on               802.1q         trunking      1

Port        Vlans allowed on trunk
Et1/2       1-4094
Po1         16,56,274

Port        Vlans allowed and active in management domain
Et1/2       1,16,56,274
Po1         16,56,274

Port        Vlans in spanning tree forwarding state and not pruned
Et1/2       1,16,56,274
Po1         16,56,274
SW1#
```

Likely cause: SW1 interface E1/2 is configured for dynamic desirable Dynamic Trunking Protocol (DTP) mode.

The configuration of **dynamic desirable** on SW1 and SW3 results in the formation of a trunk on the link.

```
SW1#show interfaces e1/2 switchport
Name: Et1/2
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: isl
Negotiation of Trunking: On
Access Mode VLAN: 56 (VLAN0056)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Appliance trust: none
SW1#
SW3#show interfaces e1/2 switchport
Name: Et1/2
Switchport: Enabled
Administrative Mode: dynamic desirable
Operational Mode: trunk
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: isl
```

```
Negotiation of Trunking: On
Access Mode VLAN: 56 (VLAN0056)
Trunking Native Mode VLAN: 1 (default)
Administrative Native VLAN tagging: enabled
Voice VLAN: none
Administrative private-vlan host-association: none
Administrative private-vlan mapping: none
Administrative private-vlan trunk native VLAN: none
Administrative private-vlan trunk Native VLAN tagging: enabled
Administrative private-vlan trunk encapsulation: dot1q
Administrative private-vlan trunk normal VLANs: none
Administrative private-vlan trunk associations: none
Administrative private-vlan trunk mappings: none
Operational private-vlan: none
Trunking VLANs Enabled: ALL
Pruning VLANs Enabled: 2-1001
Capture Mode Disabled
Capture VLANs Allowed: ALL

Appliance trust: none
SW3#
```

Resolution: Configure the command switchport mode access on the E1/2 interfaces of SW1 and SW3.

Hardcode the mode to access on SW1 and SW3:.

```
SW1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)#int e1/2
SW1(config-if)#switchport mode access
SW1(config-if)#end
SW1#

SW3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)#int e1/2
SW3(config-if)#switchport mode access
SW3(config-if)#end
SW3#
```

Verify the VLAN configuration on SW1 again:

```
SW1#show vlan brie

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et0/2, Et0/3, Et1/0, Et2/0
                                                Et2/1
16   VLAN0016                         active    Et0/0
56   VLAN0056                         active    Et1/1, Et1/2
274  VLAN0274                         active    Et0/1, Et1/3
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
SW1#
```

1.2. Symptom: SW3 interface E1/3 should be routed, not switched.

Analysis and testing:

The supplied diagram indicates that SW3 E1/3 should be a routed port, not a switched port. The error could be discovered by carefully comparing the output of **show interface status** to the supplied diagram:

```
SW3#show interface status

Port      Name               Status       Vlan       Duplex  Speed Type
```



```
Et0/0                          disabled     1            auto   auto unknown
Et0/1                          disabled     1            auto   auto unknown
Et0/2                          disabled     1            auto   auto unknown
Et0/3                          disabled     1            auto   auto unknown
Et1/0                          disabled     1            auto   auto unknown
Et1/1                          disabled     1            auto   auto unknown
Et1/2                          connected    56           auto   auto unknown
Et1/3                          connected    274          auto   auto unknown
Et2/0                          disabled     1            auto   auto unknown
Et2/1                          disabled     1            auto   auto unknown
Et2/2                          connected    56           auto   auto unknown
Et2/3                          disabled     1            auto   auto unknown
SW3#
```

Resolution: Configure SW3 E1/3 as a routed port.

Enter the command **no switchport** on SW3 interface E1/3. Also, remove interface VLAN 274 from the switch and transfer its configuration to E1/3.

```
SW3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)#no int Vlan274
SW3(config)#interface Ethernet1/3
SW3(config-if)#no switchport
SW3(config-if)# ip address 158.10.24.7 255.255.255.0
SW3(config-if)# ip pim sparse-mode
SW3(config-if)#end
SW3#
```

1.3. Symptom: SW3 is configured with unnecessary VLANs.

Analysis and testing:

The lab requirements indicate that only the required VLANs should be configured. The diagram shows only VLAN 56 configured on SW3, yet VLANs 16, 56, and 274 are in the initial configuration:

```
SW3#show vlan brie

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et0/0, Et0/1, Et0/2, Et0/3
                                                Et1/0, Et1/1, Et2/0, Et2/1
                                                Et2/3
16   VLAN0016                         active
56   VLAN0056                         active    Et1/2, Et2/2
274  VLAN0274                         active
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
SW3#
```

Resolution: Remove VLANs 16 and 274 from SW3.

```
SW3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)#no vlan 16,274
SW3(config)#end
SW3#
```

Verify the IP connectivity on SW3:

```
SW3#ping 255.255.255.255
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 255.255.255.255, timeout is 2 seconds:

Reply to request 0 from 158.10.24.40, 1 ms
Reply to request 0 from 158.10.24.20, 1 ms
Reply to request 1 from 158.10.24.40, 1 ms
Reply to request 1 from 158.10.24.20, 1 ms
Reply to request 2 from 158.10.24.40, 2 ms
Reply to request 2 from 158.10.24.20, 2 ms
Reply to request 3 from 158.10.24.20, 1 ms
Reply to request 3 from 158.10.24.40, 1 ms
Reply to request 4 from 158.10.24.40, 5 ms
Reply to request 4 from 158.10.24.20, 6 ms
SW3#
```

SW3 has full IP connectivity on the 158.10.24.0/24 subnet.

1.4. Symptom: SW4 is not in VTP transparent mode.

Analysis and testing:

The scenario requires that all switches be in VTP Transparent mode. On each switch, examine the output of the command **show vtp status**.

```
SW4#show vtp status
VTP Version capable             : 1 to 3
VTP version running             : 1
VTP Domain Name                 :
VTP Pruning Mode                : Disabled
VTP Traps Generation            : Disabled
Device ID                       : aabb.cc00.0a00
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
--------------
VTP Operating Mode              : Server
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 6
Configuration Revision          : 0
MD5 digest                      : 0xE7 0xAB 0xFC 0x24 0x2F 0xA2 0x7B 0x29
                                  0x4E 0xAA 0xB2 0x27 0xFF 0x33 0x8B 0x5F
SW4#
```

Resolution: Change the VTP mode on SW4.

On SW4, enter the command **vtp mode transparent:**

```
SW4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW4(config)#vtp mode transparent
Setting device to VTP Transparent mode for VLANS.
SW4(config)#end
SW4#
```

Make sure that VLAN 56 is present in the VLAN table on SW4:

```
SW4#sh vlan brie

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et0/0, Et0/1, Et0/2, Et0/3
                                                Et1/0, Et1/1, Et1/3, Et2/0
                                                Et2/1, Et2/3
56   VLAN0056                         active    Et1/2, Et2/2
1002 fddi-default                     act/unsup
```



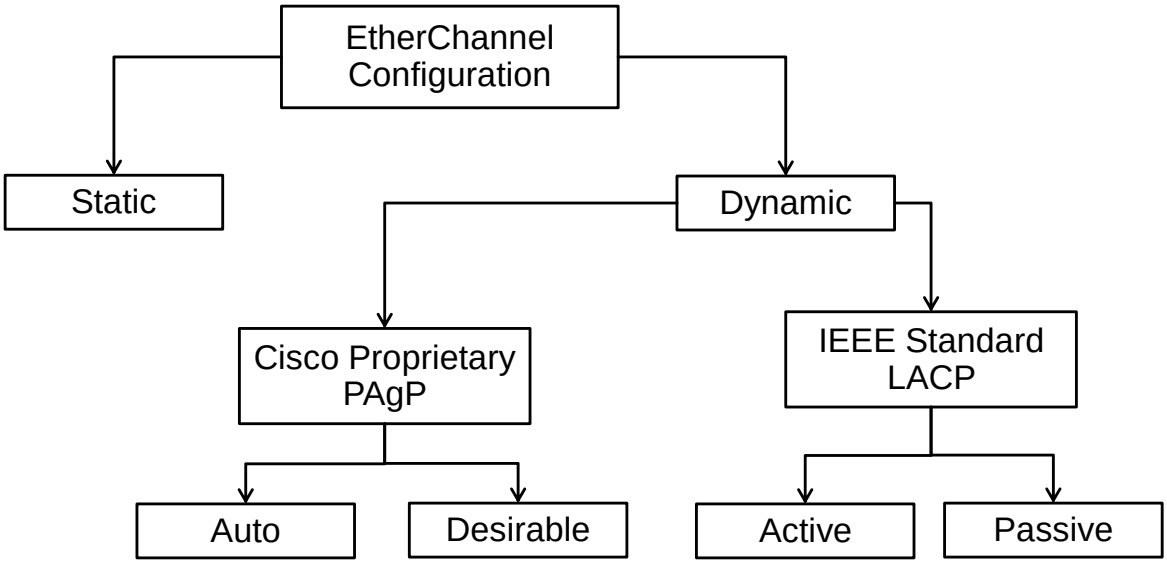
```
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
SW4#
```

1.5. Symptom: EtherChannel is not correctly configured between SW1 and SW2.

Analysis and testing:

By reviewing the supplied diagram, you can see that EtherChannel is configured between SW1 and SW2. Once this has been determined, the following EtherChannel configuration options must be determined. Review the following options analysis diagram:

EtherChannel Configuration Diagram



Given this options analysis diagram, review the specifications of the lab itself. In the "Expected Behavior and Network Policies" section of the scenario, it clearly states:

- The link between SW1 and SW2 must be LACP-negotiated.

Verify the initial configuration of this link:

```
SW1#show etherchannel summary
```

```
Flags:  D - down        P - in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)         PAgP      Et2/2(P)    Et2/3(P)
```

Clearly, this initial EtherChannel configuration is PAgP. This configuration does not conform to the requirements of this lab. Therefore, it must be changed.

Likely cause: There is a misconfigured EtherChannel protocol between SW1 and SW2.

Check the configuration between SW1 and SW2. Begin by checking the specific trunk encapsulations for both ends of this connection. First, check on SW1 and SW2:

```
SW1#sh run | begin 2/2
interface Ethernet2/2
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 16,56,274
 switchport mode trunk
 channel-group 1 mode desirable
!
interface Ethernet2/3
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 16,56,274
 switchport mode trunk
 channel-group 1 mode desirable

SW2#sh run | begin 2/2
switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1,16,56,274
 switchport mode trunk
 channel-group 1 mode auto
!
interface Ethernet2/3
switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1,16,56,274
 switchport mode
 trunk channel-group 1 mode auto
```

Look at the output of the **channel-group** commands: “desirable” and “auto.” These Cisco IOS configuration keywords indicate a PAgP configuration.

Resolution: Change the EtherChannel configuration from PAgP to LACP.

While PAgP uses the IOS configuration keywords **desirable** and **auto**, the IOS LACP configuration keywords are **active** and **passive**. Change this configuration using the LACP keywords. As a general practice, many network engineers shut interfaces before changing the EtherChannel mode. Note that the previous EtherChannel mode must be removed before configuring a different mode.

```
SW1(config)#interface range e2/2-3
SW1(config-if-range)#shutdown
```



```
SW1(config-if-range)#no channel-group 1 mode desirable
SW1(config-if-range)#channel-group 1 mode active


SW2(config)#interface range e2/2-3
SW2(config-if-range)#shutdown
SW2(config-if-range)#no channel-group 1 mode auto
SW2(config-if-range)#channel-group 1 mode passive
SW2(config-if-range)#no shut
SW2(config-if-range)#end


SW1(config-if-range)#no shut
SW1(config-if-range)#end
```

With this configuration change made, verify that LACP is now running on this EtherChannel link:

```
SW1#show etherchannel summary
Flags:  D - down        P - in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)         LACP      Et2/2(P)    Et2/3(P)
```

You see that LACP is now running. This task has been resolved.

1.6. Symptom: The allowed VLANs on the SW1-SW2 trunk are mismatched.

Analysis and testing:

Verify the running configuration of the EtherChannel member interfaces Ethernet2/2 and Ethernet2/3 on SW1 and SW2:

```
SW1#sh run | begin 2/2
interface Ethernet2/2
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 16,56,274
 switchport mode trunk
 channel-group 1 mode active
!
interface Ethernet2/3
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 16,56,274
 switchport mode trunk
 channel-group 1 mode active

SW2#sh run | begin 2/2
interface Ethernet2/2
switchport trunk encapsulation dot1q
 switchport trunk allowed vlan  1,16,56,274
 switchport mode trunk
```

```
 channel-group 1 mode passive
!
interface Ethernet2/3
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan  1,16,56,274
 switchport mode trunk
 channel-group 1 mode passive
```

Note that the allowed VLANs listed on SW1 ports Et2/2 and Et2/3 include 16, 56 and 274; however, the allowed VLANs listed on SW3 ports Et2/2 and Et2/3 include the same three VLANs as well as an additional VLAN, VLAN 1.

In the "Expected Behavior and Network Policies" subsection of the "Switched Network Troubleshooting" section of this lab, it explicitly states the following:

- The trunk between SW1 and SW2 should permit only the required VLANs.

When you review the supplied VLAN diagram, you see that none of the switches in the lab are using VLAN 1. This configuration does not conform to the requirement in the "Expected Behavior and Network Policies" subsection.

Likely cause: The** allowed vlan **command is misconfigured on ports Et2/2 and Et2/3 on SW2.

This can be verified with the following IOS **show** command:

```
SW1#show interfaces trunk

Port        Mode         Encapsulation  Status        Native vlan
Po1         on           802.1q         trunking      1

Port        Vlans allowed on trunk
Po1         16,56,274

Port        Vlans allowed and active in management domain
Po1         16,56,274

Port        Vlans in spanning tree forwarding state and not pruned
Po1         16,56,274

SW2#sh interfaces trunk

Port        Mode         Encapsulation  Status        Native vlan
Po1         on           802.1q       trunking      1

Port        Vlans allowed on trunk
Po1         1,16,56,274

Port        Vlans allowed and active in management domain
Po1         1,16,56,274

Port        Vlans in spanning tree forwarding state and not pruned
Po1         1,16,56,274
```

Notice that the two trunks are represented in the **show interfaces trunk** output as a PortChannel (Po1). Still, you can see that SW1 is allowing VLANs 16, 56 and 274 on its Po1 implementation and SW2 is allowing VLANs 1, 16, 56 and 274.

Resolution: Remove VLAN 1 from the "Allowed VLANs" list on interface Po1 on SW2.

Once an EtherChannel is formed, it is best to make any changes to the trunking configuration on the PortChannel interface, not on the physical interfaces. Changes made to Po1 will automatically be applied to the physical ports. On SW2 interface Po1, enter the command

switchport trunk allowed VLAN remove VLAN 1, or the command **switchport trunk allowed vlan 16,56,274**.

```
SW2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW2(config)#int po1
SW2(config-if)#switchport trunk allowed vlan 16,56,274
SW2(config-if)#end
SW2#show interfaces trunk

Port        Mode             Encapsulation  Status        Native vlan
Po1         on               802.1q         trunking      1

Port        Vlans allowed on trunk
Po1         16,56,274

Port        Vlans allowed and active in management domain
Po1         16,56,274

Port        Vlans in spanning tree forwarding state and not pruned
Po1         16,56,274
SW2#
```

Note To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

To learn more about Cisco Catalyst switch troubleshooting methods and techniques, download and watch the VoD sessions from the Cisco 360 "Troubleshooting" lesson module. This lesson module contains more than 8 hours of video content that is dedicated to the subject of troubleshooting.

2. IPv4 OSPF Troubleshooting Section

2.1. Symptom: There is no OSPF adjacency between R1 and R6.

Analysis and testing:

After reviewing the lab diagram, you can see that this lab has a simple OSPF topology consisting of the single VLAN between R1 and R6 and Loopback 106 on R6. Initially, there is no OSPF adjacency between R1 and R6.

```
R1#show ip ospf neighbor
R1#
```

Begin your investigation with a command that is often used to troubleshoot OSPF: the **show ip ospf interface** command:

```
R1#sh ip ospf interface e0/0
Ethernet0/0 is up, line protocol is up
  Internet Address 158.10.16.1/24, Area 0, Attached via Network Statement
  Process ID 1, Router ID 158.10.124.1, Network Type BROADCAST, Cost: 10
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           10        no          no            Base
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 158.10.124.1, Interface address 158.10.16.1
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:02
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
```

```
  IETF NSF helper support enabled
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 0
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)

R6#show ip ospf interface e0/1
Internet Address 158.10.16.6/22, Area 0, Attached via Network Statement
  Process ID 1, Router ID 158.10.106.1, Network Type BROADCAST, Cost: 10
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           10        no          no            Base
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 158.10.106.1, Interface address 158.10.16.6
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:04
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 0
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
```

After running this command on both sides of the OSPF connection, you'll note two characteristics of this configuration: one positive and one negative.

The positive characteristic is that OSPF is definitely enabled on the correct interfaces, and both OSPF-speaking routers – R1 and R6 – share the same OSPF network type.

The negative characteristic is that neither R1 nor R6 has discovered each other and consequently, they have not formed an adjacency.

Given this abnormal situation, verify that both R1 and R6 can reach each other via ping:

```
R1#ping 158.10.16.6
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 158.10.16.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

R6#ping 158.10.16.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 158.10.16.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

The ping test verified reachability. Now, enable an IOS debug utility—**debug ip ospf hello**—to determine what is preventing R1 and R6 from forming an adjacency:

```
R1#debug ip ospf hello
OSPF hello events debugging is on
*Oct 19 07:22:39.006: OSPF: Rcv hello from 158.10.106.1 area 0 from Ethernet0/0
158.10.16.6
*Oct 19 07:22:39.006: OSPF: Mismatched hello parameters from 158.10.16.6
*Oct 19 07:22:39.006: OSPF: Dead R 40 C 40, Hello R 10 C 10  Mask R 255.255.252.0 C
255.255.255.0
```


Immediately, the **debug ip ospf hello** utility uncovers a problem. There are mismatched OSPF hello parameters between R1 and R6.

Likely cause: There are mismatched OSPF hello parameters between R1 and R6.

When reading the debug output of **debug ip ospf hello**, three parameters are listed: a dead timer, a hello timer, and a prefix mask length. Each of these is represented with a "C" and an "R." The "C" stands for locally connected. This represents the locally set OSPF interface parameters. The "R" stands for remote. This represents the remotely connected neighbors OSPF interface parameters.

After inspecting all three of these parameter pairs exchanged between R1 and R6, you can see that the hello and dead timers are equal. However, the prefix mask lengths are not equal. R1 possesses a 24-bit mask length and R6 possesses a 22-bit mask length. For an OSPFv2 adjacency to form, these parameters must match between two OSPF neighbors.

The differing mask lengths can be further verified with the following command:

```
R1#sh inte e0/0 | include Internet address
  Internet address is 158.10.16.1/24

R6#sh inte e0/1 | include Internet address
  Internet address is 158.10.16.6/22
```

Resolution: Set the prefix mask lengths on both R1 and R6 to be equal.

The "Restrictions and Goals" section states that all addresses beginning with 158.10 will be assigned /24 prefix masks. Therefore, the R6 prefix mask must be changed from 22 bits to 24 bits:

```
R6#conf t
R6(config)#inte e0/1
R6(config-if)#ip address 158.10.16.6 255.255.255.0
R6(config-if)#end
R6#

*Oct 19 07:30:49.506: %OSPF-5-ADJCHG: Process 1, Nbr 158.10.124.1 on Ethernet0/1
from LOADING to FULL, Loading Done
```

As soon as this change is made, a console message displays, indicating that an OSPF adjacency has been formed with R1. This can be verified with the following IOS **show** command:

```
R6#show ip ospf neighbor
Neighbor ID     Pri   State           Dead Time   Address         Interface
158.10.124.1      1   FULL/DR         00:00:32    158.10.16.1     Ethernet0/1
```

Note	To learn more about OSPF troubleshooting methods and techniques, download and watch the VoD sessions from the Cisco 360 "Troubleshooting" lesson module. This lesson module contains more than 8 hours of video content that is dedicated to the subject of troubleshooting.

3. IPv4 EIGRP Troubleshooting Section

3.1. Symptom: EIGRP is not enabled on the R1 Ethernet0/2 interface.

Analysis and testing:

When the following basic EIGRP IOS **show** command is entered, no output is generated:

```
R1#sh ip eigrp interfaces
IP-EIGRP interfaces for process 100

                        Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
R1#
```

Only the Cisco IOS command prompt is returned, so there is a basic and fundamental problem with EIGRP. While there could be many reasons why an EIGRP neighbor relationship is not formed, there are far fewer reasons why EIGRP is not enabled on a specific interface. It is likely that the problem resides in the basic configuration of EIGRP.

Likely cause: An EIGRP network configuration command is misconfigured.

This can be easily checked with the following IOS **show** command. Note the use of the powerful **section** option. This allows for the viewing of specific sections of IOS **show** command output:

```
R1#sh run | section eigrp
router eigrp 100
 redistribute connected metric 1000 100 255 1 1500
 redistribute ospf 1 metric 1000 100 255 1 1500
 network 158.10.124.0 0.0.0.0
 auto-summary
 neighbor 158.10.124.2 Ethernet0/2
 neighbor 158.10.124.4 Ethernet0/2
 redistribute eigrp 100 subnets
  redistribute eigrp 100
```

As you can see , there is only one EIGRP network configuration command in this section. It is incorrectly configured. It specifies only a prefix address of 158.10.124.0; however, its wildcard mask is all zeros—0.0.0.0. A 0.0.0.0 wildcard mask specifies an exact match of an IP address. The 158.10.124.0 address is not assigned to the R1 Ethernet0/2 interface. This is only the prefix assigned to the R1 Ethernet0/2 interface.

Resolution: Adjust the R1 EIGRP network statement so that it is syntactically correct.

Two options exist to correct the R1 EIGRP network statement:

1) The address can be adjusted to match the current wildcard mask.

2) The wildcard mask can be adjusted to match the current prefix of 158.10.124.0.

Apply the first option to correct the problem:

```
R1#conf t
R1(config)#router eigrp 100
R1(config-router)#network 158.10.124.1 0.0.0.0
R1(config-router)#end

*Oct 19 07:53:10.326: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 158.10.124.2
(Serial0/0/0) is up: new adjacency
```



```
*Oct 19 07:53:10.334: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 158.10.124.4
(Serial0/0/0) is up: new adjacency
```

As soon as the configuration change has been made, the two expected EIGRP neighbor relationships on the 158.10.124.0/24 subnet are formed. Clearly, EIGRP is now configured on the Ethernet0/2 interface of R1. This can be verified with the following IOS **show** command:

```
R1#show ip eigrp interfaces
IP-EIGRP interfaces for process 100

                      Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Se0/0/0            2        0/0       832       0/15         4162           0
```

3.2. Symptom: Full EIGRP redundancy is not achieved.

Analysis and testing:

The "Description of the Topology" subsection of the "EIGRP Troubleshooting" section states the following:

As indicated in the IPv4 IGP diagram, the EIGRP routing domain possesses redundant paths for routes that are learned within the domain.

The "Expected Behavior and Network Policies" subsection states the following:

Because the EIGRP routing domain possesses redundant paths, make sure that all paths are used and load balancing is maintained on SW3 for the 158.10.124.0/24 subnet.

Given these requirements, full redundancy is not achieved in this lab. This is based upon the following **show ip eigrp topology** output for two specified loopback addresses:

158.10.102.0/24 – originating on router R2

158.10.104.0/24 – originating on router R4

You can investigate further with the following EIGRP **show** command:

```
R2#sh ip eigrp topology 158.10.104.0/24
IP-EIGRP (AS 100): Topology entry for 158.10.104.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 156160
  Routing Descriptor Blocks:
  158.10.24.40 (Ethernet0/0), from 158.10.24.40, Send flag is 0x0
      Composite metric is (156160/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 5100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1

R4#sh ip eigrp topology 158.10.102.0/24
IP-EIGRP (AS 100): Topology entry for 158.10.102.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 156160
```

```
  Routing Descriptor Blocks:
  158.10.24.20 (Ethernet0/1), from 158.10.24.20, Send flag is 0x0
      Composite metric is (156160/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 5100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
```

Note that both R2 and R4 maintain only one path to their respective loopback addresses. This path is over the VLAN 274 segment that they both share. Neither R2 nor R4 possess the second path for these prefixes over the Ethernet0/2 interface connection.

Check whether R1 contains routing information on these two loopback addresses:

```
R1#sh ip eigrp topology 158.10.102.0/24
EIGRP-IPv4 Topology Entry for AS(100)/ID(158.10.124.1) for 158.10.102.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 409600
  Descriptor Blocks:
  158.10.124.2 (Ethernet0/2), from 158.10.124.2, Send flag is 0x0
      Composite metric is (409600/128256), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 6000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
        Originating router is 158.10.102.1
  158.10.124.4 (Ethernet0/2), from 158.10.124.4, Send flag is 0x0
      Composite metric is (435200/409600), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2

R1#sh ip eigrp topology 158.10.104.0/24
EIGRP-IPv4 Topology Entry for AS(100)/ID(158.10.124.1) for 158.10.104.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 409600
  Descriptor Blocks:
  158.10.124.4 (Ethernet0/2), from 158.10.124.4, Send flag is 0x0
      Composite metric is (409600/128256), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 6000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
        Originating router is 158.10.104.1
  158.10.124.2 (Ethernet0/2), from 158.10.124.2, Send flag is 0x0
      Composite metric is (435200/409600), route is Internal
      Vector metric:
        Minimum bandwidth is 10000 Kbit
        Total delay is 7000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
```


As you can see, R1 possesses two possible paths for the two EIGRP loopback addresses of 158.10.102.0/24 and 158.10.104.0/24; however, both R2 and R4 possess only one possible path for these prefixes. There seems to be an issue with R1, the hub router in the hub-and-spoke topology on the Ethernet0/2 subnet, advertising these loopback prefixes to its two respective spoke routers R2 and R4.

Likely cause: Split horizon is not disabled on the Ethernet0/2 interface of R1.

From the IOS **show** command output above, it is clear that R1 is receiving routing information from both R2 and R4 on the two loopback prefixes mentioned above. However, even though R1 is receiving this routing information, it is not advertising the information to its next-hop neighbors.

This behavior is caused by the split-horizon rule, which states that a router cannot advertise a prefix on the same interface that prefix was learned on. Therefore, R1 cannot advertise these loopback prefixes on its Ethernet0/2 hub interface to its spoke routers.

The EIGRP split-horizon mechanism is manipulated with its own specific command:

```
R1#conf t
R1(config)#interface Ethernet0/2
R1(config-if)#ip split-horizon ?
  eigrp  Enhanced Interior Gateway Routing Protocol (EIGRP)
  <cr>
```

Note that EIGRP has its own specific split-horizon command that is separate from the generic split-horizon command. Determine whether either of these split-horizon commands is enabled on the Ethernet0/2 interface of R1:

```
R1#sh run interface Ethernet0/2
!
interface Ethernet0/2
 ip address 158.10.124.1 255.255.255.0
 ip pim sparse-mode
end
```

Note that there is no reference to any split-horizon command on this interface. This fact underscores the following rule relating to split horizon and Cisco IOS Software: By default, generic split horizon is enabled on all Ethernet interfaces.

Resolution: Disable EIGRP split horizon on the Ethernet0/2 interface of R1.

Disable EIGRP split horizon and see if R2 and R4 now learn their respective loopback prefixes over the Ethernet0/2 interface connection. When you have finished, enable **debug ip eigrp** to see if any EIGRP prefixes are advertised:

```
R1#debug ip eigrp
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#inte Ethernet0/2
R1(config-if)#no ip split-horizon eigrp 100
```

Once EIGRP split horizon is disabled, note the output from **debug ip eigrp**. It explicitly states that "split horizon changed" and the Ethernet0/2 interface of R1 begins to "advertise out" specific prefixes that it previously did not:

```
*Oct 19 08:39:36.142: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100:
Neighbor 158.10.124.4 (Ethernet0/2) is resync: split horizon changed
*Oct 19 08:39:36.142: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100:
Neighbor 158.10.124.2 (Ethernet0/2) is resync: split horizon changed
*Oct 19 08:39:36.146: IP-EIGRP(Default-IP-Routing-Table:100): 158.10.102.0/24 - do
advertise out Ethernet0/2
*Oct 19 08:39:36.146: IP-EIGRP(Default-IP-Routing-Table:100): 158.10.104.0/24 - do
advertise out Ethernet0/2
```

When you check the EIGRP topology tables of R2 and R4, you see that they now list two possible paths for their respective loopback interfaces:

```
R2#show ip eigrp topology 158.10.104.0/24
IP-EIGRP (AS 100): Topology entry for 158.10.104.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 156160
  Routing Descriptor Blocks:
  158.10.24.40 (Ethernet0/0), from 158.10.24.40, Send flag is 0x0
      Composite metric is (156160/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 5100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
  158.10.124.1 (Ethernet0/2), from 158.10.124.1, Send flag is 0x0
      Composite metric is (2809856/2297856), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 45000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2

R4#sh ip eigrp topology 158.10.102.0/24
IP-EIGRP (AS 100): Topology entry for 158.10.102.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 156160
  Routing Descriptor Blocks:
  158.10.24.20 (Ethernet0/1), from 158.10.24.20, Send flag is 0x0
      Composite metric is (156160/128256), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 5100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
  158.10.124.1 (Ethernet0/2), from 158.10.124.1, Send flag is 0x0
      Composite metric is (2809856/2297856), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 45000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
```

Now the EIGRP route redundancy requirements of this lab have been fulfilled.

Note	To learn more about EIGRP troubleshooting methods and techniques, download and watch the VoD sessions from the Cisco 360 "Troubleshooting" lesson module. This lesson module contains more than 8 hours of video content that is dedicated to the subject of troubleshooting.

4. IPv4 RIP Troubleshooting Section

4.1. Symptom: R6 is not receiving RIP routes from R5.

Analysis and testing:

When the lab topology is reviewed, only R5 and R6 are running RIPv2. Of these two routers, R5 is a complete stub RIP-speaking router. R5 is configured with one loopback interface that has the 158.10.105.1/24 prefix assigned to it. This prefix should be advertised from R5 to R6. However, this subnet is not being advertised, as displayed below in R6's routing table:

```
R6#show ip route rip
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

R6#
```

Clearly, R6 is not receiving any RIP routes from R5. However, R5 is receiving RIP routes from R6:

```
R5#show ip route rip | begin Gate
Gateway of last resort is not set

      158.10.0.0/16 is variably subnetted, 6 subnets, 2 masks
R        158.10.16.0/24 [120/1] via 158.10.56.6, 00:00:15, Ethernet0/1
R        158.10.106.0/24 [120/1] via 158.10.56.6, 00:00:15, Ethernet0/1
R5#
```

Note that the lab requirements state, "Ensure that as many interfaces as possible on RIP-speaking routers are made passive."

Likely cause: The E0/1 interface of R5 is set to passive under the RIP routing configuration.

It is possible that the E0/1 interface of R5 is set to a passive state. This can be verified in two ways using two different Cisco IOS **show** commands. Note that both of these **show** commands will involve the use of the **section** option. The **section** option displays only the **show** command output that is desired:

```
R5#sh run | section router rip
router rip
```

```
 version 2
 passive-interface default
 network 158.10.0.0

R5#sh ip protocols | section Passive
  Passive Interface(s):
    Ethernet0/0
    Ethernet0/1
    Ethernet0/2
    Ethernet0/3
    Serial1/0
    Serial1/1
    Serial1/2
    Serial1/3
    Loopback105
    RG-AR-IF-INPUT1
     VoIP-Null0
    VoIP-Null0
R5#
```

As you can see, the running configuration of the **router rip** process has the **passive-interface default** command configured. As a result, the E0/1 interface is set to passive, as indicated by the **show ip protocols | section Passive** display.

Resolution: Remove the E0/1 RIP interface on R5 from the passive interface state.

Remove the E0/1 interface from the passive state under the RIP routing process on R5, and then clear the IP routing table on R5 to proactively trigger the RIP process on R5 to send a routing advertisement to R6:

```
R5#conf t
R5(config)#router rip
R5(config-router)#no passive-interface e0/1
R5(config-router)#do clea ip ro *
```

Now check the routing table of R6 to see if it has received a RIP route from R5:

```
R6#show ip route rip | beg Gate
Gateway of last resort is not set
     158.10.0.0/24 is subnetted, 9 subnets
R       158.10.105.0 [120/1] via 158.10.56.5, 00:00:09, Ethernet0/0
```

As expected, R6 has received a RIP route from R5.

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

5. IPv4 Redistribution Troubleshooting Section

5.1. Symptom: The 158.10.124.0/24 prefix has not been redistributed into OSPF.

Analysis and testing:

You discovered that R6 cannot ping the 158.10.124.0/24 subnet. Here is a section of example output:

```
R6#ping 158.10.124.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 158.10.124.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R6#
```

Given the initial configuration of two-way redistribution between OSPF and EIGRP on R1, you would expect the 158.10.124.0/24 subnet to be listed in the OSPF database of R1.

```
R1#sh ip ospf database | begin Ex
                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
158.10.13.0     158.10.124.1    555         0x80000064 0x00B9A7 0
158.10.24.0     158.10.124.1    1786        0x80000016 0x00DCC7 0
158.10.56.0     158.10.106.1    53          0x80000079 0x0033FF 0
158.10.102.0    158.10.124.1    1303        0x80000015 0x0081D5 0
158.10.104.0    158.10.124.1    1786        0x80000016 0x0069EA 0
158.10.105.0    158.10.106.1    1542        0x80000014 0x00E086 0
158.10.107.0    158.10.124.1    1786        0x80000016 0x004809 0
```

However, the 158.10.124.0/24 prefix is not listed as an external OSPF prefix, and so it will not be advertised into EIGRP by the redistribution.

Review the redistribution configuration statements on router R1 under the OSPF routing process:

```
R1#sh run | section ^router ospf
router ospf 1
 log-adjacency-changes
 redistribute connected subnets route-map CONNECTED
 redistribute eigrp 100 subnets
 network 158.10.16.1 0.0.0.0 area 0
```

Note that there is both a **redistribute connected** and **redistribute eigrp** statement. The **redistribute connected** statement has a route map associated with it. It is important to note that the missing EIGRP route that was supposed to be redistributed into OSPF is both an EIGRP route *and* a connected route.

Review the route map named CONNECTED, configured on router R1:

```
R1#sh run | section route-map
route-map CONNECTED permit 10
 match ip address 1
```

You see that the route map matches on access list 1. Now, review access list 1:

```
R1#sh access-lists 1
Standard IP access list 1
```

```
10 permit 158.10.13.0 (5 matches)
```

You see that access list 1 matches only one prefix. It does not include the 158.10.124.0/24 prefix.

Likely cause: The route map that is associated with the** redistribute connected **configuration on R1 is too restrictive.

Since the 158.10.124.0/24 prefix is both a connected interface on R1 as well as an EIGRP assigned prefix, R1 must select which routing source is preferred for this prefix. By using the administrative distance value, the connected route source is preferred over all other route sources, including EIGRP. Therefore, the **redistribute connected** configuration takes precedence over the **redistribute EIGRP** configuration on R1.

Resolution: Add an additional line to the access list that is associated with the route map assigned to the** redistribute connected **command on router R1.

First, enable the following **debug** command:

```
R1#debug ip ospf lsa-generation

OSPF summary lsa generation debugging is on
```

With this debug utility enabled, add an additional line to the R1 access list that is associated with the specified route map:

```
R1#conf t
R1(config)#access-list 1 permit 158.10.124.0

<skipped>

*Jun  17  11:41:00.360:  OSPF-1  LSGEN:  Build  external  LSA  158.10.124.0,  mask
255.255.255.0, type 5, age 0, seq 0x80000001
*Jun 17 11:41:00.360: OSPF-1 LSGEN: MTID        Metric        Metric-type        FA
Tag         Topology Name
*Jun 17 11:41:00.360: OSPF-1 LSGEN: 0         20          2                0.0.0.0
0           Base
```

As soon as the 158.10.124.0 prefix is added to the access list that is associated with the route map assigned to the **redistribute connected** command, **debug ip ospf lsa-generation** generates a message advertising this prefix.

You now see the 158.10.124.0/24 prefix listed as an OSPF external route:

```
R1#sh ip ospf database | begin Ex
                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
158.10.13.0     158.10.124.1    1619        0x80000064 0x00B9A7 0
158.10.24.0     158.10.124.1    842         0x80000017 0x00DAC8 0
158.10.56.0     158.10.106.1    1117        0x80000079 0x0033FF 0
158.10.102.0    158.10.124.1    343         0x80000016 0x007FD6 0
158.10.104.0    158.10.124.1    842         0x80000017 0x0067EB 0
158.10.105.0    158.10.106.1    609         0x80000015 0x00DE87 0
158.10.107.0    158.10.124.1    842         0x80000017 0x00460A 0
158.10.124.0    158.10.124.1    127         0x80000001 0x00B69E 0
```


5.2. Symptom: No OSPF routes are being advertised into the router RIP domain.

Analysis and testing:

Given the initial configuration of two-way redistribution between OSPF and RIPv2 on R6, no redistributed routes reside in the RIP database:

```
R6#sh ip rip database
158.10.0.0/16    auto-summary
158.10.16.0/24    directly connected, Ethernet0/1
158.10.56.0/24    directly connected, Ethernet0/0
158.10.105.0/24
    [1] via 158.10.56.5, 00:00:06, Ethernet0/0
158.10.106.0/24    directly connected, Loopback106
```

Given this condition, examine the **redistribute** commands under the RIP routing process on R6:

```
R6#sh run | section router rip
router rip
 version 2
 redistribute ospf 1
 passive-interface default
 no passive-interface Ethernet0/0
 network 158.10.0.0
 no auto-summary
```

You see that there is a **redistribute** command and a **passive-interface default** command under the RIP process. The **passive-interface default** command is not the problem, because it is followed by a **no passive-interface Ethernet0/0** command.

Therefore, the reason why the OSPF routes are not redistributed into RIP must be related to the **redistribute ospf 1** command configured under the **router rip** process on R6.

Likely cause: No metric is specified for external routes redistributed into RIP. This will cause all routes to attain a metric of 16 hops, which indicates that the route is inaccessible.

If a metric is not explicitly assigned to a dynamic routing protocol that is being redistributed into either RIPv2 or EIGRP, both of these routing protocols assign the redistributed routes a metric of the absolute maximum metric value, which is often called the “infinity” metric value. When a route is advertised with the maximum metric value to indicate that the distance to the route is “infinity,” the route is poisoned and will be advertised as inaccessible.

Please note the exceptions to this rule:

1) When EIGRP routes from one AS are redistributed into another EIGRP AS, no metric is needed and the routes are not poisoned with an “infinity” metric.

2) OSPF does not need to have an explicit metric assigned to routes that are redistributed into it from another dynamic routing protocol. By default, all routes that are redistributed into OSPF are assigned a metric value of 20.

Since no explicit metric is assigned to the OSPF routes redistributed into RIP, all of the routes are poisoned with the maximum metric value of 16. To overcome this problem, three options exist:

1) Assign a metric to the routes redistributed into RIP from OSPF with the metric option associated with the **redistribute** command.

2) Assign a metric to the routes redistributed into RIP from OSPF with a set metric option associated with a route map that is also associated the **redistribute** command.

3) Assign a default metric value under the **router rip** process.

Resolution: Add the** metric **keyword with a metric value when entering the** redistribute ospf **command.

First, enable the following **debug** command:

```
R6#debug ip rip
RIP protocol debugging is on
```

With this debug utility enabled, modify the **redistribute ospf** command under the **router rip** process so that it now includes an explicit **metric** parameter:

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router rip
R6(config-router)#redistribute ospf 1 metric 2

*Oct 19 22:49:58.522: RIP: sending v2 flash update to 224.0.0.9 via Ethernet0/0
(158.10.56.6)
*Oct 19 22:49:58.522: RIP: build flash update entries
*Oct 19 22:49:58.522:   158.10.13.0/24 via 0.0.0.0, metric 2, tag 0
*Oct 19 22:49:58.522:   158.10.24.0/24 via 0.0.0.0, metric 2, tag 0
*Oct 19 22:49:58.522:   158.10.102.0/24 via 0.0.0.0, metric 2, tag 0
*Oct 19 22:49:58.522:   158.10.104.0/24 via 0.0.0.0, metric 2, tag 0
*Oct 19 22:49:58.522:   158.10.107.0/24 via 0.0.0.0, metric 2, tag 0
*Oct 19 22:49:58.522:   158.10.124.0/24 via 0.0.0.0, metric 2, tag 0
```

As you can see, the OSPF external routes are now advertised. Also, when the **show ip rip** database is examined, you now see entries that are listed as redistributed:

```
R6#show ip rip database | include redistributed
158.10.13.0/24    redistributed
158.10.24.0/24    redistributed
158.10.102.0/24    redistributed
158.10.104.0/24    redistributed
158.10.107.0/24    redistributed
158.10.124.0/24    redistributed
```

All of these routes are the routes imported into RIP via the redistribution of OSPF routes.

Now, the redistributed routes are being advertised with the exact metric that was specified in the **redistribute** command—metric 2. In conclusion, when redistributing dynamic routing protocols into RIP, always explicitly specify a metric. If you do not do so, a metric of 16 will be used and all routes will be poisoned during the redistribution process.

Now that you seem to have addressed all the IPv4 unicast issues, you will test reachability using this simple Tool Command Language (Tcl) script; enter the command **tclsh** and paste in this script. When it is complete, you will have a record of successful and unsuccessful pings. Enter the command **tclquit** to exit the command interpreter. If you have not yet completed the BGP section, then you may find that the IP address 158.10.13.3 on R3 is not yet reachable from all routers. Note that the IP addresses in the Customer A VRF are not included in this script, and do not have to be reachable from outside that VRF.

```
tclsh
foreach address {
158.10.16.1
158.10.13.1
158.10.124.1
158.10.24.20
158.10.124.2
158.10.102.1
158.10.24.40
158.10.124.4
158.10.104.1
158.10.56.5
158.10.105.1
158.10.16.6
158.10.56.6
158.10.106.1
158.10.24.7
158.10.107.1
} {ping $address}
```

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

6. BGP Troubleshooting Section

6.1. Symptom: Two BGP routes are missing from R1's BGP table.

Analysis and testing:

Under the "Expected Behavior and Network Policies" section, the requirements of this lab state the following:

- Redistribution is used to provide connectivity for IGP routes that are redistributed into BGP on R1.

After examining the BGP table on R1, you can see that all IGP routes are present with the exception of those that originated from RIP and were redistributed into OSPF on R6. Examine both the BGP table on R1 and the OSPF database on R1:

```
R1#show ip bgp
BGP table version is 23, local router ID is 158.10.13.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 158.10.16.0/24   0.0.0.0                  0         32768 ?
*> 158.10.24.0/24   158.10.124.2       2172416         32768 ?
```

```
*> 158.10.102.0/24  158.10.124.2       2297856         32768 ?
*> 158.10.104.0/24  158.10.124.4       2297856         32768 ?
*> 158.10.106.1/32  158.10.16.6              2         32768 ?
*> 158.10.107.0/24  158.10.124.2       2300416         32768 ?
*> 158.10.124.0/24  0.0.0.0                   0         32768 ?
```

Note that both the 158.10.56.0/24 and 158.10.105.0/24 prefixes are not listed in the BGP table; however, they are listed in the OSPF database on R1:

```
R1#show ip ospf database | begin Ex
               Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
158.10.13.0     158.10.124.1    1163        0x8000006A 0x00ADAD 0
158.10.24.0     158.10.124.1    416         0x8000001D 0x00CECE 0
158.10.56.0     158.10.106.1    678         0x8000007F 0x002706 0
158.10.102.0    158.10.124.1    1924        0x8000001B 0x0075DB 0
158.10.104.0    158.10.124.1    416         0x8000001D 0x005BF1 0
158.10.105.0    158.10.106.1    169         0x8000001B 0x00D28D 0
158.10.107.0    158.10.124.1    416         0x8000001D 0x003A10 0
158.10.124.0    158.10.124.1    1924        0x80000006 0x00ACA3 0
```

To determine whether there is a configuration error, examine the BGP configuration on R1:

```
R1#sh run | sec bgp
router bgp 13
 bgp router-id 158.10.13.1
 bgp log-neighbor-changes
 neighbor 158.10.13.3 remote-as 13
 !
 address-family ipv4
  redistribute eigrp 100
  redistribute ospf 1
  neighbor 158.10.13.3 activate
  no auto-summary
  no synchronization
 exit-address-family
 !
 address-family vpnv4
  neighbor 158.10.13.3 activate
  neighbor 158.10.13.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CustomerA
  redistribute connected
  no synchronization
 exit-address-family
```

Note that there are many address families related to this BGP configuration. While some of these address families are related to MPLS Layer 3 VPN configurations, these sections of the BGP configuration are not relevant to the current issue of why the two RIP originated prefixes of 158.10.56.0/24 and 158.10.105.0/24 are not in the BGP table. In order to determine this, you must limit your analysis to only the "address-family ipv4" section of the BGP configuration.

Under the "address-family ipv4" section of the BGP configuration, you see two **redistribute** commands. Based upon the current BGP table, all EIGRP routes have been redistributed into BGP. However, not all OSPF routes have been redistributed into BGP. Namely, the two routes that originated from RIPv2—158.10.56.0/24 and 158.10.105.0/24—are not in the BGP table. These routes have been successfully redistributed into OSPF, but for some reason, they are not being redistributed as OSPF external routes into BGP.

Likely cause: OSPF external routes are not automatically redistributed into BGP.

Both OSPF and EIGRP make an explicit distinction between internal routes and external routes. Furthermore, OSPF makes a distinction between External Type 1 (OSPF E1 routes) routes and External Type 2 (OSPF E2 routes) routes.

These types of OSPF routes are clearly distinguished by the following **redistribute** command options:

```
R1#conf t
R1(config)#router bgp 13
R1(config-router)# address-family ipv4
R1(config-router-af)#redistribute ospf 1 match ?
  external       Redistribute OSPF external routes
  internal       Redistribute OSPF internal routes
  nssa-external  Redistribute OSPF NSSA external routes
```

Furthermore, the OSPF external routes can be distinguished in the following manner:

```
R1(config-router-af)#redistribute ospf 1 match external ?
  1              Redistribute external type 1 routes
  2              Redistribute external type 2 routes
```

By default, OSPF internal routes are redistributed into BGP, but OSPF external routes are not. In this case, both OSPF internal and external routes need to be redistributed into BGP. This can be accomplished by adding the **match external** option to the **redistribute ospf** command.

Resolution: Add the** match external **option to the OSPF** redistribute **command.

First, enable the following debug command:

```
R1#debug ip bgp updates
BGP updates debugging is on for address family: IPv4 Unicast
```

With this debug utility enabled, modify the **redistribute ospf** command under the "address-family ipv4" section of the BGP configuration section. Add the **match external** option to the **redistribute ospf** command outer BGP process so that it now includes an explicit **metric** parameter:

```
conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router bgp 13
R1(config-router)#address-family ipv4
R1(config-router-af)#redistribute ospf 1 match external
R1(config-router-af)#end
```

As soon as this command is entered, the following **debug ip bgp updates** output is generated:

```
*Oct 20 18:31:09.441: BGP(0): nettable_walker 158.10.56.0/24 route sourced locally
*Oct 20 18:31:09.441: BGP(0): nettable_walker 158.10.105.0/24 route sourced locally
```

As soon as the **match external** option is added to the **redistribute ospf** command, the two OSPF External routes—the 158.10.56.0/24 and the 158.10.105.0/24 prefixes—are redistributed into BGP.

Note that this particular version of Cisco IOS Software automatically adds the **match internal** keyword. If your version does not, then you will need to add it manually.

```
R1#show run | inc internal
  redistribute ospf 1 match internal external 1 external 2
```

Now, all required BGP updates are listed in R1's BGP table:

```
R1#sh ip bgp
BGP table version is 25, local router ID is 158.10.13.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 158.10.16.0/24   0.0.0.0                  0         32768 ?
*> 158.10.24.0/24   158.10.124.2       2172416         32768 ?
*> 158.10.56.0/24   158.10.16.6             20         32768 ?
*> 158.10.102.0/24  158.10.124.2       2297856         32768 ?
*> 158.10.104.0/24  158.10.124.4       2297856         32768 ?
*> 158.10.105.0/24  158.10.16.6             20         32768 ?
*> 158.10.106.1/32  158.10.16.6              2         32768 ?
*> 158.10.107.0/24  158.10.124.2       2300416         32768 ?
*> 158.10.124.0/24  0.0.0.0                  0         32768 ?
```

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

7. MPLS Layer 3 VPN Troubleshooting Section

7.1. Symptom: Ping from R1 to the 172.16.103.1 address on the other side of the MPLS VPN is failing.

Analysis and testing:

Pings from R1 to IP address 172.16.103.1 fail. This destination is in the CustomerA VRF on R3.

```
R1#ping vrf CustomerA 172.16.103.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.103.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

First, check the basic VRF state on both R1 and R3:

```
R1#sh ip vrf
  Name                              Default RD          Interfaces
  CustomerA                         13:1                Lo101

R3#sh ip vrf
  Name                              Default RD          Interfaces
```



```
  CustomerA                        13:1                   Lo103
```

Now, view the following IOS **show** command output related to monitoring the state of MPLS on a specific interface:

```
R1#sh mpls ldp discovery
 Local LDP Identifier:
    158.10.13.1:0
    Discovery Sources:
    Interfaces:
        Serial1/0 (ldp): xmit

R3#sh mpls ldp discovery
R3#
```

Note that R1 has MPLS enabled on its S1/0 interface and R3 does not have MPLS enabled on any of its interfaces.

Next, examine the basic BGP configuration for MPLS Layer 3 VPNs on both R1 and R3:

R1:

```
ip vrf CustomerA
 rd 13:1
 route-target export 13:1
 route-target import 13:1
!
interface Serial1/0
 ip address 158.10.13.1 255.255.255.0
 ip access-group 101 in
mpls ip
!
router bgp 13
 bgp router-id 158.10.13.1
 bgp log-neighbor-changes
 neighbor 158.10.13.3 remote-as 13
 !
 address-family ipv4
  redistribute eigrp 100
  redistribute ospf 1 match internal external 1 external 2
  neighbor 158.10.13.3 activate
 exit-address-family
 !
 address-family vpnv4
  neighbor 158.10.13.3 activate
  neighbor 158.10.13.3 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CustomerA
  redistribute connected
 exit-address-family
```

R3:

```
ip vrf CustomerA
 rd 13:1
 route-target export 13:1
 route-target import 13:1
!
interface Serial1/0 <NO MPLS IP COMMAND ON THIS INTERFACE!!!>
 ip address 158.10.13.3 255.255.255.0
 ip access-group 101 in
```

```
!
router bgp 13
 bgp router-id 158.10.13.3
 bgp log-neighbor-changes
 neighbor 158.10.13.1 remote-as 13
 !
 address-family ipv4
  neighbor 158.10.13.1 activate
 exit-address-family
 !
 address-family vpnv4
  neighbor 158.10.13.1 activate
  neighbor 158.10.13.1 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CustomerA
  redistribute connected
 exit-address-family
```

After reviewing these configurations, you can see that **mpls ip** is not configured on the S1/0 interface on router R3.

Likely cause: There is a missing** mpls ip **interface configuration command on R3.

In order for this configuration to work, it must have the **mpls ip** command configured on both of the S1/0 interfaces of R1 and R3. Based upon the output of the running configuration commands displayed above, this requirement has not been fulfilled. This is further verified by the following IOS **show** command output:

```
R1#sh mpls interfaces
Interface              IP            Tunnel   Operational
Serial1/0              Yes (ldp)     No       Yes

R3#sh mpls interfaces
Interface              IP            Tunnel   Operational
R3#
```

Note that R1 has MPLS enabled on its S1/0 interface, but R3 does not have MPLS enabled on any of its interfaces.

Resolution: Enable** mpls ip **on the S1/0 interface of R3.

Before resolving the issue, enable the following **debug** command on router R3:

```
R3#debug mpls events
MPLS events debugging is on
```

With this debug utility enabled, add the **mpls ip** command under the S1/0 interface on router R3:

```
R3#
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#inte S1/0
R3(config-if)#mpls ip


*Jun 17 12:11:15.356: LDP: Enabling IPv4 Prefix Client on Serial1/0
*Jun 17 12:11:15.356: mpls: Add mpls app; Serial1/0
*Jun 17 12:11:15.356: LDP SB: i/f status change; Serial1/0
*Jun 17 12:11:15.356: ldp: enabling ldp on Serial1/0
```



```
*Jun 17 12:11:15.356: ldp: ldp start; tbl 0
*Jun 17 12:11:15.357: ldp: register with MPLS forwarding
*Jun 17 12:11:15.357: ldp: req startup iprm cef walk (tdp_load_tib); tbl 0
*Jun 17 12:11:15.357: ldp: i/f status change: Serial1/0; cur/des flags 0x2/0x2mcast
1
*Jun 17 12:11:15.357: mpls: Enable MPLS forwarding on Serial1/0
*Jun 17 12:11:15.357: ldp: enable MPLS forwarding on i/f Serial1/0
*Jun 17 12:11:15.357: tagcon: enable dynamic mpls; tbl 0
```

As soon as the **mpls ip** command is entered, the **debug mpls events** output generates messages that MPLS is now enabled on the S1/0 interface. Now, when the following MPLS interface-specific IOS **show** commands are entered, R3's S1/0 interface is listed:

```
R3#sh mpls interfaces
Interface              IP            Tunnel   Operational
Serial1/0              Yes (ldp)     No       Yes

R3#sh mpls ldp discovery
 Local LDP Identifier:
    158.10.13.3:0
    Discovery Sources:
    Interfaces:
        Serial1/0 (ldp): xmit/recv
            LDP Id: 158.10.124.1:0; no host route
```

Even though MPLS is now enabled on the R3 S1/0 interface, the MPLS configuration still has a problem. The remaining MPLS problem is related to the LDP neighbor relationship formation.

7.2. Symptom: MPLS LDP sessions are not active.

Analysis and testing:

The following IOS **show** command indicates that no MPLS LDP sessions are active:

```
R1#show mpls ldp neighbor
R1#
```

While the MPLS configuration is complete, there is a pair of inbound access lists configured on both R1 and R3. According to the requirements of the scenario, these two inbound access lists are restricted to permit only the following traffic on both R1 and R3:

- ICMP any any
- Telnet any any
- Necessary control plane traffic

The control plane traffic will be limited to BGP traffic and MPLS LDP traffic. You can see this in the initial configuration of the original access list supplied on both R1 and R3, shown here:

R1:

```
R1#show run | inc access-list 101
access-list 101 permit icmp any any
access-list 101 permit tcp any eq telnet any
access-list 101 permit tcp any any eq telnet
```

```
access-list 101 permit tcp host 158.10.13.3 host 158.10.13.1 eq bgp
access-list 101 permit udp host 158.10.13.3 eq 646 host 224.0.0.2 eq 646
access-list 101 permit tcp host 158.10.13.3 eq 646 host 158.10.13.1
R1#
```

R3:

```
R3#show run | inc access-list 101
access-list 101 permit icmp any any
access-list 101 permit tcp any eq telnet any
access-list 101 permit tcp any any eq telnet
access-list 101 permit tcp host 158.10.13.1 eq bgp host 158.10.13.3
access-list 101 permit udp host 158.10.13.1 eq 646 host 224.0.0.2 eq 646
access-list 101 permit tcp host 158.10.13.1 host 158.10.13.3 eq 646
R3#
```

Upon initial inspection of both of these inbound access lists, it appears that they are limited to permitting only the desired traffic – ICMP, Telnet, BGP, and MPLS LDP traffic. Given that this is a lab environment, remove the access lists from the S1/0 interfaces of R1 and R3. As soon as you do so, an MPLS LDP neighbor relationship forms, so you conclude that there is something wrong with these access lists. To determine the traffic that is being improperly denied, disable logging to the console, enable logging to the buffer, and add the following statement to the end of each access list:

```
access-list 101 deny ip any any log
```

After a moment, enter the **show logging** command on R1. You will see output similar to the following:

```
R1#
*Jun 17 12:16:41.365: %SEC-6-IPACCESSLOGP: list 101 denied tcp 158.10.13.3(60004) -
> 158.10.13.1(646), 1 packet
R1#
*Jun 17 12:16:46.081: %SEC-6-IPACCESSLOGP: list 101 denied tcp 158.10.13.3(43602) -
> 158.10.13.1(646), 1 packet
R1#
```

The access list denies traffic sourced from R3 IP address 158.10.13.3 to R1 IP address 158.10.13.1. Note that the destination TCP port is 646, the port used by LDP.

Likely cause: TCP port numbers are in the wrong order for the LDP control plane traffic.

When examining the LDP control plane connection setup between R1 and R3, it can be seen that R3 has a higher router ID than R1. Therefore, R3 will initiate the TCP session to the 158.10.13.1 address on R1 to the TCP destination port of 646. If TCP port 646 is not permitted from R3's 158.10.13.3 address, the LDP session will not form.

With that said, review the very last line of R1's inbound S1/0 access list:

```
Access-list 101 permit tcp host 158.10.13.3 eq 646 host 158.10.13.1
```

Note that in this access list line, TCP port 646 is associated with the TCP source port on R3. Given the router ID assignments on R1 and R3, TCP port 646 should be the destination port on R1, not the source port on R3.

Resolution: Modify R1's inbound S1/0 access list so that the last line references TCP port 646 as a destination port rather than a source port.

Once this change is made, R1's working access list configuration should be listed as follows:

```
access-list 101 permit icmp any any
access-list 101 permit tcp any eq telnet any
access-list 101 permit tcp any any eq telnet
access-list 101 permit tcp host 158.10.13.3 host 158.10.13.1 eq bgp
access-list 101 permit udp host 158.10.13.3 eq 646 host 224.0.0.2 eq 646
access-list 101 permit tcp host 158.10.13.3 host 158.10.13.1 eq 646
```

The access list on R3 should be changed:

```
access-list 101 permit icmp any any
access-list 101 permit tcp any eq telnet any
access-list 101 permit tcp any any eq telnet
access-list 101 permit tcp host 158.10.13.1 eq bgp host 158.10.13.3
access-list 101 permit udp host 158.10.13.1 eq 646 host 224.0.0.2 eq 646
access-list 101 permit tcp host 158.10.13.1 eq 646 host 158.10.13.3
```

Once these steps are complete, an MPLS LDP neighbor relationship should form, as you see here:

```
R3#show mpls ldp nei
    Peer LDP Ident: 158.10.13.1:0; Local LDP Ident 158.10.13.3:0
        TCP connection: 158.10.13.1.646 - 158.10.13.3.64214
        State: Oper; Msgs sent/rcvd: 5/14; Downstream
        Up time: 00:01:08
        LDP discovery sources:
          Serial1/0, Src IP addr: 158.10.13.1
        Addresses bound to peer LDP Ident:
          158.10.16.1     158.10.124.1    158.10.13.1
```

Try to ping the CustomerA 172.16.103.1 from R1 again:

```
R1#ping vrf CustomerA 172.16.103.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.103.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/8/9 ms
R1#
```

Note that the ping is successful.

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

8. EEM Troubleshooting Section

8.1. The EEM applet does not clear the Ethernet0/1 interface counters on R5.

Analysis and testing:

You manually ran the **event manager run CLEAR** command on R5 and verified the interface counters using the **show interface e0/1** command on R5, but you see that the counters are not cleared.

Verify EEM policy on R5:

```
R5#show event manager policy registered
No.  Class   Type    Event Type          Trap  Time Registered           Name
1    applet  user    none                Off   Wed Oct 26 18:07:58 2011  CLEAR
 policyname {CLEAR} sync {yes}
 maxrun 20.000
 action 1.0 cli command "enable"
 action 2.0 cli command "clear counters e0/1"
 action 3.0 cli command "y"

R5#
```

Note that the EEM applet CLEAR is configured and correctly registered as **event none** on R5. The EEM **event none** command allows running the EEM applet manually with the **event manager run** command. You confirmed that the applet is configured correctly for the manual execution.

Run the **debug event manager action cli** command on R5 and start the EEM applet CLEAR:

```
R5#debug event manager action cli
Debug EEM action cli debugging is on
R5#event manager run CLEAR

*Jun 16 21:21:23.571: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : CTL : cli_open
called.
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : ----------
----------------------------------
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Cisco 360 R&S
Exercise Workbook
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Product, POD
location:  cierswbv5-te-lab07-sc, SJ
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Device:
R5
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : ----------
----------------------------------
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT :
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : R5>
*Jun 16 21:21:23.575: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : IN  : R5>enable
*Jun 16 21:21:23.697: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : R5#
*Jun 16 21:21:23.697: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : IN  : R5#clear
counters e0/1
R5#
*Jun 16 21:21:43.625: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : CTL : cli_close
called.
*Jun 16 21:21:43.625:
*Jun 16 21:21:43.625: tty is now going through its death sequence
R5#
```

Note that the **clear counters e0/1** command is not executed properly. This command is interactive, so it expects the confirmation prompts:

```
R5#clear counters e0/1
Clear "show interface" counters on this interface [confirm]y
R5#
```

Likely cause: Expected prompts are misconfigured in the EEM applet configuration on R5.

Verify the EEM configuration on R5:

```
R5#show run | section CLEAR
```



```
event manager applet CLEAR
 event none
 action 1.0 cli command "enable"
 action 2.0 cli command "clear counters e0/1"
 action 3.0 cli command "y"
```

Fix the EEM applet CLEAR configuration and verify the applet operations again:

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#event manager applet CLEAR
R5(config-applet)# action 2.0 cli command "clear counters e0/1" pattern "confirm]"
R5(config-applet)#end
R5#


R5#event manager run CLEAR
R5#
*Jun 16 21:32:20.177: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : CTL : cli_open
called.
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : --------------
---------------------------------
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Cisco 360 R&S
Exercise Workbook
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Product, POD
location:  cierswbv5-te-lab07-sc, SJ
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Device:
R5
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : --------------
---------------------------------
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT :
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : R5>
*Jun 16 21:32:20.182: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : IN  : R5>enable
*Jun 16 21:32:20.301: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : R5#
*Jun 16 21:32:20.301: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : IN  : R5#clear
counters e0/1
*Jun 16 21:32:20.421: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : Clear "show
interface" counters on this interface [confirm]
*Jun 16 21:32:20.421: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : IN  : y
*Jun 16 21:32:20.422: %CLEAR-5-COUNTERS: Clear counter on interface Ethernet0/1 by
on vty0 (EEM:CLEAR)
R5#
*Jun 16 21:32:20.438: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : y
*Jun 16 21:32:20.438: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : R5#
*Jun 16 21:32:20.438: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : OUT : R5#
*Jun 16 21:32:20.438: %HA_EM-6-LOG: CLEAR : DEBUG(cli_lib) : : CTL : cli_close
called.
*Jun 16 21:32:20.438:
*Jun 16 21:32:20.438: tty is now going through its death sequence
R5#
R5#u all
All possible debugging has been turned off
R5#
```

Note the EEM applet CLEAR works correctly now. The interface E0/1 counters are cleared, as you can see from this line of output:

```
"%CLEAR-5-COUNTERS: Clear counter on interface Ethernet0/1 by on vty0
(EEM:CLEAR)"
```

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

9. IP QoS Troubleshooting Section

9.1. Symptom: Shaping for FTP traffic allows too high a burst.

Analysis and testing:

Verify the QoS configuration on R1:

```
R1#show policy-map interface
 Ethernet0/0

  Service-policy output: shape_traffic

    Class-map: DSCP45 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: ip dscp 45
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      shape (average) cir 256000, bc 1024, be 1024
      target shape rate 256000

    Class-map: FTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: protocol ftp
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      shape (peak) cir 256000, bc 8000, be 32000
      target shape rate 1280000

    Class-map: class-default (match-any)
      757 packets, 68533 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 757/68533
R1#
```

Note that the burst rate for the FTP traffic is 1,280,000 b/s. This lab requires the maximum burst rate of 768,000 b/s.

Likely cause: The shaping token bucket algorithm is misconfigured.

Here is the formula for the shape peak or the maximum burst rate calculation:

PIR = CIR (1 + Be / Bc)

where the PIR is a peak rate, the CIR is a committed rate, Bc is the sustained bits transmitted during each interval Tc, and the Be is the excess data allowed to be sent during the first interval Tc.

The current CIR is calculated as following:

CIR = 256,000

Bc = 8000

Be = 32,000

PIR = 256,000 (1 + 32,000 / 8000) = 256,000 * 5 = 1280000

According to the lab requirements, the PIR should be set to 768,000 b/s, the CIR should be 256,000 b/s, and the Bc should be 8000 b/s. You need to find the correct Be value.

768,000 = 256,000 (1 + Be / 8000)

1 + Be / 8000 = 768,000 / 256,000 = 3

Be / 8000 = 2

Be = 8000 * 2

Be = 16,000

Resolution: Change the Be parameter in class map FTP to 16000.

Here is the incorrect configuration on R1:

```
class-map match-all DSCP45
 match ip dscp 45
class-map match-all FTP
 match protocol ftp
!
policy-map shape_traffic
 class DSCP45
  shape average 256000
 class FTP
  shape peak 256000 8000 32000
!
```

Correct the Be configuration on R1:

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#policy-map shape_traffic
R1(config-pmap)#class FTP
R1(config-pmap-c)#  shape peak 256000 8000 16000
R1(config-pmap-c)#end
R1#
```

Verify the QoS configuration on R1 again:

```
R1#show policy-map interface e0/0
 Ethernet0/0

  Service-policy output: shape_traffic

    Class-map: DSCP45 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: ip dscp 45
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      shape (average) cir 256000, bc 1024, be 1024
      target shape rate 256000

    Class-map: FTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: protocol ftp
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      shape (peak) cir 256000, bc 8000, be 16000
      target shape rate 768000

    Class-map: class-default (match-any)
      1223 packets, 110001 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 1223/110001
R1#
```

FTP traffic can now burst to 768,000 b/s in the first interval, assuming that credit has been built up.

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.

10. IP Multicast Troubleshooting Section

10.1. Symptom: R6 is not receiving an ICMP echo response back from SW3.

Analysis and testing:

According to the requirements of this scenario, R6 is acting as the multicast source router and SW3 is acting as the one multicast receiver. Also, the multicast routing protocol is PIM sparse mode, and R1 is configured as the rendezvous point. However, R6 is not receiving a reply from SW3:

```
R6#ping 239.255.1.1 rep 10000 source Ethernet 0/1

Type escape sequence to abort.
Sending 10000, 100-byte ICMP Echos to 239.255.1.1, timeout is 2 seconds:
```



```
Packet sent with a source address of 158.10.16.6
.....
```

After examining this configuration on a hop-by-hop basis, you discover that R1 is receiving the traffic from R6 as manifested in the following **show ip mroute** display:

```
R1#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.1.1), 00:33:13/stopped, RP 158.10.124.1, flags: SP
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(158.10.16.6, 239.255.1.1), 00:27:07/00:02:52, flags: PT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list: Null

(*, 224.0.1.40), 00:33:38/00:02:50, RP 158.10.124.1, flags: SJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/0/0, Forward/Sparse, 00:33:33/00:02:47
```

Clearly an S,G entry has been created for the 158.10.16.6, 239.255.1.1 multicast group. Based on the following output, the packet count is incrementing:

```
R1#sh ip mroute count | section 239.255.1.1
Group: 239.255.1.1, Source count: 1, Packets forwarded: 105, Packets received: 18
  RP-tree: Forwarding: 0/0/0/0, Other: 0/0/0
  Source: 158.10.16.6/32, Forwarding: 105/-1/100/0, Other: 865/0/760

R1#sh ip mroute count | section 239.255.1.1
Group: 239.255.1.1, Source count: 1, Packets forwarded: 105, Packets received: 21
  RP-tree: Forwarding: 0/0/0/0, Other: 0/0/0
  Source: 158.10.16.6/32, Forwarding: 105/-1/100/0, Other: 867/0/762
```

However, there is no list of interfaces in the outgoing interface list of the mroute table. Check R1's PIM neighbors. R1 should have two of them—R2 and R4.

```
R1#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      S - State Refresh Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
158.10.124.2      Ethernet0/2              13:15:43/00:01:15 v2    1 / DR S P G
```

R1 has only one PIM neighbor—R2. For some reason, R4 is not listed as a PIM neighbor.

Now check R4:

```
R4#sh ip pim neighbor
```

```
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      S - State Refresh Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
158.10.24.20      Ethernet0/1              12:47:02/00:01:30 v2    1 / S P G
158.10.24.7       Ethernet0/1              12:47:02/00:01:38 v2    1 / S P G
```

R4 has two PIM neighbors, both of which reside on VLAN 274. R4 does not have a PIM neighbor relationship with R1 over the 158.10.124.0/24 network. Examine this interface from a PIM perspective:

```
R4#sh ip pim interface e0/2

Address          Interface                Ver/   Nbr    Query  DR     DR
                                          Mode   Count  Intvl  Prior

R4#sh ip pim interface

Address          Interface                Ver/   Nbr    Query  DR     DR
                                          Mode   Count  Intvl  Prior
158.10.24.40     Ethernet0/1           v2/S   2      30     1      158.10.24.40
```

Only R4's E0/1 interface is listed as a PIM-enabled interface. Check the running configuration of the R4 Ethernet0/2 interface:

```
R4#sh run inte e0/2
!
interface e0/2
 ip address 158.10.124.4 255.255.255.0
end
```

You can see that this interface has no PIM configuration implemented on it. This is why R1 has no PIM neighbor relationship with R4 over the Ethernet0/2 connection. How does the fact that R4 does not have PIM enabled on its Ethernet0/2 interface prevent the multicast ping from R6 to SW3 from being successful? There is an alternative path to SW3 via R1 to R2.

Likely cause: R4 is the PIM-designated router on the VLAN 274 network. Since it does not have PIM enabled on its Ethernet0/2 interface, R4 is unable to send PIM messages to R1, the rendezvous point.

Because R4 has the highest IP address on the VLAN 274 link, it is elected as the PIM DR on this link. When SW3 sends a PIM join message to indicate that it has an active multicast receiver for the 239.255.1.1 multicast group, SW3 sends the PIM join message to R4, the designated router for VLAN 274. This can be seen by enabling **debug ip pim** on SW3:

```
*Jun 18 06:35:42.651: PIM(0): Building Periodic (*,G) Join / (S,G,RP-bit) Prune
message for 239.255.1.1
*Jun 18 06:35:42.651: PIM(0): Insert (*,239.255.1.1) join in nbr 158.10.24.40's
queue
*Jun 18 06:35:42.651: PIM(0): Building Join/Prune packet for nbr 158.10.24.40
*Jun 18 06:35:42.651: PIM(0):  Adding v2 (158.10.124.1/32, 239.255.1.1), WC-bit,
RPT-bit, S-bit Join
*Jun 18 06:35:42.651: PIM(0): Send v2 join/prune to 158.10.24.40 (Ethernet1/3)
```

You can also see this by examining the contents of SW3's PIM neighbor table:

```
SW3#sh ip pim neighbor
PIM Neighbor Table
Neighbor          Interface                Uptime/Expires    Ver   DR
```



```
Address                                                          Prio/Mode
158.10.24.40      Ethernet1/3             12:48:44/00:01:32 v2   1 / DR S P G
158.10.24.20      Ethernet1/3             13:02:29/00:01:17 v2   1 / S P G
```

Out of the two neighbors listed in SW3's PIM neighbor table, router R4 is listed as the DR because it has the highest IP address among all of the PIM routers on VLAN 274. The highest IP address is the criterion for electing the DR on a segment for PIM.

Resolution: Configure PIM-SM on the Ethernet0/2 interface of R4.

```
R4#conf t
R4(config)#interface e0/2
R4(config-if)#ip pim sparse-mode
R4(config-if)#end
```

As soon as this command is entered on R4, the multicast ping on R6 is successful:

```
R6#ping 239.255.1.1 rep 10000 source 158.10.16.6

Type escape sequence to abort.
Sending 10000, 100-byte ICMP Echos to 239.255.1.1, timeout is 2 seconds:
Packet sent with a source address of 158.10.16.6

Reply to request 342 from 158.10.107.1, 7 ms
Reply to request 343 from 158.10.107.1, 1 ms
Reply to request 344 from 158.10.107.1, 1 ms
Reply to request 345 from 158.10.107.1, 1 ms
Reply to request 346 from 158.10.107.1, 1 ms
Reply to request 347 from 158.10.107.1, 1 ms
Reply to request 348 from 158.10.107.1, 1 ms
Reply to request 349 from 158.10.107.1, 1 ms
```

Note that the **mroute** entry of 158.10.16.6, 239.255.1.1 has an entry in its outgoing interface list:

```
R1#sh ip mroute | section 158.10.16.6, 239.255.1.1
(158.10.16.6, 239.255.1.1), 00:55:15/00:03:24, flags: T
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/2, Forward/Sparse, 00:05:32/00:03:09
```

This indicates that R1 received a PIM-SM join from R4 on this Ethernet0/2 interface.

Note	To obtain a comprehensive view of the configuration tasks in this section, access the Mentor Guide engine. You can enter more than 1000 Cisco IOS Software commands into the engine, as well as a collection of proprietary commands such as **show all**.