Task 1.1

SW1:

```
interface FastEthernet0/13
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface range Fa0/16,Fa0/19
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk allowed vlan except 7,77,777
```

SW2, SW3, and SW4:

```
interface FastEthernet0/13
 switchport trunk encapsulation dot1q
 switchport mode trunk
```

> ☞ **Quick Note**
> When the *allowed vlan except* option is used, the configuration will show the command without the except option displaying all of the allowed VLANs.

Task 1.1 Verification

```
Rack1SW1#show interface trunk | include (Encap|802|allowed on|4094)
Port        Mode         Encapsulation  Status        Native vlan
Fa0/13      on           802.1q         trunking      1
Fa0/16      on           802.1q         trunking      1
Fa0/19      on           802.1q         trunking      1
Port        Vlans allowed on trunk
Fa0/13      1-4094
Fa0/16      1-6,8-76,78-776,778-4094
Fa0/19      1-6,8-76,78-776,778-4094
```

Task 1.2

SW1:

```
spanning-tree vlan 1-4094 root primary
```

Task 1.2 Verification

```
Rack1SW1#show span vlan 1-4094 | i VLAN|root|FWD|BLK
VLAN0001
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/15           Desg FWD 19        128.17   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/17           Desg FWD 19        128.19   P2p
Fa0/18           Desg FWD 19        128.20   P2p
Fa0/19           Desg FWD 19        128.21   P2p
Fa0/20           Desg FWD 19        128.22   P2p
Fa0/21           Desg FWD 19        128.23   P2p
VLAN0005
             This bridge is the root
Fa0/5            Desg FWD 19        128.7    P2p
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/19           Desg FWD 19        128.21   P2p
VLAN0006
             This bridge is the root
```



```
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/19           Desg FWD 19        128.21   P2p
VLAN0007
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
VLAN0010
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/19           Desg FWD 19        128.21   P2p
VLAN0027
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/19           Desg FWD 19        128.21   P2p
VLAN0032
             This bridge is the root
Fa0/3            Desg FWD 19        128.5    P2p
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/19           Desg FWD 19        128.21   P2p
VLAN0042
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
Fa0/16           Desg FWD 19        128.18   P2p
Fa0/19           Desg FWD 19        128.21   P2p
VLAN0077
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
VLAN0777
             This bridge is the root
Fa0/13           Desg FWD 19        128.15   P2p
Rack1SW1#

Rack1SW1#show interface trunk | begin allowed and active
Port        Vlans allowed and active in management domain
Fa0/13      1,5-7,10,27,32,77,777
Fa0/16      1,5-6,10,27,32
Fa0/19      1,5-6,10,27,32

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/13      1,5-7,10,27,32,77,777
Fa0/16      1,5-6,10,27,32
Fa0/19      1,5-6,10,27,32
```

Task 1.3

SW3 and SW4:
```
system mtu 1504
```

SW3:
```
vlan 100
```

SW3:
```
interface FastEthernet0/18
switchport access vlan 100
switchport mode dot1q-tunnel
l2protocol-tunnel cdp
```

SW4:
```
interface FastEthernet0/4
switchport access vlan 100
switchport mode dot1q-tunnel
l2protocol-tunnel cdp
```

Task 1.3 Breakdown

The basic concept behind 802.1q tunneling (QinQ) is to allow for an additional tag to be applied to the Ethernet frame. This is commonly used by service providers to provide end-to-end transparent Ethernet services for their customers (Metro Ethernet). This additional tag, sometimes called the metro tag, allows for the service provider to carry all of the customer's traffic in a single separate VLAN without concern as to what traffic is being carried. This traffic could be unicast, broadcast, multicast, CDP, STP, or VTP.

QinQ tunneling can additionally allow the customer to trunk transparently across the service provider's network. When the customer's switch is trunking "to" the service provider's switch, all of the customer's trunks are carried inside the single metro VLAN when transiting the service provider's switches. In this case, the Ethernet frames will carry two tags. The inner tag was assigned by the customer's switches (i.e. the customers VLANs), and the outer tag is assigned by the service provider's edge switch (Metro tag). In order to support the additional extra 4 byte metro tag, the system MTU should be set to 1504. The default system MTU is 1500 bytes.

When using QinQ tunneling, CDP, STP and VTP are not carried across the *tunnel* by default. To allow for the carrying of these protocols, the interfaces on the service provider's edge switches need to be configured.

Additionally, QinQ also provides support for Etherchannel between customer sites. This will discussed in future labs.

> **Note**
>
> In this task, SW2 and R4 are considered the customer devices and SW3 & SW4 are the provider edge switches.

Task 1.3 Verification

Looking at the configuration afterwards on the switches, you may notice that the command no cdp enable is added automatically. Also, depending how quickly you check after configuration, you may still see an entry on R4 that shows SW4 as a neighbor. Pay attention to the holdtime values. A higher value indicates a more recently received CDP message.

```
Rack1R4#ping 191.1.48.8

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 191.1.48.8, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1R4#show cdp neighbors Fa0/1 | include SW2
Rack1SW2     Eth 0/1        121         S I       WS-C3560-2Fas 0/18

Rack1SW2#show cdp neighbors fa0/18 | include R4
Rack1R4      Fas 0/18       134         R S I     3640      Eth 0/1
```

Task 1.4

SW2:

```
interface FastEthernet0/10
 switchport mode access
 switchport port-security
 switchport port-security maximum 4
 switchport port-security violation restrict
 switchport port-security mac-address 0050.7014.8ef0
 switchport port-security mac-address 00c0.144e.07bf
 switchport port-security mac-address 00d0.341c.7871
 switchport port-security mac-address 00d0.586e.b710
!
logging 191.1.7.100
```


Task 1.4 Breakdown

Layer 2 security based on source MAC address of a frame is controlled by *port security*. Port security allows you to define either specific MAC addresses that can send traffic into a port or how many MAC addresses can send traffic into a port. The first step in enabling port security is to set the port mode to access. This is accomplished by issuing the **`switchport mode access`** command. Port security is not supported on dynamic ports. Next, enable port security by issuing the **`switchport port-security`** interface command.

By default, port security only allows one MAC address to use a port. The above task states that four MAC address should be allowed entry. The task specifically lists their addresses. Therefore, the maximum allowed addresses must be increased by issuing the **`switchport port-security maximum [num]`** command. Next, the addresses are defined by issuing the **`switchport port-security mac-address [address]`** command.

Next, the task states that other hosts which try to access the port should be logged. By default the violate action of port security is *shutdown*. This means that the port it is sent to err-disabled state when either an insecure MAC is heard, or the maximum MAC addresses is exceeded. In addition to shutdown, restrict and protect are included as additional violate actions. When the violation mode is set to protect, traffic from MAC addresses that are not secure or are in excess of the maximum value is discarded. When violation is set to restrict the behavior is the same as protect, but a syslog message an SNMP trap is generated as well. Use the interface level command **`switchport port-security violation`** command to change the violation mode.

Task 1.4 Verification

Verify the port-security configuration:

```
Rack1SW2#show port-security interface fa0/10
Port Security              : Enabled
Port Status                : Secure-down
Violation Mode             : Restrict
Aging Time                 : 0 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 4
Total MAC Addresses        : 4
Configured MAC Addresses   : 4
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 0000.0000.0000:0
Security Violation Count   : 0
```


Verify the configured secure MAC addresses:

```
Rack1SW2#show port-security interface fa0/10 address
          Secure Mac Address Table
------------------------------------------------------------------------
Vlan    Mac Address       Type                      Ports   Remaining
Age
                                                              (mins)
----    -----------       ----                      -----   ------------
  10    0050.7014.8ef0    SecureConfigured          Fa0/10        -
  10    00c0.144e.07bf    SecureConfigured          Fa0/10        -
  10    00d0.341c.7871    SecureConfigured          Fa0/10        -
  10    00d0.586e.b710    SecureConfigured          Fa0/10        -
------------------------------------------------------------------------
Total Addresses: 4
```

Task 1.5

SW2:

```
spanning-tree portfast bpdufilter default
!
interface FastEthernet0/10
 spanning-tree portfast
```


Task 2.1

R1:

```
router ospf 1
 router-id 150.1.1.1
 network 191.1.125.1 0.0.0.0 area 0
 neighbor 191.1.125.2
 neighbor 191.1.125.5
```

R2:

```
interface Serial0/0
 ip ospf priority 0
!
router ospf 1
 router-id 150.1.2.2
 network 191.1.125.2 0.0.0.0 area 0
```

R5:

```
interface Serial0/0/0
 ip ospf priority 0
!
router ospf 1
 router-id 150.1.5.5
 network 191.1.125.5 0.0.0.0 area 0
```

Task 2.1 Breakdown

As the Frame Relay section dictates that R1, R2, and R5 must use the main interface for their hub-and-spoke configuration, the default OSPF network type will be non-broadcast. Additionally, since this section dictates that the **ip ospf network** command cannot be used on any of these devices, the default of non-broadcast must remain. Therefore, R1 has been configured to specify its unicast neighbors, R2 and R5, and R2 and R5 have adjusted their OSPF priority value to remove participation in the DR/BDR election. As R1 is the only device on this segment that has a direct layer 2 connection to all endpoints of the network, it is mandatory that R1 be elected the DR.

Task 2.1 Verification

Verify OSPF network type (non-broadcast) and the DR for the segment:

```
Rack1R1#show ip ospf interface s0/0
Serial0/0 is up, line protocol is up
  Internet Address 191.1.125.1/24, Area 0
  Process ID 1, Router ID 150.1.1.1,Network Type NON_BROADCAST,Cost: 64
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 150.1.1.1, Interface address 191.1.125.1
```


Verify the OSPF neighbors:

```
Rack1R1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.5.5         0   FULL/DROTHER 00:01:43    191.1.125.5     Serial0/0
150.1.2.2         0   FULL/DROTHER 00:01:34    191.1.125.2     Serial0/0
```

Task 2.2

R2:

```
router ospf 1
 area 27 nssa no-redistribution no-summary
```

SW1:

```
router ospf 1
 area 27 nssa
```

Task 2.2 Breakdown

The above task states that SW1 does not require specific reachability information to the rest of the IGP domain, as its only connection out is through R2. As previously demonstrated, this can be accomplished by defining the area in question as a type of stub area. The next issue that must be addressed is which type of stub area to configure.

Stub Type	Keyword	LSAs	Default Injected
Stub	area x stub	1,2,3	YES
Totally Stubby	area x stub no-summary	1,2, default of 3	YES
Not-So-Stub	area x nssa	1,2,3,7	NO
Not-So-Totally-Stubby	area x nssa no-summary	1,2, default of 3, 7	YES

Recall the previously defined stub areas. The above task states that the only IGP route it should see is a default route generated by R2, the ABR. The only stub area type that does not automatically generate a default route into the area is the not-so-stubby area. However, a default route can be manually generated into the NSSA area by adding the **`default-originate`** keyword on to the end of the **`area [area] nssa`** statement. Therefore, the requirement of a default route alone does not narrow our choices. The keyword for the above ask is that SW1 should not see any *other* IGP routes except this default. This requirement implies that inter-area or external reachability information should not be injected into area 27. This narrows our choices down to two stub types, the totally stubby area and the not-so-totally-stubby area.

Recall from the previous task that the Loopback 0 interfaces of all routers were injected into the OSPF domain by issuing the **`redistribute connected`** command. This implies that redistribution must be allowed into area 27. This

further eliminates the stub area type of totally stubby, and leaves us with our last choice of not-so-totally-stubby.

The last two stipulations on this task give us a twist that has not been previously seen. The last two requirements state that SW1 should not see a specific route to R2's Loopback 0 network. As redistribution is allowed into a not-so-totally-stubby area, this route will be seen by SW1 unless additional configuration is performed. This prefix can be removed from SW1's routing table in a variety of ways. These include filtering the route out from the IP routing table with a distribute-list or a route-map, poisoning the distance of the prefix, or stopping the route from coming into the area by disallowing redistribution into the NSSA area on the ABR. The first two options cannot be used, as the requirement specifically denies their usage. Changing the distance of the prefix is a valid solution; however it was not the intended solution for the requirement.

The **no-redistribution** keyword on the end of the **area [area] nssa** statement is specifically designed to deal with the above scenario. When redistribution is performed on an OSPF device, the route is propagated into all areas unless it is manually blocked with a stub definition or filtering. This is also true of the ABR of an NSSA area. When a route is redistributed on the ABR of an NSSA, it also becomes an ASBR. This route is therefore propagated into the NSSA area as LSA 7 (N1 or N2 route), and as LSA 5 into all other areas. The **no-redistribution** keyword allows us to stop this default behavior. Although redistribution into the NSSA is still allowed, routes redistributed into the OSPF domain on the NSSA ABR itself are not propagated into the NSSA area. As in the above case, this behavior is advantageous.

Since SW1's only connection to the rest of the routing domain is through R2, it does not need specific routing information about other areas. Instead, this information can be replaced by a default route generated by R2. Therefore, SW1 does not require the amount of memory to hold the OSPF database as well as the IP routing table as other devices in the OSPF domain. This memory usage is further reduced by disallowing routes redistributed on R2 to go into area 27, as devices in area 27 will already have default reachability through R2.

Task 2.2 Verification

```
Rack1SW1#show ip ospf | begin Area 27
    Area 27
        Number of interfaces in this area is 4
        It is a NSSA area
<output omitted>

Verify the routing table on SW1:

Rack1SW1#show ip route ospf
O*IA 0.0.0.0/0 [110/2] via 191.1.27.2, 00:00:28, FastEthernet0/2

Verify that the other OSPF routers still see SW1's Loopback0 prefix:

Rack1R3#show ip route | include 150.1.7.0
O E2    150.1.7.0/24 [110/20] via 191.1.23.2, 00:01:49, Serial1/3
```

Task 2.3

R3:

```
router ospf 1
 default-information originate always route-map CONDITION
!
ip prefix-list BB2 seq 5 permit 192.10.1.0/24
!
ip prefix-list BB3 seq 5 permit 204.12.1.0/24
!
route-map CONDITION permit 10
 match ip address prefix-list BB2
!
route-map CONDITION permit 20
 match ip address prefix-list BB3
```

Task 2.3 Breakdown

The above task dictates that R3 should originate a default route into the OSPF domain. However, a stipulation is placed on its generation of this default. This default should only be generated if its connections to either BB2 or BB3 are up. This type of stipulation is known as *conditional* advertisement.

To enable the conditional advertisement of a default route in OSPF, a route-map is added onto the **default-information originate** statement. If the route-map indicated is true, a default route is originated. If the route-map is false, a default route is not originated. In the above example the route-map CONDITION specifies that either the prefix 192.10.1.0/24 or 204.12.1.0/24 must exist in the IP routing table. If this condition is true, the default route is originated.

Pitfall

When the **`default-information originate`** statement has a conditional route-map attached to it, the condition must be met in order to originate a default regardless of whether the *always* keyword is included. If the above route-map CONDITION is not met no default will be generated even if the *always* keyword is added.

Task 2.3 Verification

Verify that the conditional advertisement actually works:

```
Rack1R3#debug ip ospf lsa-generation
Rack1R3#conf t
Rack1R3(config)#int Fa0/0
Rack1R3(config-if)#shutdown
Rack1R3(config)#interface Fa0/1
Rack1R3(config-if)#shutdown

OSPF: Generate external LSA 0.0.0.0, mask 0.0.0.0, type 5, age 3600,
metric 16777215, tag 1, metric-type 2, seq 0x80000002

Rack1R3(config-if)#no shutdown
OSPF: Generate external LSA 0.0.0.0, mask 0.0.0.0, type 5, age 0,
metric 1, tag 1, metric-type 2, seq 0x80000003
```

Note that external LSA were purged and then reinstalled.

Task 2.4

R3:

```
router ospf 1
 redistribute rip subnets route-map RIP->OSPF
!
router rip
 redistribute connected metric 1 route-map CONNECTED->RIP
 redistribute ospf 1 metric 1
!
ip prefix-list R6_LOOPBACK0 seq 5 permit 150.1.6.0/24
!
route-map CONNECTED->RIP permit 10
 match interface FastEthernet0/0 Loopback0 Serial1/2 Serial1/3
Serial1/0
!
route-map RIP->OSPF permit 10
 match ip address prefix-list R6_LOOPBACK0
```


R6:

```
router rip
 version 2
 network 150.1.0.0
 network 204.12.1.0
 redistribute connected metric 1 route-map CONNECTED->RIP
 no auto-summary
!
route-map CONNECTED->RIP permit 10
 match interface Loopback0
```

Task 2.4 Breakdown

The task wording means that RIP should be redistributed into OSPF, but when RIP is redistributed into OSPF the only prefix that should be allowed is R6's Loopback 0 network. This is accomplished by matching R6's loopback in a prefix-list, then matching the prefix-list in a route-map, and using this route-map to filter the redistribution of RIP into OSPF.

Pitfall

R3's Loopback 0 interface has been advertised into the OSPF domain through redistribution. Although OSPF is redistributed into RIP, this does not imply that R3's Loopback 0 interface is redistributed into RIP. Indirect redistribution between two protocols cannot be accomplished on the same local devices. For example, suppose that protocol A is redistributed into protocol B. Protocol B is then redistributed into protocol C. This does not imply that protocol A was redistributed into protocol C. Instead, protocol A must be manually redistributed into protocol C to achieve the desired effect. This can be seen in the above output since R3's Loopback 0 network is redistributed as connected into the RIP domain.

Task 2.4 Verification

Verify that only R6's Loopback0 is redistributed into OSPF from RIP:

```
Rack1R1#show ip route ospf | include 30\.
Rack1R1#shpw ip route ospf | include 150.1.6.0
O E2    150.1.6.0/24 [110/20] via 191.1.13.3, 00:03:08, Serial0/1
```

Next verify that we have connectivity inside the OSPF domain and can ping R6's Loopback0 from every OSPF router.

Execute the following TCL script on routers R1 through R5. Note that script excludes R4's VLAN4 and R6 interfaces (except for Loopback0).

```
foreach i {
150.1.1.1
191.1.13.1
191.1.125.1
150.1.2.2
191.1.27.2
191.1.23.2
191.1.125.2
150.1.3.3
191.1.34.3
191.1.23.3
191.1.13.3
204.12.1.3
192.10.1.3
191.1.48.4
191.1.49.4
191.1.45.4
150.1.4.4
191.1.34.4
191.1.40.4
191.1.45.5
150.1.5.5
191.1.5.5
191.1.125.5
150.1.6.6
204.12.1.6
} { puts [exec "ping $i" ] }
```


Task 2.5

R6:

```
router bgp 100
 neighbor 204.12.1.254 route-map FROM_BB3 in
!
route-map SET_WEIGHT permit 10
 match ip address prefix-list SLASH_20_AND_UNDER
!
route-map FROM_BB3 permit 10
 match ip address prefix-list SLASH_20_AND_UNDER
!
ip prefix-list SLASH_20_AND_UNDER seq 5 permit 0.0.0.0/0 le 20
```

Task 2.5 Breakdown

Unlike the IP access-list, which was designed to match traffic, the IP prefix-list was designed specifically with network reachability information in mind. Prefix-lists are used to match on prefix (network) and prefix-length (subnet mask) pairs. The prefix-list has dual syntax meanings. The syntax is straightforward once you understand what it means; unfortunately the prefix-list is very sparsely documented.

Normal prefix-list syntax is as follows:

ip prefix-list [*name*] [permit | deny] [*prefix*]/[*len*]

Where *name* is any name or number, *prefix* is the exact routing prefix (network), and *len* is the exact prefix-length (subnet mask). Take the following examples:

```
ip prefix-list LIST permit 1.2.3.0/24
```

The above is an exact match for the network 1.2.3.0 with the exact subnet mask of 255.255.255.0. This list does not match 1.2.0.0/24, nor does it match 1.2.3.4/32, nor anything in between.

```
ip prefix-list LIST permit 0.0.0.0/0
```

The above is an exact match for the network 0.0.0.0 with the exact subnet mask of 0.0.0.0. This is used to match a default route.

Typical confusion about the prefix-list comes into play when the keywords "GE" (greater than or equal to) and "LE" (less than or equal to) are added to the prefix-list. This is due to the fact that the "len" value changes meaning when the GE or LE keywords are used.

This alternate syntax is as follows:

ip prefix-list [*name*] [permit | deny] [*prefix*]/[*len*] ge [*min_length*] le [*max_length*]

Where *name* is any name or number, *prefix* is the routing prefix to be checked against, *len* is the amount of bits starting from the most significant (left most) to check, *min_length* is the minimum subnet mask value, and *max_length* is the maximum subnet mask value.

When using the GE and LE values, the following condition must be satisfied:

len < GE <= LE

The above syntax, while confusing at first, simply means that a range of addresses will be matched based on the prefix and the subnet mask range.

Take the following examples:

```
ip prefix-list LIST permit 1.2.3.0/24 le 32
```

The above syntax means that the first 24 bits of the prefix 1.2.3.0 must match. Additionally, the subnet mask must be less than or equal to 32.

```
ip prefix-list LIST permit 0.0.0.0/0 le 32
```

The above syntax means that zero bits of the prefix must match. Additionally, the subnet mask must be less than or equal to 32. Since all networks have a subnet mask less than or equal to 32, and no bits of the prefix are matched, this statement equates to an explicit permit any.

```
ip prefix-list LIST permit 10.0.0.0/8 ge 21 le 29
```

The above syntax means that the first 8 bits of the prefix 10.0.0.0 must match. Additionally, the subnet mask is between 21 and 29 inclusive.

The above task states that prefixes with a subnet mask greater than /20 should not be accepted from AS 54. Therefore, zero bits of the actual prefix need to be checked. Instead, it must only be true that the subnet mask is less than or equal to /20. The syntax for this list is therefore as follows:

```
ip prefix-list SLASH_20_AND_UNDER seq 5 permit 0.0.0.0/0 le 20
```

Note

A prefix-list cannot be used to match on arbitrary bit patterns like an access-list can. Prefix-lists cannot be used to check if a number is even or odd, nor check if a number is divisible by 15, etc... Bit checking in a prefix-list is sequential, and must start with the most significant (leftmost) bit.

Task 2.5 Verification

Verify that we actually receive only /20 prefixes or shorter.

First verify the BGP configuration:

```
Rack1R6#show ip protocols | begin bgp
Routing Protocol is "bgp 100"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  IGP synchronization is disabled
  Automatic route summarization is disabled
  Neighbor(s):
    Address          FiltIn FiltOut DistIn DistOut Weight RouteMap
    54.1.3.254                                            SET_WEIGHT
    191.1.46.4
    204.12.1.3
    204.12.1.254                                          FROM_BB3
  Maximum path: 1
<output omitted>

Rack1R6#debug ip bgp updates
Rack1R6#clear ip bgp 204.12.1.254 soft in
BGP(0): 204.12.1.254 rcvd UPDATE w/ attr: nexthop 204.12.1.254, origin
i, metric 0, path 54
BGP(0): 204.12.1.254 rcvd 28.119.16.0/24 -- DENIED due to: route-map;
BGP(0): 204.12.1.254 rcvd 28.119.17.0/24 -- DENIED due to: route-map;
BGP(0): 204.12.1.254 rcv UPDATE w/ attr: nexthop 204.12.1.3, origin ?,
originator 0.0.0.0, path 54 100 200 254, community , extended community
BGP(0): 204.12.1.254 rcv UPDATE about 205.90.31.0/24 -- DENIED due to:
AS-PATH contains our own AS;
BGP(0): 204.12.1.254 rcv UPDATE about 220.20.3.0/24 -- DENIED due to:
AS-PATH contains our own AS;
BGP(0): 204.12.1.254 rcv UPDATE about 222.22.2.0/24 -- DENIED due to:
AS-PATH contains our own AS;
BGP(0): 204.12.1.254 rcvd UPDATE w/ attr: nexthop 204.12.1.254, origin
i, path 54
<output omitted>
```

Finally verify that the prefix-list has only one line:

```
Rack1R6#show running-config | include ip prefix-list
ip prefix-list SLASH_20_AND_UNDER seq 5 permit 0.0.0.0/0 le 20
```


Task 2.6

R3:

```
router bgp 200
 redistribute static
!
ip route 150.1.0.0 255.255.240.0 Null0
ip route 191.1.0.0 255.255.0.0 Null0
```

Task 2.6 Breakdown

There are four (previously three) ways to originate prefixes in BGP. The first is to use the **network** statement. Secondly, a route may be originated through the **redistribute** statement. Next, the **aggregate-address** command can originate a summary route based on more specific routes in the BGP table. A new method of BGP route generation is the **inject-map**, and will be covered in later scenarios.

By creating two static routes that point to Null0 and redistributing them into BGP, traffic that reaches R3 which is destined for a subset of these networks will only be forwarded if there is a more specific subnet installed in the IP routing table. Many protocols automatically generate a summary route to Null0 when aggregation is performed. This behavior is the desired behavior, and would rarely be modified for any practical reason.

Task 2.6 Verification

Verify that R3 originates the summaries:

```
Rack1R3#show ip bgp regexp ^$
BGP table version is 26, 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
*> 150.1.0.0/20     0.0.0.0                  0         32768 ?
*> 191.1.0.0        0.0.0.0                  0         32768 ?
```


Task 2.7

R6:

```
router bgp 100
 bgp dampening route-map DAMPENING
!
ip prefix-list AS54_CUSTOMERS seq 5 permit 112.0.0.0/8
ip prefix-list AS54_CUSTOMERS seq 10 permit 113.0.0.0/8
!
route-map DAMPENING permit 10
 match ip address prefix-list AS54_CUSTOMERS
 set dampening 15 750 2000 60
```

☞ **Quick Note**
Default values. Route-map requires values to be set.

Task 2.7 Breakdown

BGP route flap dampening (damping) is the process of suppressing consistently unstable routes from being used or advertised to BGP neighbors. Dampening is (and must be) used to minimize the amount of route recalculation performed in the global BGP table as a whole.

To understand dampening, the following terms must first be defined:

Penalty: Every time a route flaps, a penalty value of 1000 is added to the current penalty. All prefixes start with a penalty of zero.

Half-life: Configurable time it takes the penalty value to reduce by half. Defaults to 15 minutes.

Suppress Limit: Threshold at which a route is suppressed if the penalty exceeds. Defaults to 2000.

Reuse Limit: Threshold at which a suppressed route is unsuppressed if the penalty drops below. Defaults to 750.

Max Suppress: Maximum time a route can be suppressed if it has been stable. Defaults to four times the half-life value.

Each time a route flaps (leaves the BGP table and reappears), it is assigned a penalty of 1000. As soon as this occurs, the penalty of the route starts to decay based on the half-life timer. As the penalty increases, so does the rate of decay. For example, after a single flap, it will take 15 minutes for a prefix to reduce its penalty to 500.

Once the penalty of a prefix exceeds the suppress limit, the prefix is suppressed. A suppressed prefix cannot be used locally or advertised to any BGP peer. Once the penalty decay has resulted in the penalty decreasing below the reuse limit, the prefix is unsuppressed.

Lastly, the max-suppress timer dictates the maximum amount of time a prefix can be suppressed if it has been stable. This value is useful if a number of flaps have occurred in a short period of time, after which the route has been stable.

To enable BGP route flap dampening, simply enter the command **bgp dampening** under the BGP process.

Task 2.7 Verification

Verify the dampening parameters:

```
Rack1R6#show ip bgp dampening parameters
 dampening 15 750 2000 60 (route-map DAMPENING 10)
  Half-life time      : 15 mins       Decay Time       : 2320 secs
  Max suppress penalty: 12000         Max suppress time: 60 mins
  Suppress penalty    :  2000         Reuse penalty    : 750

Rack1R6#show route-map DAMPENING
route-map DAMPENING, permit, sequence 10
  Match clauses:
    ip address prefix-lists: AS54_CUSTOMERS
  Set clauses:
    dampening 15 750 2000 60
  Policy routing matches: 0 packets, 0 bytes
```

In general, you may want to do a final check to verify that you can reach your BGP networks from the various devices. Due to the summaries generated in task 3.4, and the earlier default route originated into OSPF by R3, you will have reachability to the BGP networks, even from non-BGP devices, such as the switches. For R6, a ping will need to be sourced from the loopback interface as shown below, since the interface networks are not being advertised into BGP.

```
foreach i {
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
205.90.31.1
222.22.2.1
} { ping $i source lo0 }
```


Task 3.1

R1:

```
ipv6 unicast-routing
!
interface Serial0/0
 ipv6 address 2001:CC1E:1:125::1/64
 frame-relay map ipv6 2001:CC1E:1:125::2 102 broadcast
 frame-relay map ipv6 2001:CC1E:1:125::5 105 broadcast
```

R2:

```
ipv6 unicast-routing
!
interface Serial0/0
 ipv6 address 2001:CC1E:1:125::2/64
 frame-relay map ipv6 2001:CC1E:1:125::1 201 broadcast
 frame-relay map ipv6 2001:CC1E:1:125::5 201
!
interface Serial0/1
 ipv6 address 2001:CC1E:1:23::2/64
```

R3:

```
ipv6 unicast-routing
!
interface Serial1/3
 ipv6 address 2001:CC1E:1:23::3/64
```


R5:

```
ipv6 unicast-routing
!
interface FastEthernet0/0
 ipv6 address 2001:CC1E:1:5::5/64
!
interface Serial0/0/0
 ipv6 address 2001:CC1E:1:125::5/64
 frame-relay map ipv6 2001:CC1E:1:125::1 501 broadcast
 frame-relay map ipv6 2001:CC1E:1:125::2 501
```

Task 3.1 Verification

```
Rack1R1#show ipv6 interface  brief
FastEthernet0/0            [administratively down/down]
Serial0/0                  [up/up]
    FE80::20D:BDFF:FEF6:2F80
    2001:CC1E:1:125::1
Serial0/1                  [up/up]
Loopback0                  [up/up]

Rack1R2#show ipv6 interface brief
FastEthernet0/0            [up/up]
Serial0/0                  [up/up]
    FE80::211:92FF:FE0E:5300
    2001:CC1E:1:125::2
Serial0/1                  [up/up]
    FE80::211:92FF:FE0E:5300
    2001:CC1E:1:23::2
Loopback0                  [up/up]

Rack1R3#show ipv6 interface brief
<output omitted>
Serial1/3                  [up/up]
    FE80::250:73FF:FE1C:7761
    2001:CC1E:1:23::3
<output omitted>
```



```
Rack1R3#ping ipv6 2001:CC1E:1:23::3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:CC1E:1:23::3, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/0/0 ms

Rack1R5#ping 2001:CC1E:1:125::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:CC1E:1:125::1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/61 ms

Rack1R5#ping 2001:CC1E:1:125::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:CC1E:1:125::2, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 160/167/185
ms
```

Task 3.2

R1:

```
interface Serial0/0
 ipv6 rip RIPng enable
 ipv6 address FE80::1 link-local
 frame-relay map ipv6 FE80::5 105
 frame-relay map ipv6 FE80::2 102
!
ipv6 router rip RIPng
 no split-horizon
```

R2:

```
interface Serial0/0
 ipv6 rip RIPng enable
 ipv6 address FE80::2 link-local
 frame-relay map ipv6 FE80::1 201
 frame-relay map ipv6 FE80::5 201
!
interface Serial0/1
ipv6 rip RIPng enable
!
```



```
R3:
interface FastEthernet0/0
 ipv6 rip RIPng enable
!
interface Serial1/3
 ipv6 rip RIPng enable
 ipv6 rip RIPng default-information only
!
!
interface Loopback100
 ipv6 address 2001:220:20:3::1/64
 ipv6 rip RIPng enable
!
interface Loopback100
 ipv6 address 2001:222:22:2::1/64
 ipv6 rip RIPng enable
!
interface Loopback100
 ipv6 address 2001:205:90:31::1/64
 ipv6 rip RIPng enable

R5:
interface FastEthernet0/0
 ipv6 rip RIPng enable
!
interface Serial0/0/0
 ipv6 rip RIPng enable
 ipv6 address FE80::5 link-local
 frame-relay map ipv6 FE80::1 501
 frame-relay map ipv6 FE80::2 501
```

Task 3.2 Breakdown

The above exercise demonstrates how to originate an IPv6 default route via RIPng with the interface level command **`ipv6 rip [process-id] default-information [originate | only]`**. When the *only* keyword is used, all other more specific networks are suppressed in RIPng advertisements on the interface. As seen in the below output, an IPv6 default route is expressed as the prefix ::/0.

```
Rack1R2#show ipv6 route rip
IPv6 Routing Table - 11 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
    O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
R   ::/0 [120/2]
     via FE80::250:73FF:FE1C:7761, Serial0/1
R   2001:CC1E:1:5::/64 [120/3]
     via FE80::204:27FF:FEB5:2F60, Serial0/0
```


Task 3.2 Verification

Verify that R3 sends only a default route to R2:

```
Rack1R2#show ipv6 route rip
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
R   ::/0 [120/2]
     via FE80::250:73FF:FE1C:7761, Serial0/1
R   2001:CC1E:1:5::/64 [120/3]
     via FE80::1, Serial0/0
```

Verify that R1 has split-horizon turned off for IPv6 RIPng:

```
Rack1R1#show ipv6 rip
RIP process "RIPng", port 521, multicast-group FF02::9, pid 133
     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 off; poison reverse is off
     Default routes are not generated
     Periodic updates 8, trigger updates 3
  Interfaces:
    Serial0/0
  Redistribution:
    None
```

Task 3.3

R4:

```
ipv6 unicast-routing
!
ipv6 router eigrp 100
 no shutdown
!
interface Loopback101
 ipv6 address 2001:CC1E:1:444::4/64
 ipv6 eigrp 100
!
interface Loopback102
 ipv6 address 2001:CC1E:1:454::4/64
 ipv6 eigrp 100
!
interface Loopback103
 ipv6 address 2001:CC1E:1:484::4/64
 ipv6 eigrp 100
!
interface FastEthernet 0/0.45
 ipv6 address 2001:CC1E:1:45::4/64
 ipv6 eigrp 100
 ipv6 summary-address eigrp 100 2001:CC1E:1:400::/55
```



```
R5:
ipv6 unicast-routing
!
ipv6 router eigrp 100
 no shutdown
 redistribute rip RIPng metric 1 1 1 1 1 include-connected
!
ipv6 router rip RIPng
 redistribute eigrp 100 metric 1
!
interface FastEthernet 0/1.45
 ipv6 address 2001:CC1E:1:45::5/64
 ipv6 eigrp 100
```

Task 3.3 Verification

```
Rack1R4#show ipv6 eigrp neighbors
IPv6-EIGRP neighbors for process 100
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   Link-local address:     Fa0/0.45          13 00:28:38    1   300  0
14
    FE80::21A:6DFF:FEDC:2655

Rack1R4#show ipv6 route eigrp
IPv6 Routing Table - Default - 14 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, M - MIPv6, R - RIP, I1 - ISIS L1
       I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
       EX - EIGRP external
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
EX  ::/0 [170/2560002816]
     via FE80::21A:6DFF:FEDC:2655, FastEthernet0/0.45
EX  2001:CC1E:1:5::/64 [170/2560002816]
     via FE80::21A:6DFF:FEDC:2655, FastEthernet0/0.45
EX  2001:CC1E:1:23::/64 [170/2560002816]
     via FE80::21A:6DFF:FEDC:2655, FastEthernet0/0.45
EX  2001:CC1E:1:125::/64 [170/2560002816]
     via FE80::21A:6DFF:FEDC:2655, FastEthernet0/0.45
D   2001:CC1E:1:400::/55 [5/128256]
     via Null0, directly connected

Rack1R5#show ipv6 route eigrp
IPv6 Routing Table - Default - 8 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, M - MIPv6, R - RIP, I1 - ISIS L1
       I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
       EX - EIGRP external
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF
ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
D   2001:CC1E:1:400::/55 [90/156160]
```



```
     via FE80::21E:7AFF:FE9E:417C, FastEthernet0/1.45
```

Task 4.1

R6:

```
interface Serial0/0/0
 ip access-group EIGRP_FROM_BB1_ONLY in
!
router eigrp 1
eigrp router-id 150.1.6.6
no auto-summary
 address-family ipv4 vrf VPN_A
 autonomous-system 10
 network 54.1.3.6 0.0.0.0
!
ip access-list extended EIGRP_FROM_BB1_ONLY
 permit eigrp host 54.1.3.254 any
 deny eigrp any any
 permit ip any any
```

Task 4.1 Verification

Verify the EIGRP neighbors and EIGRP routes:

```
Rack1R6#show ip eigrp vrf VPN_A neighbors
IP-EIGRP neighbors for process 10
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)
Cnt Num
0   54.1.3.254              Se0/0/0           11 20:45:20   25   200  0
140

Rack1R6#show ip route vrf VPN_A eigrp
D    200.0.0.0/24 [90/2297856] via 54.1.3.254, 00:03:16, Serial0/0/0
D    200.0.1.0/24 [90/2297856] via 54.1.3.254, 00:03:16, Serial0/0/0
D    200.0.2.0/24 [90/2297856] via 54.1.3.254, 00:03:16, Serial0/0/0
D    200.0.3.0/24 [90/2297856] via 54.1.3.254, 00:03:16, Serial0/0/0
```

Lastly, do a "debug ip packet" to check that we don't receive unnecessary EIGRP packets:

```
Rack1R6(config)#access-list 188 permit eigrp any any
Rack1R6#debug ip packet detail 188
  IP: s=54.1.1.254 (Serial0/0/0), d=224.0.0.10, len 100, access denied,
proto=88
  IP: s=54.1.10.254 (Serial0/0/0), d=224.0.0.10, len 60, access denied,
proto=88
  IP: s=54.1.3.254 (Serial0/0/0), d=224.0.0.10, len 60, rcvd 2,
proto=88
```

Task 4.2

R4:

```
router bgp 100
 address-family ipv4 vrf VPN_B
```



```
  network 150.1.44.44 mask 255.255.255.255
!
router ospf 100 vrf VPN_B
 area 90 sham-link 150.1.44.44 150.1.55.55
 redistribute bgp 100 subnets
!
interface FastEthernet 0/0.45
 mpls ip
!
router bgp 100
 address-family ipv4 vrf VPN_B
  redistribute ospf 100
```

R5:

```
router bgp 100
 address-family ipv4 vrf VPN_B
  network 150.1.55.55 mask 255.255.255.255
!
router ospf 100 vrf VPN_B
 area 90 sham-link 150.1.55.55 150.1.44.44
 redistribute bgp 100 subnets
!
interface FastEthernet 0/1.45
 mpls ip
!
router bgp 100
 address-family ipv4 vrf VPN_B
  redistribute ospf 100
```

SW3 & SW4:

```
interface Vlan 109
 ip ospf cost 9999
```

Task 4.2 Verification

```
Rack1SW3#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface
150.1.10.10       1   FULL/BDR        00:00:38    191.1.109.10
Vlan109
150.1.44.44       1   FULL/DR         00:00:35    191.1.49.4
Vlan49

Rack1SW3#show ip route vrf VPN_B

Routing Table: VPN_B
Codes: 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
```



```
Gateway of last resort is not set

     191.1.0.0/24 is subnetted, 3 subnets
O       191.1.50.0 [110/3] via 191.1.49.4, 00:04:18, Vlan49
C       191.1.49.0 is directly connected, Vlan49
C       191.1.109.0 is directly connected, Vlan109
     150.1.0.0/16 is variably subnetted, 4 subnets, 2 masks
O E2    150.1.55.55/32 [110/1] via 191.1.49.4, 00:04:18, Vlan49
O E2    150.1.44.44/32 [110/1] via 191.1.49.4, 00:04:18, Vlan49
O       150.1.10.10/32 [110/4] via 191.1.49.4, 00:04:18, Vlan49
C       150.1.9.0/24 is directly connected, Loopback0

Rack1SW3#traceroute vrf VPN_B 150.1.10.10

Type escape sequence to abort.
Tracing the route to 150.1.10.10

  1 191.1.49.4 9 msec 0 msec 0 msec
  2 191.1.50.5 0 msec 0 msec 0 msec
  3 191.1.50.10 0 msec *  0 msec

Rack1R4#show ip ospf sham-links
Sham Link OSPF_SL0 to address 150.1.55.55 is up
Area 90 source address 150.1.44.44
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 1 State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
    Hello due in 00:00:03
    Adjacency State FULL (Hello suppressed)
    Index 2/2, retransmission queue length 0, number of retransmission
0
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Task 5.1

R1:

```
interface Serial0/0
 ip pim neighbor-filter 1
!
access-list 1 deny 191.1.125.5
access-list 1 permit any
```

R5:

```
interface FastEthernet0/0
 ip pim dense-mode
 ip igmp helper-address 191.1.125.1
```

Task 5.1 Breakdown

This configuration is called multicast stub routing. With multicast stub routing, a stub router will not be allowed to become a PIM neighbor. This is accomplished by using the **ip pim neighbor-filter** interface command. The **ip pim**

neighbor-filter command takes an access-list as an option. The access-list should deny the IP addresses of the neighboring multicast devices that should not become PIM neighbors and permit all other IP addresses. Another option could be to use the reverse logic and permit only the IP addresses that are allowed to become PIM neighbors.

Since R5 will not form a PIM neighbor relationship, R5 will need to proxy for multicast clients connected to its FastEthernet0/0 interface by forwarding their IGMP host reports and IGMP leave messages to R1.

Task 5.1 Verification

Verify the PIM neighbors on R1 (note that R5 does not show up):

```
Rack1R1#show ip pim neighbor
PIM Neighbor Table
Neighbor      Interface          Uptime/Expires    Ver   DR
Address                                                  Prio/Mode
191.1.13.3    Serial0/1          00:11:03/00:01:30 v2    1 / S
```

At the same time, R5 sees R1 as PIM neighbor:

```
Rack1R5#show ip pim neighbor
PIM Neighbor Table
Neighbor     Interface          Uptime/Expires    Ver   DR
Address                                                  Prio/Mode
191.1.125.1   Serial0/0          00:11:27/00:01:36 v2    1 / S
```

Lastly verify that R5 could actually forward IGMP joins to R1:

```
Rack1R5#show ip igmp interface FastEthernet0/0 | inc help
  IGMP helper address is 191.1.125.1
```

Task 5.2

R3:

```
interface FastEthernet0/1
 ip igmp version 1
```

Task 5.2 Breakdown

The default IGMP version is 2. The Cisco IOS supports IGMP versions 1, 2, and 3. To change the IGMP version, the **ip igmp version** interface command is needed.

The basic difference between IGMP version 1 and IGMP version 2 is that IGMP version 2 incorporated an IGMP leave message to allow a host to notify the multicast router that it does not want to receive traffic for a particular multicast group. In IGMP version 1, there is not an explicit IGMP leave message. When a host wants to leave a multicast group, it just stops sending IGMP reports for the IGMP queries sent by the multicast router.

Note

There are three types of IGMP message that relate to multicast router and multicast client interaction.

1 = Host Membership Query
2 = Host Membership Report
3 = Leave Group

The IGMP query messages are sent by multicast enabled routers every 60 seconds (default) to all-hosts (224.0.0.1) in order to discover which multicast groups have hosts that would like to receive a particular multicast group.

The IGMP report messages are sent by hosts in response to IGMP queries reporting each multicast group to which they belong.

The IGMP leave messages are sent by hosts to notify a multicast router that it no longer wants to receive traffic for a particular multicast group. RFC 2236 (Internet Group Management Protocol, Version 2) states that the leave message is only mandatory if the host responded to the last IGMP query message for the group it wanted to leave. If the host was not the last to respond, RFC 2236 states that it is not mandatory to send an IGMP leave message.

Technically in IGMP version 2 there is a fourth message type, a version 1 membership report. This message is used for backward compatibility with IGMP version 1 clients.

Task 5.2 Verification

Verify the IGMP version on the interface:

```
Rack1R3#show ip igmp interface Fa0/1 | include version
  Current IGMP host version is 1
  Current IGMP router version is 1
```

Task 5.3

SW1:

```
interface Vlan7
 ip igmp static-group 225.25.25.25
```


Task 5.3 Verification

Verify that SW1 joined the multicast group:

```
Rack1SW1#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface            Uptime    Expires   Last Reporter
225.25.25.25     Vlan7                00:00:15  stopped   0.0.0.0
224.0.1.40       FastEthernet0/2      00:15:12  00:02:29  191.1.27.2
```

Next verify how this configuration affects the interface multicast switching:

```
Rack1SW1#show ip multicast interface vlan 7
Vlan7 is up, line protocol is up
  Internet address is 191.1.7.7/24
  Multicast routing: enabled
  Multicast switching: distributed
  Multicast packets in/out: 0/0
  Multicast boundary: not set
  Multicast TTL threshold: 0
  Multicast Tagswitching: disabled
```

Task 6.1

SW1 and SW2:

```
vlan access-map NO_DEC-SPANNING 10
 action drop
 match mac address DEC-SPANNING
!
vlan access-map NO_DEC-SPANNING 20
 action forward
!
vlan filter NO_DEC-SPANNING vlan-list 363
!
mac access-list extended DEC-SPANNING
 permit any any dec-spanning
```

Task 6.1 Breakdown

This section is requiring a VACL to be configured within VLAN 363 that filters off any DECnet spanning tree BPDUs.

Ensure that there is an additional **`vlan access-map`** that forwards all other traffic or at least all other DECnet traffic. If this is not added, all DECnet traffic would be denied. The logic of the VACL is that as long as you do not deny a certain protocol or all protocols, it will not be affected by the VACL.

Task 6.1 Verification

Verify the VLAN filter configuration:

```
Rack1SW2#show vlan access-map  NO_DEC-SPANNING
Vlan access-map "NO_DEC-SPANNING"  10
  Match clauses:
    mac address: DEC-SPANNING
  Action:
    drop
Vlan access-map "NO_DEC-SPANNING"  20
  Match clauses:
  Action:
    forward

Rack1SW2#show vlan filter access-map  NO_DEC-SPANNING
VLAN Map NO_DEC-SPANNING is filtering VLANs:
  363
```

Task 6.2

R2:

```
username CLI password 0 CISCO
username TELNET password 0 CISCO
username TELNET autocommand access-enable timeout 5
!
interface Serial0/0
 ip access-group DYNAMIC in
!
interface Serial0/1
 ip access-group DYNAMIC in
!
ip access-list extended DYNAMIC1
 dynamic PERMIT_TELNET permit tcp any any eq telnet
 deny   tcp any host 191.1.27.7 eq telnet
 deny   tcp any host 191.1.7.7 eq telnet
 deny   tcp any host 191.1.77.7 eq telnet
 deny   tcp any host 191.1.177.7 eq telnet
 deny   tcp any host 150.1.7.7 eq telnet
 permit ip any any
!
line vty 0 4
 login local
```


Task 6.2 Verification

To verify the dynamic ACL, first telnet to SW1 without the previous authentication on R2:

```
Rack1R3#telnet 150.1.7.7
Trying 150.1.7.7 ...
% Destination unreachable; gateway or host down
```

Next telnet and login to R2:

```
Rack1R3#telnet 150.1.2.2
Trying 150.1.2.2 ... Open

User Access Verification

Username: TELNET
Password:
[Connection to 150.1.2.2 closed by foreign host]
```

Verify the dynamic ACL on R2:

```
Rack1R2#show ip access-lists
Extended IP access list DYNAMIC
    10 Dynamic PERMIT_TELNET permit tcp any any eq telnet
       permit tcp any any eq telnet (4 matches) (time left 275)
    20 deny tcp any host 191.1.27.7 eq telnet
    30 deny tcp any host 191.1.7.7 eq telnet
    40 deny tcp any host 191.1.77.7 eq telnet
    50 deny tcp any host 191.1.177.7 eq telnet
    60 deny tcp any host 150.1.7.7 eq telnet (2 matches)
    70 permit ip any any (184 matches)
```

Finally telnet to SW1:

```
Rack1R3#telnet 150.1.7.7
Trying 150.1.7.7 ... Open


User Access Verification

Password:
Rack1SW1>
```

Task 7.1

R3:

```
access-list 25 permit 191.1.7.100
access-list 25 permit 191.1.77.100
access-list 50 permit 191.1.7.100
!
snmp-server community CISCORO RO 25
snmp-server community CISCORW RW 50
snmp-server system-shutdown
snmp-server host 191.1.7.100 CISCOTRAP
snmp-server host 191.1.77.100 CISCOTRAP
snmp-server enable traps
```


Task 7.1 Breakdown

Although this section does not explicitly state that SNMP traps need to be enabled, the wording of the task indicated that not only should the community be set to CISCOTRAP but SNMP traps should be enabled. To enable SNMP traps the **snmp-server enable traps** command was configured.

To allow a device to be reloaded via SNMP, the **snmp-server system-shutdown** will need to be configured. Technically, the device will not be shutdown, but will be reloaded. The network management station will also need RW access via SNMP to reload the device. This is why the first network management station was given RW access in this section.

Task 7.1 Verification

Verify that the SNMP traps are configured:

```
Rack1R3#show snmp | begin logging
SNMP logging: enabled
    Logging to 191.1.7.100.162, 0/10, 0 sent, 0 dropped.
    Logging to 191.1.77.100.162, 0/10, 0 sent, 0 dropped.
```

Make sure that system shutdown via SNMP is enabled:

```
Rack1R3#show running-config | include snmp.*shut
snmp-server system-shutdown
```

Task 7.2

R1:

```
logging 191.1.7.100
!
rmon event 1 log
rmon alarm 1 ifEntry.10.3 60 delta rising-threshold 80000 1 falling-
threshold 40000 1
```

R3:

```
logging 191.1.7.100
!
rmon event 1 log
rmon alarm 1 ifEntry.10.5 60 delta rising-threshold 80000 1 falling-
threshold 40000 1
```


Task 7.2 Breakdown

The key to this section is the reference to the word 'average'. RMON can monitor two values, absolute or delta. The absolute value is the value since the last reload of a device or resetting (if available) of the value's counters. Delta on the other hand is monitoring the rate of change in a value.

Certain values like CPU utilization are normally monitored for the absolute value and not the delta value. It would be more useful to know when the one minute CPU utilization rises above 75% (absolute), than it is when the one minute CPU utilization changes 10% in value (delta). Input or output interface values (i.e. input octets) normally are monitored for their rate of change. This is done by taking the delta value. In this section, a log message will be generated whenever the delta values rise above 80000 or falls below 40000.

Task 7.2 Verification

Verify the RMON configuration:

```
Rack1R3#show rmon alarms
Alarm 1 is active, owned by config
 Monitors ifInOctets.5 every 60 second(s)
 Taking delta samples, last value was 0
 Rising threshold is 840000, assigned to event 1
 Falling threshold is 40000, assigned to event 1
 On startup enable rising or falling alarm
```



```
Rack1R3#show rmon events
Event 1 is active, owned by config
Description is
Event firing causes log,
last event fired at  0y0w0d,00:00:00,
Current uptime       0y0w0d,07:49:51
```

After configuring, you may see a message on the console at the next polling interval, since your value is below the falling-threshold value.

```
Rack1R3#
Mar  3 14:26:21.297: %RMON-5-FALLINGTRAP: Falling trap is generated
because the value of ifInOctets.5 has fallen below the falling-
threshold value 40000
Rack1R3#
```

In this particular task, we were given the explicit information to use for the interface. If it is necessary to look up the interface, you can use the **show snmp mib ifmib ifindex** command to see the index numbers assigned to your interfaces.

```
Rack1R1#show snmp mib ifmib ifindex
FastEthernet0/0: Ifindex = 1
Loopback0: Ifindex = 6
Null0: Ifindex = 5
Serial0/0: Ifindex = 2
VoIP-Null0: Ifindex = 4
Serial0/1: Ifindex = 3
Rack1R1#
```

Task 7.3

R4:

```
cdp source-interface Loopback0
cdp timer 5
cdp holdtime 15
```

SW2:

```
cdp timer 5
cdp holdtime 15
```


Task 7.3 Breakdown

Cisco Discovery Protocol is a media and protocol independent layer 2 protocol. CDP advertisements include useful information such as device type, device name, and local and remote interface connections. CDP can also be used to transport routing information when used with On Demand Routing (ODR).

CDP is enabled on all Cisco devices by default, and can be globally disabled with the **no cdp run** command, or disabled on a per interface basis with the **no cdp enable** interface level command.

CDP advertisement intervals are controlled by the global configuration commands **cdp timer** and **cdp holdtime**. The **cdp source-interface** command can be used to modify which IP address information is included with CDP advertisements.

Task 7.3 Verification

```
Rack1R4#show cdp
Global CDP information:
        Sending CDP packets every 5 seconds
        Sending a holdtime value of 15 seconds
        Sending CDPv2 advertisements is  enabled
        Source interface is Loopback0
```

Task 7.4

SW2:

```
service udp-small-servers
!
interface FastEthernet0/18
 ip access-group 100 in
!
access-list 100 deny   udp any any eq discard
access-list 100 deny   udp any any eq 19
access-list 100 permit ip any any
```

Task 7.4 Breakdown

TCP and UDP small servers are simple diagnostic utilities for testing network reachability. These services include echo, chargen, discard, and daytime for TCP, and echo, chargen, and discard for UDP. Typically, these services are disabled in order to avoid various security vulnerabilities that are associated with them. To enable these services, issue the **service tcp-small-servers** or **service udp-small-servers** global configuration commands.

Task 7.4 Verification

Verify that the chargen/discard ports are reachable. Traceroute can be used to generate traffic with a UDP port destination.

```
Rack1R4#traceroute
Protocol [ip]:
Target IP address: 191.1.48.8
Source address:
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]: 9
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 191.1.48.8

  1 191.1.48.8 [AS 200] !A  *  0 msec
Rack1R4#
```

After testing with both ports 9 and 19, verify the ACL:

```
Rack1SW2#show ip access-lists 100
Rack1SW2(config)#do show access-list 100
Extended IP access list 100
    10 deny udp any any eq discard (1 match)
    20 deny udp any any eq 19 (1 matches)
    30 permit ip any any (747 matches)
```

Finally, verify that UDP small-services are enabled:

```
Rack1SW2#show running-config | include small
service udp-small-servers
```

Task 8.1

R3 and R4:

```
class-map match-all RTP
 match protocol rtp
!
policy-map QOS
 class RTP
  priority percent 25
```

R3:

```
interface Serial1/0
 service-policy output QOS
```

R4:

```
interface Serial0/0
 service-policy output QOS
```


Task 8.1 Breakdown

This type of priority queueing is known as Low Latency Queueing. Unlike the legacy priority-list, LLQ can prioritize traffic, while at the same time ensure that other traffic gets serviced. In the legacy priority queue, all packets in the upper queues are serviced before lower queues are checked for packets. This can, and does, result in packets in the lower queues being starved of bandwidth. The LLQ prevents this case by setting a maximum bandwidth threshold for which traffic will be prioritized.

The above MQC configuration dictates that RTP packets will always be dequeued first out the Frame Relay connections of R3 and R4 up to 25% of the bandwidth. When RTP traffic exceeds 25% of the output queue, the excess of 25% does not receive low latency. In the case that there is congestion on the link, traffic in excess of this 25% may be dropped.

Prior to IOS 12.2, the **bandwidth percent** and the **priority percent** commands were *relative* reservations based on what the available bandwidth of the interface. In newer IOS releases, these reservations are *absolute* reservations. The difference between these reservations can be seen as follows.

Caution

The bandwidth value that this percentage reservation is based off of is the configured bandwidth value of the interface. For a practical implementation, the **bandwidth** value of the interface should be modified to reflect the provisioned rate of the layer 2 circuit.

The *available bandwidth* of an interface is calculated as:

```
Available_Bandwidth = (Configured_Bandwidth * max-reserved-
bandwidth/100) - (LLQ - RTP - RSVP)
```

Where *Configured_Bandwidth* is the bandwidth value of the interface as specified by the **bandwidth** command, and where *max-reserved-bandwidth* is the configured **max-reserved-bandwidth** of the interface (defaults to 75%). This reservable value is put into place to ensure that necessary network traffic (layer 2 keepalives, layer 3 routing) gets the service that it requires.

To see what the available bandwidth of an interface is, issue the **show queue [interface]** command:

```
Rack1R3#show queue Fa0/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
```



```
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 75000 kilobits/sec
```

From the above output, it is evident that this interface is a 100Mbps FastEthernet interface (default configured bandwidth value of 100Mbps). The available bandwidth is 75000Kbps, which is 75% of the default interface bandwidth of 100Mbps. This above router is running 12.2(15)T5, in which a reservation is always *absolute*. The following demonstrates so:

```
ip cef
!
class-map match-all FTP
  match protocol ftp
!
 policy-map QOS
  class FTP
   bandwidth percent 25
!
interface FastEthernet0/0
 service-policy output QOS

Rack1R1#show queue Fa0/0 | in Available
     Available Bandwidth 50000 kilobits/sec
```

Notice from the above output that the available bandwidth value just decreased by 25 Mbps, or 25% of 100Mbps. This is an *absolute* reservation. This has the same effect as if the **`bandwidth percent 25`** statement actually **`said bandwidth 25000`**, as seen as follows:

```
policy-map QOS
  class FTP
   bandwidth 25000

Rack1R3#show queue Fa0/0 | include Available
     Available Bandwidth 50000 kilobits/sec
```

Notice the same output. This is still an *absolute* reservation. In **older** IOS releases, percentage reservations were *relative*, as follows:

```
Rack1R3#show queue Fa0/0 | include Available
     Available Bandwidth 75000 kilobits/sec
```

Here, we see the same FastEthernet interface with no prior reservations.
As max-reserved-bandwidth is 75 by default there is an available bandwidth of 75Mbps. Now apply the same configuration as before:


```
class-map match-all FTP
  match protocol ftp
!
policy-map QOS
  class FTP
   bandwidth percent 50
!
interface FastEthernet0/0
 service-policy output QOS
!

Rack1R3#show queue Fa0/0 | include Available
     Available Bandwidth 75000 kilobits/sec
```

Although 50% of the bandwidth on this interface is reserved for FTP, it is a relative reservation of what is available. Since the available bandwidth on the interface is 7.5Mbps, FTP is effectively guaranteed a minimum of 37.5Mbps (50% of 75% of 10Mbps). In order to actually reserve 50 Mbps for FTP in this case there are three options.

1. Set 'max-reserved-bandwidth' to 100

```
interface FastEthernet0/0
 max-reserved-bandwidth 100
 service-policy output QOS

Rack1R3#show queue Fa0/0 | include Available
     Available Bandwidth 100000 kilobits/sec
```

Since 100 Mbps is now available on the interface, FTP is guaranteed 50 Mbps (50% of 100 Mbps). This method should be used with caution, as reserving too much of the output queue of an interface can result in delay or loss of necessary layer 2 and layer 3 network control packets.

2. Do an absolute **`bandwidth [kbps]`** reservation

```
class-map match-all FTP
  match protocol ftp
!
policy-map QOS
  class FTP
   bandwidth 50000
!
interface FastEthernet0/0
 service-policy output QOS
```



```
Rack1R3#show queue Fa0/0 | include Available
     Available Bandwidth 25000 kilobits/sec
```

bandwidth [kbps] and **priority [kbps]** are always absolute reservations regardless of the IOS version, and are not based on the available bandwidth of the interface. It is evident that after configuring **bandwidth 50000** under the FTP class, only 25 Mbps is now available on the interface.

3. Change the configured **bandwidth** value on the interface

While not very practical, the bandwidth value on the interface can be adjusted so that the following would be true:

```
Interface_bandwidth = configured_bandwidth * max-reserved-bandwidth/100

Configured_bandwidth = interface_bandwidth * 100/max-reserved-bandwidth

interface FastEthernet0/0
 bandwidth 133334
 service-policy output QOS

Rack1R3#show queue Fa0/0 | include Available
     Available Bandwidth 100000 kilobits/sec
```

While the third option is a roundabout solution, the point of the exercise is to show that the available bandwidth is based on the configured **bandwidth** keyword, and not a function of the physical interface.

Task 8.1 Verification

Verify the policy-map configuration:

```
Rack1R4#show policy-map interface s0/0

 Serial0/0

  Service-policy output: QOS

    Class-map: RTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol rtp
      Queueing
        Strict Priority
        Output Queue: Conversation 264
        Bandwidth 25 (%)
        Bandwidth 386 (kbps) Burst 9650 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      10 packets, 667 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
```



```
      Match: any
```

Task 8.2

R3 and R4:

```
class-map match-all NOT_HTTP
 match not protocol http
!
policy-map QOS
  class NOT_HTTP
  class class-default
   fair-queue
   random-detect
```

Task 8.2 Breakdown

The above exercise is designed to show the usage of the **match not** keyword in the class-map, and to illustrate how random early detection works within the modular quality of service. To configure WRED in the MQC, one of two conditions must be met. There must either be a **bandwidth** reservation made within a class, or the default-class must be running weighted fair queuing.

As the above task states that HTTP traffic should not be reserved any bandwidth, the only way to accomplish this task is to remove all non-HTTP traffic from the default class, and run WRED on the default class in which only HTTP remains.

Task 8.2 Verification

Verify the new policy-map configuration (note WRED configured in class-default):

```
Rack1R3#show policy-map interface s1/0
 Serial1/0

  Service-policy output: QOS

    Class-map: RTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol rtp
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 25 (%)
        Bandwidth 32 (kbps) Burst 800 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: NOT_HTTP (match-all)
      22 packets, 1592 bytes
      5 minute offered rate 0 bps
      Match: not protocol http
```



```
    Class-map: class-default (match-any)
      166 packets, 11960 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 0/0/0
         exponential weight: 9

  class    Transmitted  Random drop  Tail drop Minimum Maximum  Mark
           pkts/bytes   pkts/bytes   pkts/bytes thresh  thresh  prob
      0       0/0          0/0          0/0          20      40  1/10
      1       0/0          0/0          0/0          22      40  1/10
<output omitted>
```

Task 8.3

R1:

```
interface Serial0/1
 ip tcp header-compression
 ip tcp compression-connections 64
```

R3:

```
interface Serial1/2
 ip tcp header-compression
 ip tcp compression-connections 64
```

Task 8.3 Verification

```
Rack1R3#show ip tcp header-compression
TCP/IP header compression statistics:
  Interface Serial1/2 (compression on, VJ)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 64 rx slots, 64 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 64 free
contexts
```

