Troubleshooting

You can either check items before you start, or begin configuring and address them when they arise. Start with the basics, make sure that your interfaces are not shut down, and that they have the correct addresses and masks.

R3 – wrong subnet mask on Serial1/2 interface.
R1 – configured for "no ip routing"
R4 – keepalives disabled on Serial0/0 interface.

1. Bridging and Switching

Task 1.1

SW1:

```
interface FastEthernet0/16
 switchport trunk encapsulation dot1q
 switchport mode dynamic desirable
 no shutdown
!
interface FastEthernet0/19
 switchport trunk encapsulation dot1q
 switchport mode dynamic desirable
 no shutdown
```

☞ Quick Note
ISL could have been used for this task

SW3 and SW4:

```
interface FastEthernet0/13
 switchport trunk encapsulation dot1q
 switchport mode dynamic desirable
 no shutdown
```

Task 1.1 Verification

```
Rack1SW1#show interfaces trunk | include Mode|desirable
Port        Mode         Encapsulation  Status        Native vlan
Fa0/16      desirable    802.1q         trunking      1
Fa0/19      desirable    802.1q         trunking      1

Rack1SW3#show interfaces trunk | include Mode|desirable
Port        Mode         Encapsulation  Status        Native vlan
Fa0/13      desirable    802.1q         trunking      1

Rack1SW4#show interfaces trunk | include Mode|desirable
Port        Mode         Encapsulation  Status        Native vlan
Fa0/13      desirable    802.1q         trunking      1
```


Task 1.2

SW1 and SW2:

```
interface Port-channel13
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1-6,8-4094
 switchport mode trunk
!
interface FastEthernet0/13
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1-6,8-4094
 switchport mode trunk
 channel-group 13 mode on
!
interface FastEthernet0/14
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1-6,8-4094
 switchport mode trunk
 channel-group 13 mode on
!
interface FastEthernet0/15
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1-6,8-4094
 switchport mode trunk
 channel-group 13 mode on
!
interface range Fa0/13–15
 no shutdown
```

Task 1.3

SW1:

```
system mtu 1504
!
interface FastEthernet0/1
 switchport access vlan 100
 switchport mode dot1q-tunnel
 l2protocol-tunnel cdp
 !
interface FastEthernet0/3
 switchport access vlan 101
 switchport mode dot1q-tunnel
 l2protocol-tunnel cdp
```

☞ Quick Note
A reload will be required if the system MTU was not 1504 before the last reload.


```
SW4:
system mtu 1504
!
interface FastEthernet0/17
 switchport access vlan 101
 switchport mode dot1q-tunnel
 l2protocol-tunnel cdp
 no cdp enable
 no shutdown
!
interface FastEthernet0/18
 switchport access vlan 100
 switchport mode dot1q-tunnel
 l2protocol-tunnel cdp
 no cdp enable
 no shutdown
```

Task 1.3 Verification

When verifying, make sure that you give the neighbors time to update. If you configure and then immediately look, you may not see the desired result. Remember that CDP is sent out every 60 seconds by default, so it may take a minute for the new advertisement to be received. The old advertisement will time out, but it will take 3 minutes.

```
Rack1R1#show cdp neighbors fa0/0
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID     Local Intrfce    Holdtme   Capability  Platform  Port ID
Rack1SW2      Fas 0/0           125        S I       WS-C3560-2Fas 0/21

Rack1R3#show cdp neighbors fa0/0
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID      Local Intrfce   Holdtme   Capability  Platform  Port ID
Rack1SW2       Fas 0/0          155        S I       WS-C3560-2Fas 0/20
```



```
Rack1SW2#show cdp neighbors fa0/20
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone

Device ID   Local Intrfce    Holdtme  Capability    Platform   Port ID
Rack1R3        Fas 0/20       153          R S I     3640      Eth 0/0

Rack1SW2#show cdp neighbors fa0/21
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone

Device ID   Local Intrfce    Holdtme  Capability    Platform   Port ID
Rack1R1        Fas 0/21       129           R S      2610XM    Fas 0/0
```

☠ Pitfall

Changing the MTU on the edge switches may not be enough. If one of your other switches was being used as transit, you would need to adjust the MTU on that switch as well. Make sure to check your spanning tree root. Since we have a direct trunk between Switches 1 and 4, then no further configuration is necessary.

Task 1.4

SW1:

```
interface FastEthernet0/5
 storm-control unicast level 25.00
```

Task 1.4 Verification

```
Rack1SW1#show storm-control unicast
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Fa0/5      Forwarding      25.00%       25.00%        0.00%
Rack1SW1#
```

Task 1.5

SW2:

```
interface FastEthernet0/7
 switchport voice vlan 4
!
interface FastEthernet0/8
 switchport voice vlan 4
```


Task 1.5 Breakdown

Since ports on the 3560/3550 series switches default to dynamic mode, installing Cisco IP Phones into the network is a very straightforward process. When a phone is connected the switch will automatically form an 802.1q trunk to the phone. Traffic destined for the PC attached to the IP phone will be carried in the access VLAN. Voice traffic destined for the IP phone itself will be carried in the voice VLAN. These VLANs are defined with the **`switchport access vlan`** and **`switchport voice vlan`** command respectively.

For this task since the CallManager server will be located in VLAN 4 the VoIP traffic from the IP phone should also be placed in VLAN 4. Although technically the voice VLAN could be a different VLAN than the CallManager server is located in, the task asked for the minimal configuration to be used for this task. This ruled out other possible configurations.

☠ Pitfall

Unlike a data VLAN, a voice VLAN will not automatically be created when it is assigned. Be sure to create the voice VLAN in the VLAN database before assigning it.

Task 1.5 Verification

```
Rack1SW2#show interfaces fa0/7 switchport | include Voice
Voice VLAN: 4 (VLAN0004)
Rack1SW2#show interfaces fa0/8 switchport | include Voice
Voice VLAN: 4 (VLAN0004)
```

Task 1.6

SW2:

```
mls qos
!
interface FastEthernet0/7
 switchport access vlan 5
 switchport priority extend cos 1
 mls qos trust cos
!
interface FastEthernet0/8
 switchport access vlan 5
 switchport priority extend cos 1
 mls qos trust cos
```


Task 1.6 Breakdown

Since VoIP traffic requires special treatment throughout the network, a carefully designed end-to-end QoS policy is required in a network utilizing voice over data. In order to facilitate in creating this policy, QoS must extend down to the access layer. Traffic marking at the access layer is supported through layer 2 Class of Service (CoS) values. By default, the 3560/3550 does not process CoS values, and rewrites all frames with a CoS value of zero. To enable the processing of CoS, QoS must be enabled globally by issuing the **`mls qos`** global configuration command.

Once QoS is enabled, you must decide how the switch will process frames that already have a CoS value set. Typically, you would want to set the switch to trust the CoS value that is coming from the IP phone. This is accomplished by issuing the **`mls qos trust cos`** interface level command.

In order to prevent the device attached to the phone from getting better service throughout the network and interfering with VoIP traffic, the Cisco IP phone by default will re-tag all frames received from its extension with a CoS value of zero. To tag them with a different value or to leave them untagged, use the interface level command **`switchport priority extend [ trust | cos (value) ].`** In the above case, all traffic received from the PC attached to the IP phone is remarked with a CoS value of one.

Task 1.6 Verification

```
Rack1SW2#show interfaces fa0/7 switchport | include Access|Appliance
Access Mode VLAN: 5 (VLAN0005)
Appliance trust: 1

Rack1SW2#show interfaces fa0/8 switchport | include Access|Appliance
Access Mode VLAN: 5 (VLAN0005)
Appliance trust: 1

Rack1SW2#show mls qos interface fa0/7
FastEthernet0/7
trust state: trust cos
trust mode: trust cos
trust enabled flag: ena
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
Trust device: none
qos mode: port-based
```



```
Rack1SW2#show mls qos interface fa0/8
FastEthernet0/8
trust state: trust cos
trust mode: trust cos
trust enabled flag: ena
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
Trust device: none
qos mode: port-based
```

Task 1.7

SW1:

```
ip subnet-zero
interface Port-channel14
 no switchport
 ip address 163.1.0.1 255.255.255.128
!
interface FastEthernet0/20
 no switchport
 no ip address
 channel-group 14 mode desirable
!
interface FastEthernet0/21
 no switchport
 no ip address
 channel-group 14 mode desirable
!
interface range fa0/20 - 21
 no shutdown
```

SW4:

```
ip subnet-zero
!
interface Port-channel14
 no switchport
 ip address 163.1.0.4 255.255.255.128
!
interface FastEthernet0/14
 no switchport
 no ip address
 channel-group 14 mode desirable
!
interface FastEthernet0/15
 no switchport
 no ip address
 channel-group 14 mode desirable
!
interface range fa0/14 - 15
 no shutdown
```


SW3:
```
interface Port-channel34
 no switchport
 ip address 163.1.0.133 255.255.255.128
```

SW4:
```
interface Port-channel34
 no switchport
 ip address 163.1.0.134 255.255.255.128
```

SW3 and SW4:
```
interface range fa0/19 - 21
 no switchport
 no ip address
 channel-group 34 mode desirable
 no shutdown
```

Task 1.7 Verification

```
Rack1SW4#ping 163.1.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1SW4#ping 163.1.0.133

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.0.133, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1SW4#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: 2
Number of aggregators:           2

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------
14     Po14(RU)        PAgP      Fa0/14(P)   Fa0/15(P)
34     Po34(RU)        PAgP      Fa0/19(P)   Fa0/20(P)   Fa0/21(P)
```


Task 1.8

SW1 and SW2:

```
vtp mode transparent
!
!
vlan 500
  private-vlan isolated
!
vlan 42
  private-vlan primary
  private-vlan association 500
!
interface FastEthernet0/9
 switchport private-vlan host-association 42 500
 switchport mode private-vlan host
```

SW2:

```
interface FastEthernet0/4
 no switchport access vlan 42
 switchport private-vlan mapping 42 500
 switchport mode private-vlan promiscuous
!
interface FastEthernet0/24
 no switchport access vlan 42
 switchport private-vlan mapping 42 500
 switchport mode private-vlan promiscuous
```

> ☞ **Quick Note**
> Technically, the **switchport access vlan** command could have been left on but it will be not used when the interface is configured for private VLANs

Task 1.8 Verification

```
Rack1R4#ping 192.10.1.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.10.1.254, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms

Rack1SW2#show vlan private-vlan

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------
42      500       isolated          Fa0/4, Fa0/9, Fa0/24

Rack1SW1#show vlan private-vlan

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------
42      500       isolated          Fa0/9
```


Task 1.9

R3:

```
interface Serial1/0
 ip address 163.1.35.3 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 163.1.35.5 305 broadcast
 no frame-relay inverse-arp
```

R4:

```
interface Serial0/0
 ip address 163.1.54.4 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 163.1.54.5 405 broadcast
 no frame-relay inverse-arp
```

R5:

```
interface Serial0/0
 encapsulation frame-relay
!
interface Serial0/0.35 point-to-point
 ip address 163.1.35.5 255.255.255.0
 frame-relay interface-dlci 503
!
interface Serial0/0.54 point-to-point
 ip address 163.1.54.5 255.255.255.0
 frame-relay interface-dlci 504
```

Task 1.9 Breakdown

Although this task can be solved by using the solution above, it is important to remember that that there are four methods to deal with layer 3 to layer 2 mappings. The first and simplest is to use inverse-ARP. The second is to use the **`frame-relay map`** command. The third is to use point-to-point subinterfaces. Finally, the last option would be to use PPP over Frame Relay (PPPoFR). By using PPP over Frame Relay, you now are running IP over PPP over Frame Relay. As far as IP is concerned, there isn't any layer 3 to layer 2 mapping needed since it's now running over PPP.

Task 1.9 Verification

```
Rack1R5#show frame-relay map
Serial0/0.35 (up): point-to-point dlci, dlci 503(0x1F7,0x7C70),
broadcast
          status defined, active
Serial0/0.54 (up): point-to-point dlci, dlci 504(0x1F8,0x7C80),
broadcast
          status defined, active

Rack1R5#ping 163.1.35.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.35.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/60 ms

Rack1R5#ping 163.1.54.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.54.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/60/60 ms
```

Task 1.10

R4:

```
interface Serial0/0
 frame-relay interface-dlci 405
  class DLCI_405
!
map-class frame-relay DLCI_405
 frame-relay end-to-end keepalive mode reply
```

R5:

```
interface Serial0/0.54 point-to-point
 frame-relay interface-dlci 504
  class DLCI_504
!
map-class frame-relay DLCI_504
 frame-relay end-to-end keepalive mode request
```

Task 1.10 Breakdown

Since Frame Relay uses *virtual* circuits, a failure in the Frame Relay cloud may not be detected by all switches in the transit path. Therefore, it is the end node's (i.e. router's) responsibility to check the availability of the circuit by using Frame Relay end-to-end keepalives (EEK). To enable EEK, use the map-class subcommand **`frame-relay end-to-end keepalive mode [`*`mode`*`],`** where *mode* is request, reply, bidirectional, or passive-reply.

Task 1.10 Verification

```
Rack1R4#show frame-relay end-to-end keepalive

End-to-end Keepalive Statistics for Interface Serial0/0 (Frame Relay
DTE)

DLCI = 405, DLCI USAGE = LOCAL, VC STATUS = ACTIVE (EEK UP) ← status

RECEIVE SIDE STATISTICS

Send Sequence Number: 254,      Receive Sequence Number: 254
Configured Event Window: 3,     Configured Error Threshold: 2
Total Observed Events: 6,       Total Observed Errors: 3
Monitored Events: 1,            Monitored Errors: 0
Successive Successes: 1,        End-to-end VC Status: UP ← status
```

Task 1.11

R1:

```
interface Serial0/0
 ip address 163.1.12.1 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 163.1.12.2 102
 no frame-relay inverse-arp
```

R2:

```
interface Serial0/0
 ip address 163.1.12.2 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 163.1.12.1 201
 no frame-relay inverse-arp
```

Task 1.11 Verification

```
Rack1R1#show frame-relay map
Serial0/0 (up): ip 163.1.12.2 dlci 102(0x66,0x1860), static,
              CISCO, status defined, active

Rack1R1#ping 163.1.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```


Task 1.12

R6:

```
interface Virtual-Template1
 ip address 54.1.7.6 255.255.255.0
 ppp chap hostname ROUTER6
 ppp chap password 0 CISCO

interface Serial0/0
 encapsulation frame-relay
 frame-relay interface-dlci 201 ppp Virtual-Template1
 no frame-relay inverse-arp
```

Task 1.12 Breakdown

Since Frame Relay does not natively support features such as authentication, link quality monitoring, and reliable transmission, it is sometimes advantageous to run PPP over Frame Relay in order to enable these features.

Pitfall

When using a virtual-template interface, it's important to understand that a virtual-access interface is cloned from the virtual-template interface when the PPP connection comes up. The virtual-template interface itself will always be in the down/down state.

```
Rack1R6#show ip interface brief | include 54.1.7.6
Virtual-Access1      54.1.7.6     YES TFTP     up      up
Virtual-Template1    54.1.7.6     YES manual   down    down
Rack1R6#
```


Task 1.12 Verification

Verify that PPPoFR link is authenticated:

```
Rack1R6#debug ppp authentication
Rack1R6#
 Vi1 PPP: Using default call direction
 Vi1 PPP: Treating connection as a dedicated line
 Vi1 PPP: Session handle[7C000008] Session id[3]
 Vi1 PPP: Authorization required
 Vi1 PPP: No authorization without authentication
 Vi1 CHAP: I CHALLENGE id 36 len 24 from "BB1"
 Vi1 CHAP: Using hostname from interface CHAP
 Vi1 CHAP: Using password from interface CHAP
 Vi1 CHAP: O RESPONSE id 36 len 28 from "ROUTER6"
 Vi1 CHAP: I SUCCESS id 36 len 4
```

2. IP IGP Routing

Task 2.1

R1:

```
router rip
 version 2
 passive-interface default
 no passive-interface FastEthernet0/0
 network 163.1.0.0
 no auto-summary
```

R2:

```
router rip
 version 2
 network 204.12.1.0
 no auto-summary
```

R3:

```
router rip
 version 2
 passive-interface default
 no passive-interface FastEthernet0/0
 network 163.1.0.0
 no auto-summary
```


R4:
```
router rip
 version 2
 passive-interface default
 no passive-interface Serial0/1
 network 163.1.0.0
 no auto-summary
```

R5:
```
router rip
 version 2
 passive-interface default
 no passive-interface Serial0/1
 network 163.1.0.0
 no auto-summary
```

R6:
```
router rip
 version 2
 network 54.0.0.0
 network 150.1.0.0
 network 163.1.0.0
 network 204.12.1.0
 no auto-summary
```

SW2:
```
ip routing
!
router rip
 version 2
 network 150.1.0.0
 network 163.1.0.0
 no auto-summary
```


Task 2.1 Verification

```
Rack1R1#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 26 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/0       2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    163.1.0.0
  Passive Interface(s):
    VoIP-Null0
    Serial0/0
    Serial0/1
    Virtual-Access1
    Loopback0
  Routing Information Sources:
    Gateway         Distance      Last Update
    163.1.18.8            120      00:00:04
  Distance: (default is 120)
```

Verify routes received via RIP (note SW2 Loopback0 prefix):

```
Rack1R1#show ip route rip
     163.1.0.0/16 is variably subnetted, 6 subnets, 2 masks
R       163.1.35.0/24 [120/2] via 163.1.18.8, 00:00:13, FastEthernet0/0
R       163.1.38.0/24 [120/1] via 163.1.18.8, 00:00:13, FastEthernet0/0
     150.1.0.0/24 is subnetted, 2 subnets
R       150.1.8.0 [120/1] via 163.1.18.8, 00:00:13, FastEthernet0/0
```

Task 2.2

R2:

```
router rip
 distribute-list gateway R6 in
!
ip prefix-list R6 seq 5 permit 204.12.1.6/32
```

R6:

```
router rip
 distribute-list gateway R2 in
 distribute-list prefix RIP out FastEthernet0/1
!
ip prefix-list R2 seq 5 permit 204.12.1.2/32
!
ip prefix-list RIP seq 5 permit 163.1.6.0/24
ip prefix-list RIP seq 10 permit 150.1.6.0/24
```


Task 2.2 Breakdown

An alternate application of the IP prefix-list is in the **distribute-list gateway** statement. This allows prefixes to be filtered as they are received based on the source of the update. In the above task, this syntax is used on both R2 and R6 to only accept RIP updates from each other. This allows updates learned from both BB1 and BB3 to be denied, but still allows updates to be received from R2 and R6 respectively.

Task 2.2 Verification

```
Verify that R2 receives prefixes for R6 Loopback0 and Gig0/0
interfaces:

Rack1R2#show ip route rip
     163.1.0.0/24 is subnetted, 2 subnets
R       163.1.6.0 [120/1] via 204.12.1.6, 00:00:01, FastEthernet0/0
     150.1.0.0/24 is subnetted, 2 subnets
R       150.1.6.0 [120/1] via 204.12.1.6, 00:00:01, FastEthernet0/0

Verify that R6 does not receive any prefix from the backbone routers:

Rack1R6#show ip route rip

Rack1R6#
```

Task 2.3

R5:

```
router rip
 no passive-interface FastEthernet0/1
 default-information originate
 distribute-list prefix DEFAULT out FastEthernet0/1
 no auto-summary
!
ip prefix-list DEFAULT seq 5 permit 0.0.0.0/0
```

SW1:

```
ip routing
!
router rip
 version 2
 network 150.1.0.0
 network 163.1.0.0
 no auto-summary
```


Task 2.3 Breakdown

To advertise a default route into RIP, simply issue the **`default-information originate`** routing process subcommand. In the above case, a prefix-list matching a default route is used to filter R5's advertisement to SW1. By only permitting 0.0.0.0/0, SW1 only has default reachability to the rest of the IGP domain.

Task 2.3 Verification

Verify that R5 only sends the default route to SW1:

```
Rack1R5#debug ip rip

 RIP: received v2 update from 163.1.57.7 on FastEthernet0/1
      150.1.7.0/24 via 0.0.0.0 in 1 hops
      163.1.7.0/24 via 0.0.0.0 in 1 hops

 RIP: sending v2 update to 224.0.0.9 via FastEthernet0/1 (163.1.57.5)
 RIP: build update entries
   0.0.0.0/0 via 0.0.0.0, metric 1, tag 0
```

Task 2.4

R4:

```
key chain RIP
 key 1
  key-string CISCO
!
interface FastEthernet0/0
 ip rip authentication mode md5
 ip rip authentication key-chain RIP
 ip summary-address rip 163.1.0.0 255.255.192.0
 ip summary-address rip 150.1.0.0 255.255.240.0
!
router rip
 version 2
 network 192.10.1.0
 no passive-interface FastEthernet0/0
 distribute-list prefix RIP_SUMMARY out FastEthernet0/0
 distribute-list 1 in FastEthernet0/0
 no auto-summary
!
ip prefix-list RIP_SUMMARY seq 5 permit 163.1.0.0/18
ip prefix-list RIP_SUMMARY seq 10 permit 150.1.0.0/20
!
access-list 1 deny any
```


Task 2.4 Breakdown

The first step in properly summarizing the internal address space is to list all known addresses sequentially. The addresses used in this network are as follows:

```
163.1.4.0/24
163.1.5.0/24
163.1.6.0/24
163.1.7.0/24
163.1.12.0/24
163.1.13.0/24
163.1.18.0/24
163.1.35.0/24
163.1.38.0/24
163.1.45.0/24
163.1.54.0/24
163.1.57.0/24
```

From this list, it is evident that the first two octets are the same. Therefore, the minimum summary that will encompass all of this address space is 163.1.0.0/16. To determine what the maximum summarization is that will encompass all of the above address space, next write out all addresses in the third octet in binary:

	128	64	32	16	8	4	2	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
12	0	0	0	0	1	1	0	0
13	0	0	0	0	1	1	0	1
18	0	0	0	1	0	0	1	0
35	0	0	1	0	0	0	1	1
38	0	0	1	0	0	1	1	0
45	0	0	1	0	1	1	0	1
54	0	0	1	1	0	1	1	0
57	0	0	1	1	0	1	0	1

Next, count how many bit positions are consistent. From the above output, it is evident that two places, the 128 and 64 bits, are consistent. Add these two bits onto the previous summarization of /16, and our resulting summary is /18. Therefore, the final summary for this task is 163.1.0.0/16

Task 2.4 Verification

Verify that R4 only sends the summary prefixes to BB2:

```
Rack1R4#debug ip rip

*Apr 23 00:19:50.939: RIP: sending v2 update to 224.0.0.9 via
FastEthernet0/0 (192.10.1.4)
*Apr 23 00:19:50.939: RIP: build update entries
*Apr 23 00:19:50.939:   150.1.0.0/20 via 0.0.0.0, metric 3, tag 0
*Apr 23 00:19:50.939:   163.1.0.0/18 via 0.0.0.0, metric 1, tag 0
```

Verify that we do not receive any routing information from BB2 via RIP:

```
Rack1R4#show ip route rip | include via 192.10.2.254
Rack1R4#
```

Task 2.5

R3:

```
interface Serial1/0
 ip ospf network point-to-point
!
router ospf 1
 router-id 150.1.3.3
 network 163.1.35.3 0.0.0.0 area 1
```

R4:

```
interface Serial0/0
 ip ospf network point-to-point
!
router ospf 1
 router-id 150.1.4.4
 network 163.1.54.4 0.0.0.0 area 0
```

R5:

```
router ospf 1
 router-id 150.1.5.5
 network 163.1.35.5 0.0.0.0 area 1
 network 163.1.54.5 0.0.0.0 area 0
```


Task 2.5 Verification

```
Rack1R5#show ip ospf neighbor

Neighbor ID     Pri   State      Dead Time   Address         Interface
150.1.4.4         0  FULL/  -   00:00:35    163.1.54.4      Serial0/0.54
150.1.3.3         0  FULL/  -   00:00:39    163.1.35.3      Serial0/0.35
```

Verify network type on Serial1/0:

```
Rack1R3#show ip ospf interface s1/0
Serial1/0 is up, line protocol is up
  Internet Address 163.1.35.3/24, Area 1
  Process ID 1, Router ID 150.1.3.3, Network Type POINT->POINT, Cost:
781
  Transmit Delay is 1 sec, State POINT->POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

Task 2.6

R4:

```
router ospf 1
 network 150.1.4.4 0.0.0.0 area 0
 network 163.1.4.4 0.0.0.0 area 0
```

R5:

```
router ospf 1
 area 0 range 150.1.4.0 255.255.254.0
 network 150.1.5.5 0.0.0.0 area 0
 network 163.1.5.5 0.0.0.0 area 0
```

Task 2.6 Verification

Verify that the summary LSA is generated:

```
Rack1R5#show ip ospf database summary 150.1.4.0

            OSPF Router with ID (150.1.5.5) (Process ID 1)

                Summary Net Link States (Area 1)

  LS age: 50
  Options: (No TOS-capability, DC, Upward)
  LS Type: Summary Links(Network)
  Link State ID: 150.1.4.0 (summary Network Number)
  Advertising Router: 150.1.5.5
  LS Seq Number: 80000002
  Checksum: 0x1AE4
  Length: 28
  Network Mask: /23
        TOS: 0  Metric: 1
```


Verify the route on R3:

```
Rack1R3#show ip route 150.1.4.0
Routing entry for 150.1.4.0/23
  Known via "ospf 1", distance 110, metric 782, type inter area
  Last update from 163.1.35.5 on Serial1/0, 00:01:48 ago
  Routing Descriptor Blocks:
  * 163.1.35.5, from 150.1.5.5, 00:01:48 ago, via Serial1/0
      Route metric is 782, traffic share count is 1
```

Task 2.7

R1:

```
interface Tunnel0
 ip address 163.1.15.1 255.255.255.0
 tunnel source Loopback0
 tunnel destination 163.1.35.5
!
interface Serial0/1
 ip ospf network non-broadcast
!
router ospf 1
 router-id 150.1.1.1
 area 0 range 150.1.4.0 255.255.254.0
 network 163.1.12.1 0.0.0.0 area 2
 network 163.1.13.1 0.0.0.0 area 1
 network 163.1.15.1 0.0.0.0 area 0
 neighbor 163.1.13.3
 neighbor 163.1.12.2
```

Quick Note
The "Do's and Don'ts" section for this lab did not specify that additional IP addressing can not be used

R2:

```
interface Serial0/0
 ip ospf priority 0
!
router ospf 1
 router-id 150.1.2.2
 network 163.1.12.2 0.0.0.0 area 2
```

R3:

```
interface Serial1/2
 ip ospf network non-broadcast
 ip ospf priority 0
!
router ospf 1
 router-id 150.1.3.3
 network 163.1.13.3 0.0.0.0 area 1
```

R5:

```
interface Tunnel0
 ip address 163.1.15.5 255.255.255.0
 tunnel source Serial0/0.35
 tunnel destination 150.1.1.1
!
router ospf 1
 network 163.1.15.5 0.0.0.0 area 0
```


Task 2.7 Breakdown

This solution uses a tunnel as opposed to a virtual-link, due to the fact that once a virtual-link is brought up between R1 and R5, R1 would then "leak" the 150.1.4.4/32 and 150.1.5.5/32 routes to R3 that R5 is summarizing. This will break Task 2.6.

By using a tunnel and also summarizing the loopbacks on R1, it will enable R3 to only receive the /23 summary and not the specifics. Note: Without reachability to R1's loopback, the tunnel will not come up. Since the loopback isn't added to OSPF until the next section, you will need to complete that section before verifying the tunnel.

Task 2.7 Verification

Check that tunnel is working:

```
Rack1R1#ping 163.1.15.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.15.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5),round-trip min/avg/max=104/106/108 ms
```

Verify the OSPF neighbors. Verify the neighbors' state to be sure that R1 is the DR.

```
Rack1R1#show ip ospf neighbor

Neighbor ID     Pri State          Dead Time   Address         Interface
150.1.5.5         0 FULL/  -       00:00:34    163.1.15.5      Tunnel0
150.1.3.3         0 FULL/DROTHER  00:01:32    163.1.13.3      Serial0/1
150.1.2.2         0 FULL/DROTHER  00:01:39    163.1.12.2      Serial0/0
```

Verify the network types on R1 interfaces:

```
Rack1R1#show ip ospf interface s0/0
Serial0/0 is up, line protocol is up
  Internet Address 163.1.12.1/24, Area 2
  Process ID 1, Router ID 150.1.1.1,Network Type NON_BROADCAST,Cost: 64
<output omitted>

Rack1R1#show ip ospf interface s0/1
Serial0/1 is up, line protocol is up
  Internet Address 163.1.13.1/24, Area 1
  Process ID 1, Router ID 150.1.1.1,Network Type NON_BROADCAST,Cost: 64
```


Verify OSPF routes on R1 to ensure we receive prefixes from R4 and R5:

```
Rack1R1#show ip route ospf
     163.1.0.0/16 is variably subnetted, 8 subnets, 2 masks
O       163.1.35.0/24 [110/845] via 163.1.13.3, 00:05:11, Serial0/1
O       163.1.54.0/24 [110/11175] via 163.1.15.5, 00:03:31, Tunnel0
     150.1.0.0/16 is variably subnetted, 5 subnets, 3 masks
O       150.1.4.0/23 is a summary, 00:03:31, Null0
O       150.1.5.5/32 [110/11112] via 163.1.15.5, 00:03:31, Tunnel0
O       150.1.4.4/32 [110/11176] via 163.1.15.5, 00:03:31, Tunnel0
```

Verify that R3 still only has the summary prefix 150.1.4.0/23:

```
Rack1R3#show ip route 150.1.4.0
Routing entry for 150.1.4.0/23
  Known via "ospf 1", distance 110, metric 782, type inter area
  Last update from 163.1.35.5 on Serial1/0, 00:04:22 ago
  Routing Descriptor Blocks:
  * 163.1.35.5, from 150.1.5.5, 00:04:22 ago, via Serial1/0
      Route metric is 782, traffic share count is 1
```

Task 2.8

R1, R2 and R3:

```
router ospf 1
 redistribute connected subnets route-map CONNECTED->OSPF
!
route-map CONNECTED->OSPF permit 10
 match interface Loopback0
```

Task 2.8 Breakdown

Although a route-map permitting only the loopback interface to be redistributed into OSPF isn't explicitly asked for in this task, it is good practice to only redistribute the interface the task is asking for.

Technically, putting the loopback interface into another routing protocol and then redistributing that protocol into OSPF would also work, redistributing the connected interface into OSPF directly is a better solution.

Task 2.8 Verification

Verify prefixes length and OSPF route-types:

```
Rack1R5#show ip route ospf  | include E2
O E2    150.1.3.0/24 [110/20] via 163.1.35.3, 00:01:34, Serial0/0.35
O E2    150.1.2.0/24 [110/20] via 163.1.15.1, 00:01:34, Tunnel0
O E2    150.1.1.0/24 [110/20] via 163.1.35.3, 00:01:34, Serial0/0.35
```


Task 2.9

SW1:

```
router ospf 1
 router-id 150.1.7.7
 network 163.1.0.1 0.0.0.0 area 0
```

SW3:

```
ip routing
!
router ospf 1
 router-id 10.3.3.3
 network 163.1.3.3 0.0.0.0 area 0
 network 163.1.0.133 0.0.0.0 area 0
```

SW4:

```
ip routing
!
router ospf 1
 router-id 10.4.4.4
 network 163.1.0.4 0.0.0.0 area 0
 network 163.1.0.134 0.0.0.0 area 0

int po34
 ip mtu 1500
```

Task 2.9 Breakdown

For the peering between SW3 and SW4, you may run into issues due to an MTU mismatch. The MTU was changed earlier for the tunneling section. If the MTU doesn't match, it can prevent an adjacency from forming. The side with the smaller MTU will give the error message shown below in the output of debug ip ospf adjacency:

"OSPF: Nbr 10.4.4.4 has larger interface MTU"

There are a couple options. If SW4 is a 3550, you can use the command ip mtu on the port channel interface. If SW4 is a 3560, you have the option to adjust the system routing MTU to 1500. You can also configure the command ip ospf mtu-ignore on the port channel interface.

Task 2.9 Verification

```
Rack1SW4#show ip ospf neighbor

Neighbor ID Pri  State       Dead Time   Address         Interface
10.3.3.3     1   FULL/BDR    00:00:32    163.1.0.133     Port-channel34
150.1.7.7    1   FULL/DR     00:00:38    163.1.0.1       Port-channel14

Rack1SW1#show ip route ospf
     163.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
O       163.1.0.128/25 [110/2] via 163.1.0.4, 00:00:56, Port-channel14
O       163.1.3.0/24 [110/3] via 163.1.0.4, 00:00:56, Port-channel14

Rack1SW3#show ip route ospf
     163.1.0.0/16 is variably subnetted, 3 subnets, 2 masks
O       163.1.0.0/25 [110/2] via 163.1.0.134, 00:01:10, Port-channel34
```

Task 2.10

SW1:

```
router ospf 1
 network 163.1.0.1 0.0.0.0 area 0
 distribute-list 1 in
!
access-list 1 deny   10.3.3.3
access-list 1 permit any
```

☞ Quick Note
Deny SW3's Loopback prefix from being installed in SW1's routing table

SW3:

```
router ospf 1
 network 10.3.3.3 0.0.0.0 area 34
```

☞ Quick Note
Arbitrary area

SW4:

```
router ospf 1
 area 34 range 10.0.0.0 255.0.0.0
 network 10.4.4.4 0.0.0.0 area 34
```

Task 2.10 Breakdown

Start by adding the two loopback networks to OSPF. After adding the loopbacks to OSPF, initially you should see both routes on SW1. In order to see a 10.0.0.0/8 route, a summary will need to be configured. The summary could be configured on either SW3 or SW4 or both. Here, we are showing the summary configured on SW4. If the summary is just configured on one of the two switches, then some sort of filtering is necessary so that the other network does not show up on SW1. Reading the question carefully will show that the network should not be in the routing table, not that it could not be in the OSPF database on R1. With OSPF, a distribute list inbound can be used to filter what shows up in the routing table.

When configuring a summary, a route to null0 is added to the routing table so that traffic arriving for the summary where there is not a more particular route in the routing table can be discarded. If you did happen to configure a summary on both SW3 and SW4, this discard route could cause connectivity when trying to ping the loopbacks of SW3 and SW4. Having the desired result of only seeing the summary is not going to gain points if you lose connectivity in the process. The discard route can be removed with the no discard-route command under the OSPF process.

Task 2.10 Verification

Verify that the routing table shows up as listed in the lab section, and that you can ping the two loopbacks.

```
Rack1SW1#show ip route ospf
     163.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
O       163.1.0.128/25 [110/2] via 163.1.0.4, 00:26:59, Port-channel14
O       163.1.3.0/24 [110/3] via 163.1.0.4, 00:26:59, Port-channel14
O IA 10.0.0.0/8 [110/2] via 163.1.0.4, 00:26:59, Port-channel14
Rack1SW1#

Rack1SW1#ping 10.3.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/9 ms
Rack1SW1#ping 10.4.4.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.4.4.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms
Rack1SW1#
```


Task 2.11

R1:

```
router ospf 1
 redistribute rip subnets route-map RIP->OSPF
!
router rip
 redistribute connected metric 1
 redistribute ospf 1 metric 1 route-map OSPF->RIP
!
ip prefix-list SUB24_OR_SHORTER seq 5 permit 0.0.0.0/0 le 24
!
route-map OSPF->RIP deny 10
 match tag 120
!
route-map OSPF->RIP permit 20
 match ip address prefix-list SUB24_OR_SHORTER
 set tag 110
!
route-map RIP->OSPF deny 10
 match tag 110
!
route-map RIP->OSPF permit 20
 set tag 120
```

R2:

```
router ospf 1
 redistribute rip subnets
!
router rip
 redistribute connected
 redistribute ospf 1 metric 1

route-map CONNECTED->OSPF permit 20
 match interface FastEthernet0/0
```

R3:

```
router ospf 1
 redistribute rip subnets route-map RIP->OSPF
!
router rip
 redistribute connected metric 1
 redistribute ospf 1 metric 1 route-map OSPF->RIP
!
route-map OSPF->RIP deny 10
 match tag 120
!
route-map OSPF->RIP permit 20
 set tag 110
!
route-map RIP->OSPF deny 10
 match tag 110
!
route-map RIP->OSPF permit 20
 set tag 120
```


R4:

```
router ospf 1
 redistribute rip subnets
!
router rip
 redistribute ospf 1 metric 1
 distance 109 163.1.45.5 0.0.0.0 RIP_ROUTES
!
ip access-list standard RIP_ROUTES
 permit 150.1.7.0
 permit 163.1.57.0
 permit 163.1.7.0
```

R5:

```
router rip
 distance 109 0.0.0.0 255.255.255.255 RIP_ROUTES
!
ip access-list standard RIP_ROUTES
 permit 150.1.7.0
 permit 163.1.7.0
 permit 163.1.4.0
 permit 192.10.1.0
 permit 10.0.0.0
 permit 163.1.0.0
 permit 163.1.0.128
 permit 163.1.3.0
```

SW1:

```
router ospf 1
 redistribute rip subnets
 default-information originate
!
router rip
 redistribute ospf 1 metric 1
!
```

SW2:

```
router rip
 offset-list 1 in 2 FastEthernet0/20
 offset-list 0 in 1 FastEthernet0/21
!
access-list 1 permit 150.1.6.0
access-list 1 permit 150.1.2.0
access-list 1 permit 150.1.1.0
access-list 1 permit 163.1.6.0
access-list 1 permit 163.1.12.0
access-list 1 permit 204.12.1.0
```


Task 2.11 Breakdown

Previous routing exercises have focused on path selection between two or more different protocols. The above exercise focuses on route selection within a single protocol. This path selection is accomplished by altering routing metrics. The above requirement states that SW2 should use the optimal IGP path for all routes throughout the network.

SW2 has two connections to the rest of the routing domain, one through R1 and the other through R3. Behind R1 exists R2 and R6. All other networks will be optimally reached through R3. Since there are many more routes learned from behind R3 than from behind R1, the easiest way to achieve optimal routing is to match the routes learned from behind R1 and prefer the metric for these through R1. This is accomplished in the above solutions by creating a standard access-list which matches these prefixes. Next, the metric of these routes is offset by 2 when they are learned from R3. Next, all other routes learned from R1 are offset by 1. Therefore SW2 will prefer the routes accordingly.

In the earlier IGP section, we were told that SW1 should only receive a default from R5. In order for SW3 and SW4 to have reachability to the rest of the topology, the default route can be passed on to SW3 and SW4 by passing it on to OSPF with the default information originate command.

Also, we were told that R3 should see the loopbacks for R4 and R5 as a summary /23 in section 2.6. Due to the tunnel, R1 will still see the more specific routes to the loopback networks. In order to prevent leaking these into RIP, the redistribution is also filtering networks with a mask longer than 24 bits. Alternatively, you could just filter the specific loopbacks from the redistribution from OSPF into RIP.

Task 2.11 Verification

V*erify connectivity for internal network with following TCL script (run it on routers R1 through R6):*

```
foreach i {
150.1.1.1
163.1.15.1
163.1.13.1
163.1.12.1
163.1.18.1
150.1.2.2
163.1.12.2
204.12.1.2
163.1.35.3
163.1.38.3
150.1.3.3
163.1.13.3
163.1.45.4
163.1.54.4
150.1.4.4
```



```
163.1.4.4
192.10.1.4
163.1.35.5
163.1.45.5
163.1.54.5
150.1.5.5
163.1.57.5
163.1.5.5
163.1.15.5
150.1.6.6
163.1.6.6
204.12.1.6
150.1.7.7
163.1.57.7
163.1.7.7
163.1.0.1
163.1.38.8
150.1.8.8
163.1.18.8
163.1.3.3
163.1.0.133
10.3.3.3
163.1.0.4
163.1.0.134
10.4.4.4
} { ping $i }
```

Next verify best path selection on SW2.

```
Rack1SW2#show ip route | include FastEthernet0/21
R    204.12.1.0/24 [120/2] via 163.1.18.1, 00:00:02, FastEthernet0/21
R       163.1.6.0/24 [120/2] via 163.1.18.1, 00:00:02, FastEthernet0/21
R       163.1.12.0/24 [120/2] via 163.1.18.1,00:00:02, FastEthernet0/21
C       163.1.18.0/24 is directly connected, FastEthernet0/21
R       150.1.6.0/24 [120/2] via 163.1.18.1, 00:00:02, FastEthernet0/21
R       150.1.2.0/24 [120/2] via 163.1.18.1, 00:00:02, FastEthernet0/21
R       150.1.1.0/24 [120/2] via 163.1.18.1, 00:00:02, FastEthernet0/21
```

Task 2.12

R3:

```
ipv6 unicast-routing
!
interface FastEthernet0/0
 ipv6 address FEC0:CC1E:1:38::/64 eui-64
!
interface Serial1/0
 ipv6 address FEC0:CC1E:1:35::3/64
 ipv6 address fe80::3 link-local
 frame-relay map ipv6 FEC0:CC1E:1:35::5 305 broadcast
```



```
R4:
ipv6 unicast-routing
!
interface FastEthernet0/1
 ipv6 address FEC0:CC1E:1:4::/64 eui-64
!
interface Serial0/0
 ipv6 address FEC0:CC1E:1:54::4/64
 ipv6 address fe80::4 link-local
 frame-relay map ipv6 FEC0:CC1E:1:54::5 405 broadcast
!
interface Serial0/1
 ipv6 address FEC0:CC1E:1:45::4/64

R5:
ipv6 unicast-routing
!
interface Serial0/1
 ipv6 address FEC0:CC1E:1:45::5/64
!
interface Serial0/0.54 point-to-point
 ipv6 address FEC0:CC1E:1:54::5/64
 ipv6 address fe80::5 link-local
!
interface Serial0/0.35 point-to-point
 ipv6 address FEC0:CC1E:1:35::5/64
 ipv6 address fe80::5 link-local
```

Task 2.12 Verification

Verify layer 3 to layer 2 mapping on the Frame-Relay interfaces:

```
Rack1R4#show frame-relay map
Serial0/0 (up): ipv6 FEC0:CC1E:1:54::5 dlci 405(0x195,0x6450), static,
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ipv6 FE80::5 dlci 405(0x195,0x6450), static,
              broadcast,
              CISCO, status defined, active
Serial0/0 (up): ip 163.1.54.5 dlci 405(0x195,0x6450), static,
              broadcast,
              CISCO, status defined, active
```

Verify connectivity:

```
Rack1R4#ping FEC0:CC1E:1:54::5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0:CC1E:1:54::5, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/59/60 ms
```


Task 2.13

R3:

```
interface FastEthernet0/0
 ipv6 rip RIPng enable
!
interface Serial1/0
 ipv6 rip RIPng enable
 frame-relay map ipv6 FE80::5 305
!
ipv6 router rip RIPng
```

R4:

```
interface Loopback100
 ipv6 address 2001:220:20:3::1/64
 ipv6 rip RIPng enable
!
interface Loopback101
 ipv6 address 2001:222:22:2::1/64
 ipv6 rip RIPng enable
!
interface Loopback103
 ipv6 address 2001:205:90:31::1/64
 ipv6 rip RIPng enable
!
interface FastEthernet0/0
 ipv6 rip RIPng enable
!
interface Serial0/0
 ipv6 rip RIPng enable
 frame-relay map ipv6 FE80::5 405
!
interface FastEthernet0/1
 ipv6 rip RIPng enable
!
interface Serial0/1
 ipv6 rip RIPng enable
!
ipv6 router rip RIPng
```

R5:

```
interface Serial0/0.35 point-to-point
 ipv6 rip RIPng enable
!
interface Serial0/0.54 point-to-point
 ipv6 rip RIPng enable
 ipv6 rip RIPng metric-offset 2
!
interface Serial0/1
 ipv6 rip RIPng enable
```


Task 2.13 Breakdown

Adding the loopback networks and adding the interfaces should be very straightforward. For the path preference, the metric offset can be used. Alternatively, policy routing could be used but is prohibited by the initial lab requirements.

Task 2.13 Verification

Verify the RIPng configuration:

```
Rack1R3#show ipv6 rip
RIP process "RIPng", port 521, multicast-group FF02::9, pid 194
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 4, trigger updates 4
  Interfaces:
    Serial1/0
    FastEthernet0/0
  Redistribution:
    None
```

Verify the IPv6 RIPng routes:

```
Rack1R3#show ipv6 route rip
IPv6 Routing Table - 13 entries
<snip>
R   2001:205:90:31::/64 [120/4]
     via FE80::5, Serial1/0
R   2001:220:20:3::/64 [120/4]
     via FE80::5, Serial1/0
R   2001:222:22:2::/64 [120/4]
     via FE80::5, Serial1/0
R   FEC0:CC1E:1:4::/64 [120/3]
     via FE80::5, Serial1/0
R   FEC0:CC1E:1:45::/64 [120/2]
     via FE80::5, Serial1/0
R   FEC0:CC1E:1:54::/64 [120/2]
     via FE80::5, Serial1/0
```

Verify that packets to the selected IPv6 prefixes flow via the R4-R5 serial link:

```
Rack1R3#traceroute 2001:222:22:2::1

Type escape sequence to abort.
Tracing the route to 2001:222:22:2::1

  1 FEC0:CC1E:1:35::5 44 msec 56 msec 48 msec
  2 FEC0:CC1E:1:45::4 68 msec 80 msec 68 msec
```


Task 2.14

R4:

```
interface Serial0/0
 ipv6 traffic-filter DENY_FROM_38 in
!
interface Serial0/1
 ipv6 traffic-filter DENY_FROM_38 in

!
ipv6 access-list DENY_FROM_38
 deny ipv6 FEC0:CC1E:1:38::/64 any
 permit ipv6 any any
```

Task 2.14 Breakdown

Access-list filtering in IPv6 is configured with the **`ipv6 access-list [name]`** command, where *name* is the access-list identifier. Note that there is no separation between standard access-lists and extended access-lists in IPv6. To apply the access-list to the interface, the more intuitive **`ipv6 traffic-filter`** command is used. Keep in mind that just like IPv4 access-lists, IPv6 access-lists end in implicit deny.

Task 2.14 Verification

To verify the IPv6 traffic filter, ping VLAN4 using the source address of R3's FastEthernet interface.

First, determine VLAN4 interface IPv6 address:

```
Rack1R4#show ipv6 interface Fa0/1
FastEthernet0/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::230:94FF:FE7E:E582
  Global unicast address(es):
    FEC0:CC1E:1:4:230:94FF:FE7E:E582, subnet is FEC0:CC1E:1:4::/64
[EUI]
<output omitted>
```

Now Ping:

```
Rack1R3#ping FEC0:CC1E:1:4:230:94FF:FE7E:E582 source Fa0/0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0:CC1E:1:4:230:94FF:FE7E:E582,
timeout is 2 seconds:
Packet sent with a source address of FEC0:CC1E:1:38:250:73FF:FE1C:7761
AAAAA
Success rate is 0 percent (0/5)
```


Ping using another source IPv6 address:

```
Rack1R3#ping FEC0:CC1E:1:4:230:94FF:FE7E:E582

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0:CC1E:1:4:230:94FF:FE7E:E582,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 88/96/104
ms
```

Verify the access-list hits on R4:

```
Rack1R4#show ipv6 access-list
IPv6 access list DENY_FROM_VLAN38
    deny ipv6 FEC0:CC1E:1:38::/64 any (5 matches) sequence 10
    permit ipv6 any any (38 matches) sequence 20
```

3. BGP

Task 3.1

R1:

```
router bgp 100
 bgp router-id 150.1.1.1
 neighbor 163.1.12.2 remote-as 100
 neighbor 163.1.13.3 remote-as 300
 neighbor 163.1.18.8 remote-as 300
```

R2:

```
router bgp 100
 bgp router-id 150.1.2.2
 neighbor 163.1.12.1 remote-as 100
 neighbor 163.1.12.1 route-reflector-client
 neighbor 204.12.1.6 remote-as 100
 neighbor 204.12.1.6 route-reflector-client
 neighbor 204.12.1.254 remote-as 54
```

R3:

```
router bgp 300
 bgp router-id 150.1.3.3
 neighbor 163.1.13.1 remote-as 100
 neighbor 163.1.35.5 remote-as 200
 neighbor 163.1.38.8 remote-as 300
 neighbor 163.1.38.8 send-community
```

R6:

```
router bgp 100
 bgp router-id 150.1.6.6
 neighbor 54.1.7.254 remote-as 54
 neighbor 204.12.1.2 remote-as 100
 neighbor 204.12.1.254 remote-as 54
```


SW2:
```
router bgp 300
 bgp router-id 150.1.8.8
 neighbor 163.1.18.1 remote-as 100
 neighbor 163.1.38.3 remote-as 300
 neighbor 163.1.38.3 send-community
```

Task 3.1 Verification

Verify BGP peering configuration, for instance on R1:

```
Rack1R1#show ip bgp summary | begin Neighbor
Neighbor    V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down  State/PfxRcd
163.1.12.2  4 100  21      17       11    0    0 00:13:39       10
163.1.13.3  4 300  17      18       11    0    0 00:12:38        0
163.1.18.8  4 300   4       6       11    0    0 00:00:10        0
```

Verify that R3 is sending communities to SW2, and vice versa:

```
Rack1R3#show ip bgp neighbors 163.1.38.8 | include Comm
  Community attribute sent to this neighbor

Rack1SW2#show ip bgp neighbors 163.1.38.3 | include Comm
  Community attribute sent to this neighbor
```

Task 3.2

R4:
```
router bgp 65004
 bgp router-id 150.1.4.4
 bgp confederation identifier 200
 bgp confederation peers 65005
 neighbor 150.1.5.5 remote-as 65005
 neighbor 150.1.5.5 ebgp-multihop
 neighbor 150.1.5.5 update-source Loopback0
 neighbor 192.10.1.254 remote-as 254
 neighbor 192.10.1.254 password CISCO
```

R5:
```
router bgp 65005
 bgp router-id 150.1.5.5
 bgp confederation identifier 200
 bgp confederation peers 65004 65007
 neighbor 150.1.4.4 remote-as 65004
 neighbor 150.1.4.4 ebgp-multihop
 neighbor 150.1.4.4 update-source Loopback0
 neighbor 163.1.35.3 remote-as 300
 neighbor 163.1.57.7 remote-as 65007
```

SW1:
```
router bgp 65007
 bgp router-id 150.1.7.7
 bgp confederation identifier 200
 bgp confederation peers 65005
 neighbor 163.1.57.5 remote-as 65005
```


Task 3.2 Breakdown

BGP confederation is used in large scale BGP implementations in order to reduce the amount of iBGP peering sessions required to transport Network Layer Reachability Information (NLRI) throughout the AS. By subdividing the autonomous system into smaller sub-autonomous systems, the amount of iBGP peering sessions can be dramatically reduced.

The first step in defining a confederation is to enable the BGP process by issuing the **router bgp [*sub-ASN*]** command. Note that the AS number specified when the process is initiated is the sub-AS number.

Next, the **bgp confederation identifier [*public-AS*]** command is issued in order to define the public autonomous system number. All routers within the confederation must specify this option. Next, for routers that are on the edge of the sub-autonomous system and are peering with other sub-autonomous systems, the **bgp confederation peers [*peer sub-ASs*]** command is used. Only the sub-ASs that are directly peered with need be listed in this statement. Lastly, BGP neighbors are established as usual. The sub-AS number should be referenced for any peers that are within the confederation.

Inter sub-AS peerings exhibit both EBGP and iBGP characteristics. The most notable difference between a normal iBGP peering relationship and an inter sub-AS peering is that iBGP mesh need not be maintained between sub-ASs. However, each sub-AS exhibits normal iBGP behavior within the sub-AS. This means that within each sub-AS there must either be full iBGP mesh or route-reflection. The majority of other attributes between inter sub-AS peers behave as normal iBGP peers. For example, next-hop, local-preference, and MED maintain their values as they are passed between sub-ASs. However, AS-Path information does not.

Confederation sub-AS path information is denoted within parentheses in the AS-path list. However, when it comes to best path selection, all sub-ASs within the parentheses count as a single AS.

Task 3.2 Verification

Verify BGP peering:

```
Rack1R5#show ip bgp summary  | begin Neighbor
Neighbor   V AS   MsgRcvd MsgSent TblVer InQ OutQ Up/Down  State/PfxRcd
150.1.4.4  4 65004   8      10       14    0    0 00:04:07        3
163.1.35.3 4   300  11       9       14    0    0 00:04:25       10
163.1.57.7 4 65007   7      10       14    0    0 00:03:52        0
```

Verify BGP paths inside the confederation; for instance verify the paths that contain two confederation sub-ASs on SW1:

```
Rack1SW1#show ip bgp regexp \([0-9]+_[0-9]+\)
BGP table version is 14, local router ID is 150.1.7.7
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
*> 205.90.31.0  192.10.1.254        0    100      0 (65005 65004) 254 ?
*> 220.20.3.0   192.10.1.254        0    100      0 (65005 65004) 254 ?
*> 222.22.2.0   192.10.1.254        0    100      0 (65005 65004) 254 ?
```

Lastly, verify that confederation announces prefixes, learned from AS254, to AS 300:

```
Rack1R3#show ip bgp regexp ^200
BGP table version is 14, local router ID is 150.1.3.3
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
*> 205.90.31.0      163.1.35.5                             0 200 254 ?
*> 220.20.3.0       163.1.35.5                             0 200 254 ?
*> 222.22.2.0       163.1.35.5                             0 200 254 ?
```


Task 3.3

R6:

```
router bgp 100
 neighbor 54.1.7.254 route-map BB1 in
 neighbor 204.12.1.254 route-map BB3 in
!
ip as-path access-list 1 permit _54$
!
route-map BB1 permit 10
 match as-path 1
 set local-preference 200
!
route-map BB1 permit 20
!
route-map BB3 permit 10
 match as-path 1
!
route-map BB3 permit 20
 set local-preference 200
```

Task 3.3 Breakdown

Recall the four common attributes used to affect the BGP best path selection, and how they are applied:

Attribute	Direction Applied	Traffic Flow Affected
Weight	Inbound	Outbound
Local-Preference	Inbound	Outbound
AS-Path	Outbound	Inbound
MED	Outbound	Inbound

As a general rule, weight and local-preference are used to affect how traffic leaves the autonomous system, while AS-Path and MED are used to affect how traffic enters the AS. The above task requires that all traffic leaving AS 100 going to prefixes originated in AS 54 exits to BB1, while traffic going to AS 54's customers exits to BB3. Therefore, as prefixes are learned from AS 54, either the weight or local-preference attribute should be modified to obtain the desired effect. However, since the above task states that the configuration should be done only on R6, weight cannot be used. Weight is only local to the router, and is not passed on with BGP updates.

Task 3.3 Verification

Verify that R6 has selected the best paths for AS54 originated prefixes via BB1:

```
Rack1R6#show ip bgp regexp _54$
BGP table version is 24, local router ID is 150.1.6.6
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
*  28.119.16.0/24   204.12.1.254             0             0 54 i
* i                 204.12.1.254             0    100      0 54 i
*>                  54.1.7.254                    200      0 54 i
*  28.119.17.0/24   204.12.1.254             0             0 54 i
* i                 204.12.1.254             0    100      0 54 i
*>                  54.1.7.254                    200      0 54 i
*  114.0.0.0        204.12.1.254                           0 54 i
<output omitted>
```

Now verify R2's BGP table:

```
Rack1R2#show ip bgp regexp _54$
BGP table version is 16, local router ID is 150.1.2.2
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
* i28.119.16.0/24   54.1.7.254               0    200      0 54 i
*>                  204.12.1.254             0             0 54 i
* i28.119.17.0/24   54.1.7.254               0    200      0 54 i
*>                  204.12.1.254             0             0 54 i
<output omitted>
```

Note that R2 does not prefer the path via BB1 due to the fact that the next-hop of 54.1.7.254 is unreachable. This will be corrected in the next task.

Lastly verify that the best paths to AS54 customers are through BB3:

```
Rack1R2#show ip bgp regexp 54(_[0-9]+)+$
BGP table version is 16, local router ID is 150.1.2.2
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
*>i112.0.0.0        204.12.1.254             0    200      0 54 50 60 i
*                   204.12.1.254                           0 54 50 60 i
*>i113.0.0.0        204.12.1.254             0    200      0 54 50 60 i
*                   204.12.1.254                           0 54 50 60 i
```


Task 3.4

R6:

```
router bgp 100
 neighbor 204.12.1.2 next-hop-self
```

Task 3.4 Breakdown

Two conditions must be met before a route can be considered for BGP best-path selection. The synchronization rule must be met, and there must be a route to the next-hop IP address of the prefix in question.

When a BGP update is passed on to an EBGP neighbor, the next-hop value of the prefix is updated to the local address used to peer with that neighbor. However, when an update is passed on to an iBGP neighbor, the next-hop value is not updated. In certain cases, this may cause a problem.

In the above scenario, R6 is learning BGP prefixes from BB1. As this is an EBGP peering relationship, the next-hop IP address that R6 sees for these prefixes is the address it uses to peer with BB1, 54.1.7.254. When these routes are propagated on to R2, the next hop value is not updated, as R2 is an iBGP neighbor of R6. However, since R2 does not have an IGP route to the network 54.1.7.0/24, it cannot consider any prefixes learned from BB1 passed on via R6 for the BGP best path selection. In such a case, there are two options.

The first option is to inject the transit network into the IGP domain of the BGP network. This will allow all iBGP speaking neighbors to do an IGP lookup on the next-hop values of any prefixes learned from the external system. This method may be inefficient for networks with an exorbitant amount of EBGP neighbors, as these transit networks take up unnecessary space in the IGP table. This is due to the fact that traffic is never sent to a transit network, only past it.

The second option is to modify the next-hop value as the prefix is advertised into the iBGP domain. This is accomplished by issuing the **neighbor [*address*] next-hop-self** BGP process subcommand. This command overrides the default next-hop processing behavior, and modifies the next-hop value to the local address when an EBGP learned prefix is advertised to an iBGP neighbor. This way, all internal BGP speaking routers only need to maintain IGP reachability information about the internal network.

Specifically in the case of this particular scenario, when R6 uses the **neighbor 204.12.1.2 next-hop-self** command, updates learned from BB1 have the next-hop value adjusted to 204.12.1.6 when advertised on to R2, as this is the address that R2 is peering with. Since this network is advertised into the IGP (RIP) domain, R1 can do an IGP lookup on all BGP routes learned from AS 54.

Task 3.4 Verification

Check to see that next-hop problem at R6 has been resolved:

```
Rack1R2#sh ip bgp regexp _54$
BGP table version is 26, local router ID is 150.1.2.2
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
*>i28.119.16.0/24   204.12.1.6               0    200      0 54 i
*                   204.12.1.254             0             0 54 i
*>i28.119.17.0/24   204.12.1.6               0    200      0 54 i
*                   204.12.1.254             0             0 54 i
*>i114.0.0.0        204.12.1.6               0    200      0 54 i
*                   204.12.1.254                           0 54 i
```

Note: Changing the next hop value on the neighbor adjacency to R2 will affect routes that R6 learns from both BB1 and BB3. If you want to just change the next hop value for the routes from BB1, you can be more granular by changing with an outbound route map that sets the next hop value.

Task 3.5

R6:

```
router bgp 100
 no bgp fast-external-fallover
 neighbor 54.1.7.254 timers 10 30
```

Task 3.5 Breakdown

By default, when a directly connected interface goes down, any EBGP peers which are directly connected on that interface have their BGP session reset. This will cause the router to withdrawn any BGP prefixes learned from that neighbor from all other BGP peers. This behavior may be undesired in certain cases, as in the case that this task describes. To disable this behavior and wait for the dead time to expire before declaring a directly connected EBGP down, use the **`no bgp fast-external-fallover`** bgp subcommand.

Additionally, the BGP timers have been adjusted for this neighbor by using the **`neighbor [address] timers [hello] [dead]`** command.

Task 3.5 Verification

Verify the BGP timers configuration. Pay close attention to what is listed as hold time and keepalive time. If you just change the timers, and the adjacency is still up, then you may not see the correct values, even though the values for "configured" hold time and keepalive are what you desire. You may want to shut/no shut the adjacency.

```
Rack1R6#show ip bgp neighbors 54.1.7.254
BGP neighbor is 54.1.7.254,  remote AS 54, external link
  BGP version 4, remote router ID 212.18.3.1
  BGP state = Established, up for 00:02:13
  Last read 00:00:03, last write 00:00:03, hold time is 30, keepalive
interval is 10 seconds
  Configured hold time is 30,keepalive interval is 10 seconds  Minimum
holdtime from neighbor is 0 seconds
<output omitted>
```

Just to be 100% sure we'll do some debugging (note the timestamps):

```
Rack1R6#debug ip bgp keepalives

*Apr 26 09:34:38.610: BGP: 204.12.1.254 sending KEEPALIVE (io)
*Apr 26 09:34:38.614: BGP: 204.12.1.254 received KEEPALIVE, length
(excl. header) 0
*Apr 26 09:34:39.610: BGP: 54.1.7.254 sending KEEPALIVE (io)
Rack1R6#
*Apr 26 09:34:39.654: BGP: 54.1.7.254 received KEEPALIVE, length (excl.
header) 0
*Apr 26 09:34:40.610: BGP: 204.12.1.2 sending KEEPALIVE (io)
*Apr 26 09:34:40.610: BGP: 204.12.1.2 received KEEPALIVE, length (excl.
header) 0
Rack1R6#
*Apr 26 09:34:49.610: BGP: 54.1.7.254 sending KEEPALIVE (io)
*Apr 26 09:34:49.654: BGP: 54.1.7.254 received KEEPALIVE, length (excl.
header) 0
```

To verify that fast fallover is disabled, try shutting down Serial 0/0/0 interface on R6:

```
Rack1R6(config)#int s0/0
Rack1R6(config-if)#shutdown
Apr 23 08:32:29.676: %LINK-5-CHANGED: Interface Serial0/0, changed
state to administratively down
Apr 23 08:32:29.680: %LINK-3-UPDOWN: Interface Virtual-Access1, changed
state to down
Apr 23 08:32:30.676: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial0/0, changed state to down
Apr 23 08:32:30.680: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Virtual-Access1, changed state to down
….
Apr 23 08:32:49.156: %BGP-5-ADJCHANGE: neighbor 54.1.7.254 Down BGP
Notification sent
Apr 23 08:32:49.156: %BGP-3-NOTIFICATION: sent to neighbor 54.1.7.254
4/0 (hold time expired) 0 bytes
```


Note that the peering session was not shut down immediately as with fast fallover enabled.

Task 3.6

R1:

```
router bgp 100
 aggregate-address 28.119.16.0 255.255.254.0 summary-only
 aggregate-address 112.0.0.0 248.0.0.0 summary-only
```

Task 3.6 Breakdown

By default, when configuring aggregation, the aggregate plus all subnets of the aggregate are advertised. By adding the summary-only keyword, only the aggregate is advertised.

Task 3.6 Verification

Verify that only the summary prefixes are sent to AS300 from R1:

```
Rack1R1#show ip bgp neighbors 163.1.18.8 advertised-routes  | begin Net
   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   0.0.0.0                              32768 i
*> 112.0.0.0/5      0.0.0.0                              32768 i
```

Task 3.7

R1:

```
router bgp 100
 neighbor 163.1.13.3 send-community
 neighbor 163.1.13.3 unsuppress-map UNSUPPRESS->R3
 neighbor 163.1.18.8 send-community
 neighbor 163.1.18.8 unsuppress-map UNSUPPRESS->SW2
!
ip prefix-list UNSUPPRESS->R3 seq 5 permit 28.119.17.0/24
ip prefix-list UNSUPPRESS->R3 seq 10 permit 116.0.0.0/8
ip prefix-list UNSUPPRESS->R3 seq 15 permit 117.0.0.0/8
ip prefix-list UNSUPPRESS->R3 seq 20 permit 118.0.0.0/8
ip prefix-list UNSUPPRESS->R3 seq 25 permit 119.0.0.0/8
!
ip prefix-list UNSUPPRESS->SW2 seq 5 permit 28.119.16.0/24
ip prefix-list UNSUPPRESS->SW2 seq 10 permit 112.0.0.0/8
ip prefix-list UNSUPPRESS->SW2 seq 15 permit 113.0.0.0/8
ip prefix-list UNSUPPRESS->SW2 seq 20 permit 114.0.0.0/8
ip prefix-list UNSUPPRESS->SW2 seq 25 permit 115.0.0.0/8
!
route-map UNSUPPRESS->SW2 permit 10
 match ip address prefix-list UNSUPPRESS->SW2
 set community no-export
!
route-map UNSUPPRESS->R3 permit 10
 match ip address prefix-list UNSUPPRESS->R3
 set community no-export
```


Task 3.7 Breakdown

The above task uses a combination of selective prefix unsuppressing and community manipulation in order to accomplish complex traffic engineering. When a router does a lookup in the IP routing table, it always chooses the route with the longest match to the destination. For example, suppose a packet is received with a destination IP address of 1.2.3.4, and the following routes are in the IP routing table:

```
0.0.0.0/0
1.0.0.0/8
1.2.0.0/16
1.2.3.0/24
```

Regardless of metric, administrative distance, or protocol, packets destined for 1.2.3.4 will always use the route 1.2.3.0/24. This is due to the fact that this is the longest match for the prefix in the routing table. By leaking specific longer matches to AS 300, AS 100 is able to affect the return traffic flow. This is due to the fact that routers in AS 300 have no choice but to use the longest match in the IP routing table.

As aggregation with the **`summary-only`** keyword has already been configured on R1, AS 300 only has the summary route regarding the prefixes learned from AS 54. Therefore, in order to force AS 300 to make routing decisions based on longer matches in the routing table, the prefixes mentioned in the task must be selectively unsuppressed on a per neighbor basis.

Next, the task states that devices beyond AS should not have this specific reachability information. This has been accomplished by setting the community value to no-export as the prefixes are unsuppressed. In this manner, AS 100 is able to force AS 300 to route a specific way, while at the same time ensuring upstream neighbors have the minimum amount of reachability information necessary.

☠ Pitfall

The above task can lead to problems with the order of operations of the BGP engine. For example, if the community no-export was set using a separate route-map than the unsuppress-map, it would not have been applied. This is due to the fact that the route-map is processed *before* the unsuppress-map. In that case, the route-map would be setting the community on prefixes that do not exist in the output engine. The subnets do not exist in the output engine until they are unsuppressed in the unsuppress-map. For this reason it is best always to consolidate all attribute setting in a single route-map. Do not mix and match the application of route-maps, unsuppress-maps, attribute-maps, distribute-lists, prefix-lists, or filter-lists to the same neighbor in the same direction.

Task 3.7 Verification

Verify that the specific prefixes are unsuppressed when sending updates to the respective neighbors:

Prefixes sent to SW2:

```
Rack1R1#show ip bgp neighbors 163.1.18.8 advertised-routes  | begin Net
   Network          Next Hop            Metric LocPrf Weight Path
s>i28.119.16.0/24   204.12.1.6               0    200      0 54 i
*> 28.119.16.0/23   0.0.0.0                               32768 i
s>i112.0.0.0        204.12.1.6               0    200      0 54 50 60 i
*> 112.0.0.0/5      0.0.0.0                               32768 i
s>i113.0.0.0        204.12.1.6               0    200      0 54 50 60 i
s>i114.0.0.0        204.12.1.6               0    200      0 54 i
s>i115.0.0.0        204.12.1.6               0    200      0 54 i
```

Next, prefixes sent to R3:

```
Rack1R1#show ip bgp neighbors 163.1.13.3 advertised-routes | begin Net
   Network          Next Hop        Metric LocPrf Weight Path
*> 28.119.16.0/23   0.0.0.0                        32768 i
s>i28.119.17.0/24   204.12.1.6           0    200      0 54 i
*> 112.0.0.0/5      0.0.0.0                        32768 i
s>i116.0.0.0        204.12.1.6           0    200      0 54 i
s>i117.0.0.0        204.12.1.6           0    200      0 54 i
s>i118.0.0.0        204.12.1.6           0    200      0 54 i
s>i119.0.0.0        204.12.1.6           0    200      0 54 i
*> 205.90.31.0      163.1.18.8                        0 300 200 254 ?
*> 220.20.3.0       163.1.18.8                        0 300 200 254 ?
*> 222.22.2.0       163.1.18.8                        0 300 200 254 ?
```


Finally verify that R3 advertises only the summary prefixes to AS200:

```
Rack1R3#show ip bgp neighbors 163.1.35.5 advertised-routes | begin Net
   Network          Next Hop            Metric LocPrf Weight Path
*> 28.119.16.0/23   163.1.13.1               0             0 100 i
*> 112.0.0.0/5      163.1.13.1               0             0 100 i
*> 205.90.31.0      163.1.35.5                             0 200 254 ?
*> 220.20.3.0       163.1.35.5                             0 200 254 ?
*> 222.22.2.0       163.1.35.5                             0 200 254 ?

Rack1R3#show ip bgp 28.119.17.0
BGP routing table entry for 28.119.17.0/24, version 31
Paths: (1 available, best #1, table Default-IP-Routing-Table, not
advertised to EBGP peer)
  Advertised to update-groups:
     1
  100 54
    163.1.13.1 from 163.1.13.1 (150.1.1.1)
      Origin IGP, localpref 100, valid, external, best
      Community: no-export
```

Lastly, verify full BGP connectivity. Run the following TCL script on every BGP enabled router:

```
foreach i {
205.90.31.1
220.20.3.1
222.22.2.1
28.119.16.1
28.119.16.1
28.119.17.1
112.0.0.1
113.0.0.1
114.0.0.1
115.0.0.1
116.0.0.1
117.0.0.1
118.0.0.1
119.0.0.1
} {puts [exec "ping $i"] }
```

Note that pings to AS254 networks from R6 using the source IP address of interface Fa0/1 were unsuccessful. This is due to the fact that this IP subnet is not announced to AS254.

4. IP and IOS Features

Task 4.1

R1 – SW4:

```
logging 163.1.5.100
logging 163.1.6.100
service timestamps log datetime msec
```

R2, R4, and R6:

```
logging facility local3
```

R1, R3, and R5:

```
logging facility local4
```

SW1 – SW4:

```
logging facility local5
```

Task 4.1 Verification

Verify basic syslog configuration:

```
Rack1SW2#show logging
Syslog logging: enabled (0 messages dropped, 1 messages rate-limited, 0
flushes, 0 overruns, xml disabled, filtering disabled)
<output omitted>

    File logging: disabled
    Trap logging: level informational, 99 message lines logged
        Logging to 163.1.5.100, 1 message lines logged, xml disabled,
               filtering disabled
        Logging to 163.1.6.100, 1 message lines logged, xml disabled,
```

Verify that milliseconds are included into timestamps:

```
Rack1SW2#show logging  | begin Log Buffer
Log Buffer (4096 bytes):

*Mar  1 08:44:51.441: %SYS-5-CONFIG_I: Configured from console by
console
```

Task 4.2

R1 – SW4:

```
ntp source Loopback0
ntp server 204.12.1.254
```


Task 4.2 Verification

```
Rack1SW2#show ntp status
Clock is synchronized, stratum 5, reference is 204.12.1.254
nominal freq is 250.0000 Hz, actual freq is 250.0000 Hz, precision is
2**18
reference time is AF82175D.3E67403F (07:21:01.243 UTC Fri Apr 23 1993)
clock offset is 1.1874 msec, root delay is 7.68 msec
root dispersion is 1.28 msec, peer dispersion is 0.08 msec

Rack1SW2#show ntp associations

      address  ref clock   st  when  poll reach  delay  offset    disp
*~204.12.1.254 127.127.7.1  4    11    64  377     7.7    1.19     0.1
 * master (synced), # master (unsynced), + selected, - candidate, ~
configured
```

Task 4.3

```
R6:
ip dhcp excluded-address 163.1.6.0 163.1.6.127
ip dhcp excluded-address 163.1.6.130
ip dhcp excluded-address 163.1.6.251 163.1.6.255
!
ip dhcp pool VLAN6_POOL
   network 163.1.6.0 255.255.255.0
   default-router 163.1.6.6
   domain-name InternetworkExpert.com
```

Task 4.3 Breakdown

In SOHO environments, the DHCP server functionality of IOS can significantly reduce the administrative overhead of maintaining static IP addressing or dedicating another machine to run DHCP. To enable the DHCP server process, first create a DHCP pool by issuing the **ip dhcp pool [name]** global configuration command. This will bring you to the DHCP server mode.

Once inside the DHCP server mode, specify the address pool by issuing the **network** command. The default gateway is then specified with the **default-router** option, DNS servers with the **dns-server** command, and domain-name with the **domain-name** command.

In order to control which portion of the specified network is assigned, exclude address space that should not be leased out by issuing the **ip dhcp excluded-address [start] [end]** global configuration command.

Task 4.3 Verification

```
Rack1R6#show ip dhcp pool

Pool VLAN6_POOL :
 Utilization mark (high/low)    : 100 / 0
 Subnet size (first/next)       : 0 / 0
 Total addresses                : 254
 Leased addresses               : 0
 Pending event                  : none
 1 subnet is currently in the pool :
 Current index        IP address range                    Leased
addresses
 163.1.6.1            163.1.6.1        - 163.1.6.254       0
```

To verify address allocation, temporarily enable Vlan6 SVI on SW1:

```
Rack1SW1#debug dhcp
Rack1SW1(config)#interface vlan 6
Rack1SW1(config-if)#ip address dhcp

 DHCP: DHCP client process started: 10
 RAC: Starting DHCP discover on Vlan6
 DHCP: Try 1 to acquire address for Vlan6
 DHCP: allocate request
 DHCP: new entry. add to queue
 DHCP: SDiscover attempt # 1 for entry:
 DHCP: SDiscover: sending 298 byte length DHCP packet
 DHCP: SDiscover 298 bytes
             B'cast on Vlan6 interface from 0.0.0.0
 DHCP: Received a BOOTREP pkt
 DHCP: offer received from 163.1.6.6
 DHCP: SRequest attempt # 1 for entry:
 DHCP: SRequest- Server ID option: 163.1.6.6
 DHCP: SRequest- Requested IP addr option: 163.1.6.128
 DHCP: SRequest placed lease len option: 86400
 DHCP: SRequest: 316 bytes
 DHCP: SRequest: 316 bytes
             B'cast on Vlan6 interface from 0.0.0.0
 DHCP: Received a BOOTREP pkt
Interface Vlan6 assigned DHCP address 163.1.6.128, mask 255.255.255.0

 DHCP Client Pooling: ***Allocated IP address: 163.1.6.128
 Allocated IP address = 163.1.6.128  255.255.255.0
```


Verify the DHCP bindings on R6:

```
Rack1R6#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address          Client-ID/              Lease expiration
Type
                    Hardware address/
                    User name
163.1.6.128         0063.6973.636f.2d30.    Apr 24 1993 07:26 AM
Automatic
                    3030.662e.3866.6232.
                    2e65.3830.302d.566c.
                    36
```

Task 4.4

R6:

```
interface FastEthernet0/0
 ip mobile arp access-group 2
!
router rip
 redistribute mobile metric 1
!
access-list 2 permit 163.1.5.25
!
ip prefix-list RIP seq 15 permit 163.1.5.25/32
```

Task 4.4 Breakdown

Not to be confused with Mobile IP, Local Area Mobility (LAM) offers a simple way for mobile hosts to roam around the network. When the **ip mobile arp** command is issued on the interface, the LAM process starts listening for ARP requests received on the interface that are from hosts which are not in the IP subnet of that interface. When these requests are received, the LAM processes knows that the ARP came from a mobile host. This hosts IP address is then installed in the IP routing table as a mobile host route. Additionally, ARP requests are sent to the host at a more frequent interval (minutes instead of hours) in order to ensure that they are still on the segment.

The **access-group** option tells the router which hosts to listen for ARP requests from. By default, any host on the segment can be mobile.

Note that LAM is not very scalable, as each mobile host requires a host route entry in the IP routing table.

Task 4.4 Verification

To verify LAM, configure a new interface VL6 SVI on SW1.

```
Rack1SW1#conf t
Rack1SW1(config)#interface vlan 6
Rack1SW1(config-if)#ip address 163.1.5.25 255.255.255.0
```

Generate ARP packets from SW1:

```
Rack1SW1#ping 163.1.5.254

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 163.1.5.254, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Check mobile routes on R6:

```
Rack1R6#show ip route mobile
     163.1.0.0/16 is variably subnetted, 15 subnets, 2 masks
M       163.1.5.25/32 [3/1] via 163.1.5.25, 00:00:51, FastEthernet0/0
```

Verify that the mobile route is redistributed:

```
Rack1R2#show ip route 163.1.5.25
Routing entry for 163.1.5.25/32
  Known via "rip", distance 120, metric 1
  Redistributing via rip, ospf 1
  Advertised by ospf 1 subnets
  Last update from 204.12.1.6 on FastEthernet0/0, 00:00:14 ago
  Routing Descriptor Blocks:
  * 204.12.1.6, from 204.12.1.6, 00:00:14 ago, via FastEthernet0/0
      Route metric is 1, traffic share count is 1
```

Finally do a traceroute from the "mobile" host:

```
Rack1SW1#traceroute 163.1.5.5

Type escape sequence to abort.
Tracing the route to 163.1.5.5

  1  *
    163.1.6.6 0 msec 0 msec
  2 204.12.1.2 0 msec 0 msec 0 msec
  3 163.1.12.1 4 msec 4 msec 4 msec
  4 163.1.15.5 68 msec *  64 msec
```


5. IP Multicast

Task 5.1

R4:

```
ip multicast-routing
!
interface FastEthernet0/1
 ip pim dense-mode
!
interface Serial0/1
 ip pim dense-mode
!
ip mroute 0.0.0.0 0.0.0.0 Serial0/1
```

R5:

```
ip multicast-routing
!
interface FastEthernet0/0
 ip pim dense-mode
!
interface Serial0/1
 ip pim dense-mode
```

Task 5.1 Breakdown

Another option to using a static mroute would be to manipulate the IGP routing protocols so that R4 prefers to route across the Serial over the Frame Relay to reach R5. This will ensure that the RPF check is successful when a static mroute is not used.

Task 5.1 Verification

To verify the multicast configuration, join VLAN4 interface to group 239.4.4.4.

Next, ping 239.4.4.4 from R5 using VLAN5 as the source:

```
Rack1R5#ping
Protocol [ip]:
Target IP address: 239.4.4.4
Repeat count [1]: 10
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Interface [All]: Serial0/1
Time to live [255]:
Source address: 163.1.5.5
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 239.4.4.4, timeout is 2 seconds:
Packet sent with a source address of 163.1.5.5

Reply to request 0 from 163.1.45.4, 28 ms
Reply to request 1 from 163.1.45.4, 28 ms
Reply to request 2 from 163.1.45.4, 28 ms
Reply to request 3 from 163.1.45.4, 28 ms
Reply to request 4 from 163.1.45.4, 28 ms
```

Verify multicast routing table on R4:

```
Rack1R4#show ip mroute
IP Multicast Routing Table
<snip>

(*, 239.4.4.4), 00:04:57/stopped, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1, Forward/Dense, 00:04:57/00:00:00
    FastEthernet0/1, Forward/Dense, 00:04:57/00:00:00

(163.1.5.5, 239.4.4.4), 00:00:49/00:02:23, flags: LT
  Incoming interface: Serial0/1, RPF nbr 163.1.45.5, Mroute
  Outgoing interface list:
    FastEthernet0/1, Forward/Dense, 00:00:50/00:00:00
```


6. QoS

Task 6.1

R4:

```
!
ip access-list extended VOIP
 permit tcp any any eq 1720
 permit udp any any range 16384 32767
!
class-map match-all VOIP
 match access-group name VOIP
!
policy-map MARK_VOIP
 class VOIP
  set dscp cs5
 class class-default
  set dscp cs1
policy-map SET_DSCP_CS1
 class class-default
  set dscp cs1
!
interface FastEthernet0/1
 service-policy input MARK_VOIP
!
interface FastEthernet0/0
 service-policy input SET_DSCP_CS1
```

R5:

```
interface FastEthernet0/0
 service-policy input MARK_VOIP
!
interface FastEthernet0/1
 service-policy input SET_DSCP_CS1
!
interface Serial0/0.35 point-to-point
 service-policy input SET_DSCP_CS1
!
ip access-list extended VOIP
 permit tcp any any eq 1720
 permit udp any any range 16384 32767
!
class-map match-all VOIP
 match access-group name VOIP
!
policy-map MARK_VOIP
 class VOIP
  set dscp cs5
 class class-default
  set dscp cs1
policy-map SET_DSCP_CS1
 class class-default
  set dscp cs1
```


Task 6.1 Verification

Verify the policy-map configuration (note the QoS set operations):

```
Rack1R4#show policy-map interface
 FastEthernet0/0

  Service-policy input: SET_DSCP_CS1

    Class-map: class-default (match-any)
      1 packets, 118 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        dscp cs1
          Packets marked 1
 FastEthernet0/1

  Service-policy input: MARK_VOIP

    Class-map: VOIP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group name VOIP
      QoS Set
        dscp cs5
          Packets marked 0

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        dscp cs1
          Packets marked 0
```

Check access-list used for classification purposes:

```
Rack1R4#show ip access-list VOIP
Extended IP access list VOIP
    10 permit tcp any any eq 1720
    20 permit udp any any range 16384 32767
```


Task 6.2

R3:

```
interface Serial1/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 305
  class DLCI_305
!
map-class frame-relay DLCI_305
 frame-relay cir 768000
```

R4:

```
interface Serial0/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 405
  class DLCI_405
!
map-class frame-relay DLCI_405
 frame-relay cir 768000
 frame-relay bc 7680
 frame-relay fragment 960
```

R5:

```
interface Serial0/0
 frame-relay traffic-shaping
!
interface Serial0/0.35 point-to-point
 frame-relay interface-dlci 503
  class DLCI_503
!
interface Serial0/0.54 point-to-point
 frame-relay interface-dlci 504
  class DLCI_504
!
map-class frame-relay DLCI_504
 frame-relay cir 768000
 frame-relay bc 7680
 frame-relay fragment 960
!
map-class frame-relay DLCI_503
 frame-relay cir 768000
```


Task 6.2 Breakdown

Frame Relay fragmentation allows for minimal delay when sending small real time packets across a Frame Relay circuit, such as VoIP. As previously discussed, a router always sends traffic out an interface at the serialization, or clocking, of that interface. Typically, this results in periods of traffic bursts, followed by periods of no activity. Traffic shaping attempts to smooth these peaks and valleys by pacing the output of packets on the interface. However, as the transmit ring (hardware queue) of an interface is always first in, first out (FIFO), small real time packets can experience unacceptable delay when they are stuck behind large data packets, even when priority queueing is enabled.

By reducing the maximum packet size that can be sent out the Frame Relay circuit, fragmentation reduces the worst case delay that a real time packet must endure. Typically, the fragmentation size is set to match the Bc. This guarantees that the worst case delay a packet will endure is a single Tc interval.

Task 6.2 Verification

Verify shaping & fragmentation configuration:

```
Rack1R5#show frame-relay pvc 504

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE (EEK UP), INTERFACE
= Serial0/0.54

<output omitted>

  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    64          16           0
  Output queue size 0/max total 600/drops 0
  fragment type end-to-end fragment size 960
  cir 768000    bc   7680      be 0         limit 960    interval 10
  mincir 384000    byte increment 960   BECN response no  IF_CONG no
  frags 107       bytes 10122     frags delayed 0       bytes delayed 0
  shaping inactive
  traffic shaping drops 0
```

Task 6.3

R4:

```
class-map match-all DSCP_CS5
 match  dscp cs5
!
policy-map FR_QOS
 class DSCP_CS5
 class class-default
  set fr-de
!
map-class frame-relay DLCI_405
 service-policy output FR_QOS
```

R5:

```
class-map match-all DSCP_CS5
 match  dscp cs5
!
policy-map FR_QOS
 class DSCP_CS5
 class class-default
  set fr-de
!
map-class frame-relay DLCI_504
 service-policy output FR_QOS
```

Task 6.3 Breakdown

The Frame Relay discard-eligible bit tells the Frame Relay switches in the provider cloud which frames to drop first when congestion occurs. By manually setting the DE bit on non essential traffic, real time traffic such as VoIP is less likely to be dropped in the event of congestion.

Task 6.3 Verification

Verify DE-marking configuration:

```
Rack1R5#show frame-relay pvc 504

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE (EEK UP), INTERFACE
= Serial0/0.54

<output omitted>

 Serial0/0.54: DLCI 504 -

  Service-policy output: FR_QOS

    Class-map: DSCP_CS5 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps
      Match:  dscp cs5 (40)

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        fr-de
          Packets marked 1
  <output omitted>
```

Task 6.4

R4:

```
policy-map FastEthernet_QOS
 class DSCP_CS5
  priority 256
policy-map FR_QOS
 class DSCP_CS5
  priority 256
!
interface FastEthernet0/1
 service-policy output FastEthernet_QOS
```

R5:

```
policy-map FastEthernet_QOS
 class DSCP_CS5
  priority 256
policy-map FR_QOS
 class DSCP_CS5
  priority 256
!
interface FastEthernet0/0
 service-policy output FastEthernet_QOS
```


Task 6.4 Breakdown

The **priority 256** command ensures that all VoIP traffic up to 256Kbps in the output queue is dequeued first. This prioritization ensures the minimum amount of delay for priority packets as they sit in the output queue.

Task 6.4 Verification

Verify LLQ configuration:

```
Rack1R5#show frame-relay pvc 504

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 504, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE (EEK UP), INTERFACE
= Serial0/0.54

<output omitted>

  service policy FR_QOS
 Serial0/0.54: DLCI 504 -

  Service-policy output: FR_QOS

    Class-map: DSCP_CS5 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:  dscp cs5 (40)
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 256 (kbps) Burst 6400 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      172 packets, 16103 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        fr-de
          Packets marked 172
<output omitted>
```


7. Security

Task 7.1

R4:

```
interface FastEthernet0/1
 ip access-group NO_FRAGMENTS out
!
ip access-list extended NO_FRAGMENTS
 deny   ip any any fragments
 permit ip any any
```

Task 7.1 Verification

To verify the access-list NO_FRAGMENTS, you can add an interface on that VLAN. Configuring an access list on R4 to only permit ICMP will allow you to restrict debug output for "debug ip packet detail" by referencing the ACL. Ping from R5 using a packet size that will require fragmentation:

SW3 (only for testing, remove when done)

```
Int vlan 4
 Ip address 163.1.4.99 255.255.255.0
Ip route 150.1.5.5 255.255.255.255 163.1.4.4
```

R4

```
Access-list 123 permit icmp any any
(Used in conjunction with "debug ip packet detail 123")

Rack1R5#ping 163.1.4.99 size 1700 source 150.1.5.5

Type escape sequence to abort.
Sending 5, 1700-byte ICMP Echos to 163.1.4.99, timeout is 2 seconds:
Packet sent with a source address of 150.1.5.5
.....
Success rate is 0 percent (0/5)
Rack1R5#
```

R4 Debug output:

```
IP: tableid=0, s=150.1.5.5 (Serial0/1), d=163.1.4.99 (FastEthernet0/1),
routed via FIB
 IP: s=150.1.5.5 (Serial0/1), d=163.1.4.99 (FastEthernet0/1), len 220,
access denied
     IP Fragment, Ident = 55, fragment offset = 1480
Rack1R4#
 IP: tableid=0, s=150.1.5.5 (Serial0/1), d=163.1.4.99
(FastEthernet0/1), routed via FIB
 IP: s=150.1.5.5 (Serial0/1), d=163.1.4.99 (FastEthernet0/1), len 220,
access denied
     IP Fragment, Ident = 56, fragment offset = 1480
Rack1R4#
 IP: tableid=0, s=150.1.5.5 (Serial0/1), d=163.1.4.99
(FastEthernet0/1), routed via FIB
 IP: s=150.1.5.5 (Serial0/1), d=163.1.4.99 (FastEthernet0/1), len 220,
access denied
     IP Fragment, Ident = 57, fragment offset = 1480
Rack1R4#
```



```
 IP: tableid=0, s=150.1.5.5 (Serial0/1), d=163.1.4.99
(FastEthernet0/1), routed via FIB
 IP: s=150.1.5.5 (Serial0/1), d=163.1.4.99 (FastEthernet0/1), len 220,
access denied
Mar 19 07:18:26.048:     IP Fragment, Ident = 58, fragment offset =
1480
Rack1R4#
 IP: tableid=0, s=150.1.5.5 (Serial0/1), d=163.1.4.99
(FastEthernet0/1), routed via FIB
 IP: s=150.1.5.5 (Serial0/1), d=163.1.4.99 (FastEthernet0/1), len 220,
access denied
     IP Fragment, Ident = 59, fragment offset = 1480
```

Note that non-initial fragments are denied.

Next, try pinging with packets small enough not to require fragmentation:

```
Rack1R5#ping 150.1.4.99 size 1000

Type escape sequence to abort.
Sending 5, 1000-byte ICMP Echos to 150.1.4.99, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 508/508/508
ms
```

Task 7.2

R4:

```
ip cef
!
class-map match-all ROOT_EXPLOIT
  match protocol http url "*root.exe*"
!
policy-map SET_DSCP_CS1
 class ROOT_EXPLOIT
  drop
!
interface FastEthernet0/0
 service-policy input SET_DSCP_CS1
```

Task 7.2 Breakdown

As of IOS 12.2(13)T, the **drop** command is an available option in the policy-map. As the command implies, any matching traffic is simply dropped. Prior to this option, traffic could be unconditionally dropped using the **police** keyword with both the conform and exceed actions set to drop.

Task 7.2 Verification

To verify the configuration, you can temporarily add an interface on SW3 and test, since you would not have access to the BB in the actual lab. Make sure to remove after testing.

SW3 (only for testing, remove when done)

```
int vlan42
 ip address 192.10.1.3 255.255.255.0
 ip route 150.1.6.6 255.255.255.255 192.10.1.4

Rack1SW3#copy http://admin:cisco@150.1.6.6/root.exe null:
%Error opening http://admin:cisco@150.1.6.6/root.exe (Unknown error -1)
```

Your output should hang if the configuration is working.

Next verify the policy-map on R4:

```
Rack1R4#show policy-map interface F0/0
 FastEthernet0/0

  Service-policy input: SET_DSCP_CS1

    Class-map: ROOT_EXPLOIT (match-all)
      6 packets, 1135 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol http url "*root.exe*"
      drop
<output omitted>
```


