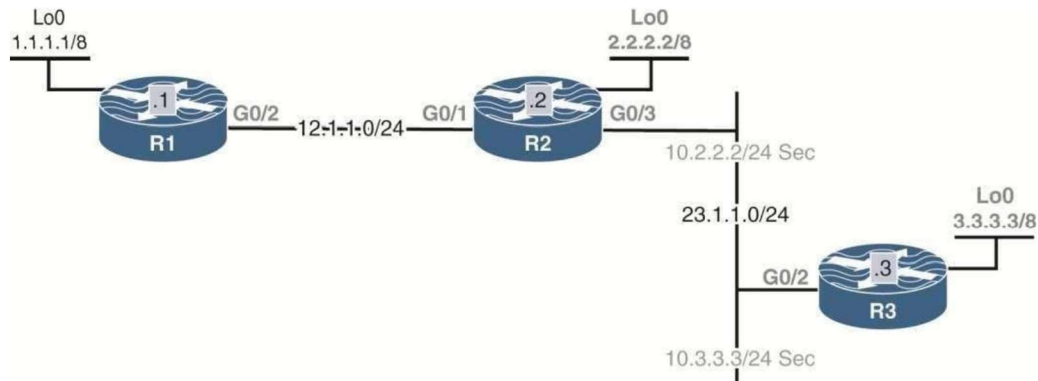


OSPF Mastery Workbook

Lab 1

Running OSPF on the Interfaces



Lab Setup:

To copy and paste the initial configurations, go to the Initial-config folder ☐ OSPF folder ☐ Lab-1.

All Labs Setup Note:

The initial configurations have already been applied for your convenience. If you need to review, copy, or reapply them at any point during the lab, you can find all device configurations in the **Initial-config** folder.

Task 1

Configure OSPF Area 0 on all directly connected interfaces in the previous topology, including the secondary IP addresses.

- Do not use the **network** command to accomplish this task.
- The loopback interfaces should be advertised with their correct mask.
- If this configuration is performed successfully, each router will have reachability to all routes within the previous topology.
- The OSPF RID should be configured based on the Lo0 interface of these three routers.

This task focuses on basic OSPF neighbor configuration and how OSPF advertises prefixes attached to interfaces.

The task first specifies that all routers participate in OSPF and belong to area 0. They should be configured to advertise their directly connected interface IP addresses into OSPF, including the secondary IP addresses configured on R2 and R3.

OSPF can be enabled on an interface using one of the two methods:

1. Using the **network ip-address wildcard-mask area area-id** command under the OSPF router configuration. The command also includes assigning the interfaces covered with the network range to a specified area, with the area keyword
2. Directly on the interface with the **ip ospf process-id area area-id [secondaries none]** command

The first configuration style utilizes the familiar network address wildcard area area-number command. It uses an address/wildcard configuration to specify a range of IP addresses OSPF should advertise. Any interface that falls within the range specified by the net command will be enabled to run the configured OSPF process and the IP prefix assigned to that interface will be advertised into OSP. The command also specifies to what area the matching IP address and interface is associated. OSPF design utilizes the concept of areas to define regions of the network where topology information is synchronized.

The second configuration style directly configures the interface to participate in a specific OSPF process number and area. This configuration style will automatically create the indicated OSPF process and assign the interface to the indicated area. It also automatically advertises all IP addresses associated with the interface into the configured OSPF process and area.

When configuring OSPF using either of the two methods above some things need to be noted:

1. When configured using the **interface-level** configuration style, secondary addresses are automatically advertised into OSPF.
2. When configured using the **network** command configuration style, secondary addresses must be matched by a **network** command statement that includes the secondary IP address of the interface in its address/wildcard mask combination. Also, if an interface is only matched by a **network** command matching its secondary IP address, OSPF will not be enabled on that interface.

Breaking down the two points above, when configured using interface-level configuration, the router pretty much takes care of everything. It creates the OSPF process and adds all IP addresses, including secondary addresses, to be advertised into OSPF in the indicated area. OSPF is enabled on the interface and will begin discovering OSPF neighbors, if any exist, connected to that interface.

The **network** command is more selective. It will only advertise the networks explicitly matched into OSPF. If the goal is to advertise both the primary and secondary interfaces into OSPF then the **network statement**

utilized should include both addresses or separate **network** command statements for each address is required. In addition, if only an interface's secondary IP address is matched by a **network** command, that interface is not added into the OSPF process nor is the secondary IP address advertised into the OSPF domain.

To complete the first requirement of the task, OSPF needs to be enabled on all routers. Every interface on the router should be configured to be in area 0 and all secondary addresses should be advertised as well. The task explicitly forbids the use of the **network** command, leaving the **interface-level** configuration style as the only other option. The following configuration enables OSPF for all directly connected and loopback 0 interfaces on R1, R2, and R3:

On R1:

```
R1 (config) #interface g0/2
R1 (config-if) #ip ospf 1 area 0
```

```
R1 (config) #interface lo0
R1 (config-if) #ip ospf 1 area 0
```

On R2:

```
R2 (config) #interface lo0
R2 (config-if) #ip ospf network point-to-point
R2 (config-if) #ip ospf 1 area 0
```

```
R2 (config) #interface g0/1
R2 (config-if) #ip ospf 1 area 0
```

```
R2 (config) #interface g0/3
R2 (config-if) #ip ospf 1 area 0
```

```
R2 (config) #router ospf 1
R2 (config-router) #router-id 2.2.2.2
```

You should see the following console message:

```
%OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on GigabitEthernet0/1 from
LOADING to FULL, Loading Done
```

On R3:

```
R3 (config) #interface lo0
R3 (config-if) #ip ospf 1 area 0
R3 (config-if) #ip ospf network point-to-point
```

```
R3(config)#interface g0/2
R3(config-if)#ip ospf 1 area 0

R3(config)#router ospf 1
R3(config-router)#router-id 3.3.3.3
```

You should see the following console message:

```
%OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on GigabitEthernet0/2 from
LOADING to FULL, Loading Done
```

NOTE: It may take around 40 seconds for the UP notification to be displayed.

To verify the configuration:

The **show ip ospf interface brief** command output can then be used to verify the interfaces enabled for OSPF:

On R1:

```
R1#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	1.1.1.1/8	1	P2P	0/0	
Gi0/2	1	0	12.1.1.1/24	1	DR	0/0	

On R2:

```
R2#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	2.2.2.2/8	1	LOOP	0/0	
Gi0/3	1	0	23.1.1.2/24	1	DR	0/0	
Gi0/1	1	0	12.1.1.2/24	1	BDR	1/1	

On R3:

```
R3#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	3.3.3.3/8	1	LOOP	0/0	
Gi0/2	1	0	23.1.1.3/24	1	BDR	1/1	

The **show ip route ospf | begin Gate** command on R1, R2 and R3 confirms which OSPF routes the routers have installed into their routing tables:

On R1:

```
R1#show ip route ospf | begin Gate
```

```
Gateway of last resort is not set
```

```

    2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/2] via 12.1.1.2, 00:05:28, GigabitEthernet0/2
    3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/3] via 12.1.1.2, 00:02:01, GigabitEthernet0/2
    10.0.0.0/24 is subnetted, 2 subnets
O       10.2.2.0 [110/2] via 12.1.1.2, 00:05:38, GigabitEthernet0/2
O       10.3.3.0 [110/3] via 12.1.1.2, 00:02:01, GigabitEthernet0/2
    23.0.0.0/24 is subnetted, 1 subnets
O       23.1.1.0 [110/2] via 12.1.1.2, 00:05:38, GigabitEthernet0/2

```

On R2:

```
R2#show ip route ospf | begin Gate
```

```
Gateway of last resort is not set
```

```

    1.0.0.0/32 is subnetted, 1 subnets
O       1.1.1.1 [110/2] via 12.1.1.1, 00:13:17, GigabitEthernet0/1
    3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/2] via 23.1.1.3, 00:09:35, GigabitEthernet0/3
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O       10.3.3.0/24 [110/2] via 23.1.1.3, 00:09:35, GigabitEthernet0/3

```

On R3:

```
R3#show ip route ospf | begin Gate
```

```
Gateway of last resort is not set
```

```

    1.0.0.0/32 is subnetted, 1 subnets
O       1.1.1.1 [110/3] via 23.1.1.2, 02:34:05, GigabitEthernet0/2
    2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/2] via 23.1.1.2, 02:34:05, GigabitEthernet0/2
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O       10.2.2.0/24 [110/2] via 23.1.1.2, 02:34:05, GigabitEthernet0/2
    12.0.0.0/24 is subnetted, 1 subnets
O       12.1.1.0 [110/2] via 23.1.1.2, 02:34:05, GigabitEthernet0/2

```

Observing the routing tables above, we notice that secondary IP addresses configured on R2's G0/3 interface and the G0/2 interface on R3 are also advertised into OSPF. This is a direct result of the interface-level **ip ospf process area area-id** command configuration as mentioned earlier.

The **show ip ospf database router self-originate** command output can be used to check the router LSAs generated by a router. This command lists the Type-1 router LSA that describes an individual router and its attached interfaces that are enabled for OSPF. Each interface is described as a link in the OSPF database and is given a metric value. These links are identified as being connected to one of the following type of networks:

1. Point-to-point
2. Transit
3. Stub

Point-to-point networks indicate a link that is connected to a single OSPF-speaking device. **Transit networks** are networks that can carry traffic that was not locally generated or destined and contain many OSPF-speaking devices. Finally, a **stub network** indicates a network where there are no other OSPF-speaking devices and only traffic destined to or from the network will flow over the link to that network.

The metric value in OSPF can be manually set on a per-interface basis or precalculated by IOS. By default, IOS will use a reference bandwidth value that defaults as 100,000Kbps which equals a 100Mbps FastEthernet connection. Interface cost values are reported as ratios of this bandwidth using the following formula:

reference-bandwidth/interface-bandwidth.

The following is an example of the router LSA generated by R2 and R3:

On R2:

R2#**show ip ospf database router self-originate**

OSPF Router with ID (2.2.2.2) (Process ID 1)

Router Link States (Area 0)

```
LS age: 1283
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 2.2.2.2
Advertising Router: 2.2.2.2
LS Seq Number: 8000000A
Checksum: 0x4735
```

Length: 72
 Number of Links: 4

Link connected to: a Stub Network
 (Link ID) Network/subnet number: 2.2.2.2
 (Link Data) Network Mask: 255.255.255.255

Number of MTID metrics: 0
 TOS 0 Metrics: 1

Link connected to: a Transit Network
 (Link ID) Designated Router address: 23.1.1.2
 (Link Data) Router Interface address: 23.1.1.2
 Number of MTID metrics: 0
 TOS 0 Metrics: 1

Link connected to: a Stub Network
 (Link ID) Network/subnet number: 10.2.2.0
 (Link Data) Network Mask: 255.255.255.0
 Number of MTID metrics: 0
 TOS 0 Metrics: 1

Link connected to: a Transit Network
 (Link ID) Designated Router address: 12.1.1.1
 (Link Data) Router Interface address: 12.1.1.2
 Number of MTID metrics: 0
 TOS 0 Metrics: 1

On R3:

R3#show ip ospf database router self-originate

OSPF Router with ID (3.3.3.3) (Process ID 1)

Router Link States (Area 0)

LS age: 1460
 Options: (No TOS-capability, DC)
 LS Type: Router Links
 Link State ID: 3.3.3.3
 Advertising Router: 3.3.3.3
 LS Seq Number: 80000007
 Checksum: 0x3A65
 Length: 60
 Number of Links: 3

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 3.3.3.3
(Link Data) Network Mask: 255.255.255.255
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

```
Link connected to: a Transit Network
(Link ID) Designated Router address: 23.1.1.2
(Link Data) Router Interface address: 23.1.1.3
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 10.3.3.0
(Link Data) Network Mask: 255.255.255.0
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

The output above lists an array of links connecting to different types of networks. Most notably, highlighted ingreen, is OSPF's interpretation of the link connecting to the secondary IP address of R2's G0/3 interface. It is described as being connected to a stub interface. In OSPF, stub networks are typically used to describe IP addresses assigned to point-to-point and loopback interfaces. This allows OSPF to solve the SPF algorithm required to reach the network.

Next, highlighted is the metric assigned to each link. This number was automatically assigned to the link based on the IOS default OSPF bandwidth calculation. The calculation takes the reference bandwidth of 100,000Kbps (100Mbps) and divides it by the interface bandwidth of, for example, R2's G0/3 interface, which is 1,000,000 Kbps (1,000Mbps or 1Gbps) as seen below:

On R2:

```
R2#show interface g0/3 | include BW
```

```
MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
```

The result is $100,000/1,000,000$ which equals 0.1. OSPF rounds this fraction to 1 which is the value seen as the metric value for the links.

Also, highlighted is the OSPF router ID configured on R2 and R3. Router IDs are a 32-bit number that uniquely identify the device running OSPF. By default, router-ID on a device is selected in the following manner:

1. Manually configured router-id with the **router-id** command.

2. If no manual configuration exists, then, select the highest IP address of an active loopback interface.
3. If no loopback exists, then, select the highest IP address of an active physical interface

This task requires that R1, R2 and R3 use the IP address assigned to their loopback 0 interfaces as the OSPF router IDs. As seen in the output above, these loopback interfaces are already being used as the OSPF router IDs, since they are the only loopbacks configured on these devices. However, should another loopback interface be configured with a higher IP address followed by a restart of the OSPF process, the router ID would change to a different loopback address. To ensure the IP assigned to the loopback 0 address is always used as the router-id, the **router-id** command under the OSPF router configuration mode is used to manually assign the router IDs. The configuration for this is seen below on R1, R2 and R3:

On R1:

```
R1(config)#router ospf 1
R1(config-router)#router-id 1.1.1.1
```

On R2:

```
R2(config)#router ospf 1
R2(config-router)#router-id 2.2.2.2
```

On R3:

```
R3(config)#router ospf 1
R3(config-router)#router-id 3.3.3.3
```

Note: Since the manually set router IDs above match the existing router ID on the routers, IOS does require the OSPF process to be restarted. If the manually set router ID did not match the existing router ID, IOS would prompt to restart the OSPF process for the new router ID to take effect.

On the subject of loopback interfaces, the task makes special mention that all loopbacks enabled for OSPF should be advertised with their actual masks. The **show run interface loopback 0** command output shows a /8 mask used for the loopback 0 addresses on R1, R2 and R3

On R1:

```
R1#show run interface lo0 | begin interface
```

```
interface Loopback0
 ip address 1.1.1.1 255.0.0.0
 ip ospf 1 area 0
```

On R2:

```
R2#show run interface lo0 | begin interface
```

```
interface Loopback0
ip address 2.2.2.2 255.0.0.0
ip ospf 1 area 0
```

On R3:

```
R3#show run int lo0 | begin interface
```

```
interface Loopback0
ip address 3.3.3.3 255.0.0.0
ip ospf 1 area 0
```

The router LSAs on each router tell a different story as shown in the below:

On R1:

```
R1#show ip ospf database router self-originate | section Stub
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 1.1.1.1
(Link Data) Network Mask: 255.255.255.255
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

On R2:

```
R2#show ip ospf database router self-originate | section Stub
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 2.2.2.2
(Link Data) Network Mask: 255.255.255.255
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

(The rest of the output is omitted for brevity)

On R3:

```
R3#show ip ospf database router self-originate | section Stub
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 3.3.3.3
(Link Data) Network Mask: 255.255.255.255
Number of MTID metrics: 0
```

TOS 0 Metrics: 1

(The rest of the output is omitted for brevity)

The highlighted portion of the output points out the fact that each router is advertising its loopback 0 interface into OSPF with /32 mask. The reason for this is the OSPF network type assigned by default to the loopback interfaces is **loopback**. According to the OSPF RFC (2328) loopback interfaces should be advertised as /32 host routes because they represent addresses attached to interfaces that are physically looped back to the device in hardware or logically looped back to the device in software. Since the interface is looped back, traffic will not be transited to such addresses but reachability to the address may be needed to perform testing. Because of this, these interfaces are considered to be stub interfaces much like the secondary addresses described above and are advertised as /32 host routes.

Following excerpt from RFC 2328, section 9.1 highlights the above:

In this state, the router's interface to the network is looped back. The interface may be looped back in hardware or software. The interface will be unavailable for data traffic. However, it may still be desirable to gain information on the quality of this interface, either through sending ICMP pings to the interface or through something like a bit error test. For this reason, IP packets may still be addressed to an interface in Loopback state. **To facilitate this, such interfaces are advertised in router-LSAs as single host routes**, whose destination is the IP interface address.[4]

The default behavior can be changed. To allow OSPF to advertise the IP addresses on the loopback interface with their actual mask, the network type on the loopback interface needs to be changed to be point-to-point with the **ip ospf network point-to-point** command under the loopback interface. The following configures this on R1, R2 and R3:

On R1, R2, and R3:

```
Rx(config)#interface lo0
Rx(config-if)#ip ospf network point-to-point
```

The self generated router LSA is observed again on each router.. Notice the difference from the earlier output. The router LSAs on each device now show the loopback interfaces with the true network mask of /8:

On R1:

```
R1#show ip ospf database router self-originate | section Stub
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 1.0.0.0
(Link Data) Network Mask: 255.0.0.0
```

```
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

On R2:

```
R2#show ip ospf database router self-originate | section Stub
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 2.0.0.0
(Link Data) Network Mask: 255.0.0.0
```

```
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

(The rest of the output is omitted for brevity)

On R3:

```
R3#show ip ospf database router self-originate | section Stub
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 3.0.0.0
(Link Data) Network Mask: 255.0.0.0
```

```
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

(The rest of the output is omitted for brevity)

The **show ip route ospf** command on R1, R2 and R3 also verify the above configuration changes. A routers install routes to each others loopback 0 interfaces with a mask of /8:

On R1:

```
R1#show ip route ospf | include /8
```

```
O      2.0.0.0/8 [110/2] via 12.1.1.2, 00:00:21, GigabitEthernet0/2
O      3.0.0.0/8 [110/3] via 12.1.1.2, 00:00:04, GigabitEthernet0/2
```

On R2:

```
R2#show ip route ospf | include /8
```

```
O      1.0.0.0/8 [110/2] via 12.1.1.1, 00:01:35, GigabitEthernet0/1
O      3.0.0.0/8 [110/2] via 23.1.1.3, 00:01:06, GigabitEthernet0/3
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
```

On R3:

```
R3#show ip route ospf | include /8
```

```
O    1.0.0.0/8 [110/3] via 23.1.1.2, 00:02:35, GigabitEthernet0/2
O    2.0.0.0/8 [110/2] via 23.1.1.2, 00:02:23, GigabitEthernet0/2
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
```

A hidden command in IOS shows what is known as the OSPF private routing table containing all routes calculated by OSPF. This command is the **show ip ospf route**. Below is output of this command from R1's perspective:

On R1:

```
R1#show ip ospf route
```

```
OSPF Router with ID (1.1.1.1) (Process ID 1)
```

```
Base Topology (MTID 0)
```

```
Area BACKBONE(0)
```

```
Intra-area Route List
```

```
*    12.1.1.0/24, Intra, cost 1, area 0, Connected
    via 12.1.1.1, GigabitEthernet0/2
```

```
*> 23.1.1.0/24, Intra, cost 2, area 0
    via 12.1.1.2, GigabitEthernet0/2
```

```
*    1.0.0.0/8, Intra, cost 1, area 0, Connected
    via 1.1.1.1, Loopback0
```

```
*> 2.0.0.0/8, Intra, cost 2, area 0
    via 12.1.1.2, GigabitEthernet0/2
```

```
*> 3.0.0.0/8, Intra, cost 3, area 0
    via 12.1.1.2, GigabitEthernet0/2
```

```
*> 10.2.2.0/24, Intra, cost 2, area 0
    via 12.1.1.2, GigabitEthernet0/2
```

```
*> 10.3.3.0/24, Intra, cost 3, area 0
    via 12.1.1.2, GigabitEthernet0/2
```

```
First Hop Forwarding Gateway Tree
```

```
12.1.1.1 on GigabitEthernet0/2, count 1
```

```
12.1.1.2 on GigabitEthernet0/2, count 5
```

```
1.1.1.1 on Loopback0, count 1
```

The output lists a concise view of all routes calculated by OSPF that will be submitted to the routing table as OSPF-learned routes. It provides information of the area in which the route was calculated, the OSPF cost to the route, and whether the route is an intra-area, inter-area, or external route. The output uses a BGP-like identification system where * indicates the best route and a > indicates that route has been installed into the global RIB on the router.. In R1's case, the directly-connected routes calculated by OSPF have not been added to its global RIB. This is because the global RIB on R1 will hold them as directly connected routes and OSPF learned routes.

The **show ip ospf route** output is helpful when determining if a synchronization problem exists between the global RIB and the OSPF process' calculated routes.

Task 2

Configure R2 and R3 such that the secondary IP addresses are not advertised into OSPF.

- Do not use filtering, a route map, an access list, or a prefix list.
- Do not remove any commands.
- Use a minimum number of commands to accomplish this task.

As mentioned in task 1, enabling OSPF on an interface with the interface-level command **ip ospf process-id area area-id** command will cause the router to automatically advertise the primary IP address and any secondary IP address configured on that interface into OSPF.

The task amends the previous task where secondary addresses were allowed to be advertised. Instead, R2 and R3 should not advertise the secondary addresses assigned to their G0/3 and G0/2 interfaces respectively, into OSPF. These networks are the 10.2.2.0/24 and 10.3.3.0/24 addresses configured on each interface.

There are two ways this task can be completed:

1. Remove the interface-level command and only match the primary IP address of each interface on R2 and R3 with a **network** command in OSPF router configuration mode
2. Modify the existing interface-level command to exclude secondary IP addresses from being advertised by default.

The task explicitly forbids removing commands from the configuration, this rules out the first option of configuration. What remains is to modify the existing interface-level command to exclude the secondary addresses from being advertised into OSPF. This is achieved by using the **secondaries none** keyword option with the **ip ospf** command under the interface as seen below:

On R2:

```
R2(config)#interface g0/3
R2(config-if)#ip ospf 1 area 0 secondaries none
```

On R3:

```
R3(config)#interface g0/2
R3(config-if)#ip ospf 1 area 0 secondaries none
```

Once configured, the R2 and R3 will no longer advertise the secondary networks into OSPF. Observing the router LSAs on R2 and R3 confirms the two routers are no longer advertising those secondary addresses as directly connected links:

On R2:

```
R2#show ip ospf database router self-originate
```

```
OSPF Router with ID (2.2.2.2) (Process ID 1)
```

```
Router Link States (Area 0)
```

```
LS age: 1017
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 2.2.2.2
Advertising Router: 2.2.2.2
LS Seq Number: 8000000D
Checksum: 0x3945
Length: 72
Number of Links: 4
```

```
Link connected to: a Stub Network
(Link ID) Network/subnet number: 2.0.0.0
(Link Data) Network Mask: 255.0.0.0
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

```
Link connected to: a Transit Network
(Link ID) Designated Router address: 23.1.1.2
(Link Data) Router Interface address: 23.1.1.2
Number of MTID metrics: 0
TOS 0 Metrics: 1
```

```

Link connected to: a Transit Network
(Link ID) Designated Router address: 12.1.1.2

(Link Data) Router Interface address: 12.1.1.2
Number of MTID metrics: 0
TOS 0 Metrics: 1

```

On R3:

R3#**show ip ospf database router self-originate**

```

      OSPF Router with ID (3.3.3.3) (Process ID 1)

```

```

      Router Link States (Area 0)

```

```

LS age: 115
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 3.3.3.3
Advertising Router: 3.3.3.3
LS Seq Number: 8000002D
Checksum: 0xE3BF
Length: 48
Number of Links: 2

```

```

Link connected to: a Stub Network
(Link ID) Network/subnet number: 3.0.0.0
(Link Data) Network Mask: 255.0.0.0
Number of MTID metrics: 0
TOS 0 Metrics: 1

```

```

Link connected to: a Transit Network
(Link ID) Designated Router address: 23.1.1.2
(Link Data) Router Interface address: 23.1.1.3
Number of MTID metrics: 0
TOS 0 Metrics: 1

```

In contrast to the same output from task 1, notice the missing entry for the secondary IP addresses in the router LSAs on R2 and R3 above.

Note: The hidden command **show ip ospf route** can also be quickly referenced to verify the same

A final verification is done on R1 with the **show ip route ospf** command. Notice R1 has no entry in its routing table for the secondary prefixes on R2 and R3:

On R1:

```
R1#show ip route ospf | begin Gate
```

```
Gateway of last resort is not set
```

```
O      2.0.0.0/8 [110/2] via 12.1.1.2, 00:22:07, GigabitEthernet0/2
O      3.0.0.0/8 [110/3] via 12.1.1.2, 00:19:06, GigabitEthernet0/2
      23.0.0.0/24 is subnetted, 1 subnets
O          23.1.1.0 [110/2] via 12.1.1.2, 00:21:29, GigabitEthernet0/2
```

Task 3

R3 is getting flooded with LSA Type 6 packets. Ensure that R3 does not generate a syslog message for this LSA type.

The original OSPFv2 specification in RFC 1583 was extended to include extensions for advertising multicast information. These extensions were defined in RFC 1584 the Multicast Extensions to OSPF further elaborated upon in RFC 1585 MOSPF: Analysis and Experience. These two RFCs outlined a by which multicast destination information can be shared using a new LSA type for that time the Typ LSA.

Some networking vendors support the RFC 1584 Multicast Extensions to OSPF. Network devices manufactured by these vendors can be configured to send and receive Type-6 LSAs. Cisco routers, however, do not support these LSAs. As a result, whenever a router configured for OSPF receives T LSA packets, it generates a number of syslog messages indicating the LSA type was received and is supported. Depending on the size of the network and the number of active multicast groups, logging syslog message can be CPU intensive for the router. In such cases it is best if the router silently disc the Type-6 LSA without logging a message.

This task deals with this very subject. Its goal is to prevent R3 from generating syslog messages whenever a Type-6 LSA is received. The router can be configured to ignore the Type 6 LSAs, using the command **ignore lsa mospf** under the OSPF router configuration mode as seen below on R3:

On R3:

```
R3(config)#router ospf 1
```

```
R3(config-router)#ignore lsa mospf
```

An interesting fact about this command. The context-sensitive help for the **ignore** command actually lists the **lsa** keyword as having the effect of “do not complain upon receiving LSA of the specified type” as seen below:

```
R3(config)#router ospf 1
```

```
R3(config-router)#ignore ?
```

```
lsa Do not complain upon receiving LSA of the specified type
```

This is a further indication of the router's default behavior to alert (or complain) about receiving the unsupported Type-6 LSA.

Task 4

To ensure fast detection of a neighbor being down, configure R2 and R3 to send their hellos every 250 milliseconds with a hold time of 1 second for their Ethernet link.

OSPF hello packets are sent between routers to allow the routers to discover each other and form adjacencies. Once the adjacencies have been formed, routers keep transmitting hello packets on the link every hello interval time. These hellos work as a heartbeat that announces the operational status of a router.

If an OSPF router does not receive a hello packet from one of its neighbors for a period of time, called the dead interval timer, the router declares its neighbor dead and tears down the adjacency with that neighbor. The default hello timer for broadcast and point-to-point links is 10 seconds (for NBMA, the default is 30 seconds). The dead timer on these same links is set to be four times the hello interval. Once a periodic hello packet is received from the neighbor, the router resets the dead interval to its maximum value.

In order to ensure all routers comply by the same hello and dead interval settings, these timers must match between two OSPF-speaking routers in order for them to become neighbors. The current hello and dead timer values can be observed in the **show ip ospf interface** command output. The example output below shows these default values on R2's link G0/3 that connects to R3:

On R2:

```
R2#show ip ospf interface g0/3 | include Hello
```

```
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:04
```

The default dead interval means that the local router will tear down adjacencies with OSPF neighbors if it does not receive four consecutive hello packets from that neighbor. The result of these settings equates to a maximum dead timer of 40 seconds on broadcast and point-to-point links.

The hello and dead timers can be manually set on a per-interface basis using the **ip ospf hello-interval** and **ip ospf dead-interval** interface-level commands. These commands are applied as follows:

- If the **ip ospf hello-interval** is set without manually setting a dead interval, the dead interval is automatically set to 4 times the hello period
- Any interface-level configuration of **ip ospf dead-interval** command will take precedence even if the hello interval is changed at a later time
- The dead interval can be reset to defaults (4 times the hello interval) using the **no** version of the command.

The list above basically means that when configuring the OSPF hello interval on an interface using the **ip ospf hello-interval** command, the dead-interval is automatically set to be 4 times this value. This automatic change is only affected if no manual **ip ospf dead-interval** setting has been applied to the interface. If a dead interval has been specified, that configuration will be applied to the interface. Using the **no** form of the **ip ospf dead-interval** command on the interface returns the dead interval to the default settings.

The task lays out requirements for sending and receiving hello packets between R2 and R3. It specifies that hellos should be sent every 250ms with a 1 second dead interval. Looking at the configuration options for setting these timers reveals a bit of a problem:

```
R2(config-if) #ip ospf hello-interval ?
<1-65535>  Seconds

R2(config-if) #ip ospf dead-interval ?
<1-65535>  Seconds
minimal    Set to 1 second
```

Both hello and dead interval configurations are limited to 1 second defaults. It appears as though there is no way to set a sub-second hello interval as the task desires.

The answer to solving the task lies in the OSPF fast hello feature. The fast hello feature allows the administrator to set sub-second hello timers. When configured, the OSPF fast hello feature first sets the dead interval to 1 second. The hello interval is then specified in how many hellos should be sent within that one second period. This value is called a multiplier value. In other words, if the fast hello feature is configured to send 2 hellos within one second, the hellos will be sent every 500ms. The math works as follows.

A second is made up of 1000ms. With a hello multiplier value of 2, the router should send 2 hello packets at an even interval within that time period. The router divides 1000ms by 2 to reach a 500ms hello time interval.

To solve the task, the hello interval should be set to 250ms. To figure out the proper multiplier value, 1000ms is divided by 250ms to equal a value of 4. This means the multiplier value used for the OSPF fast hello feature should be 4.

Being able to perform that calculation is important as the configuration for OSPF fast hello is through the **ip ospf dead-interval minimal hello-multiplier *multiplier*** command. In this command, the hello multiplier value is used to calculate the resultant hello interval. In case of R2 and R3, the **ip ospf dead-interval minimal hello-multiplier 4** command should be configured the g0/3 and g0/2 on R2 and R3 respectively as in the below:

```
R2 (config) #interface g0/3
R2 (config-if) #ip ospf dead-interval minimal hello-multiplier 4
```

On R3:

```
R3 (config) #interface g0/2
R3 (config-if) #ip ospf dead-interval minimal hello-multiplier 4
```

To verify the configuration:

Following the above configurations, the **show ip ospf interface** command output from R2 and R3 reflect the new timers:

On R3:

```
R3#show ip ospf interface g0/2 | include Hello

Timer intervals configured, Hello 250 msec, Dead 1, Wait 1, Retransmit 5
Hello due in 18 msec
```

Task 5

Ensure that these routers look up DNS names for use in most of the OSPF **show** commands. Test this task to ensure proper operation. Since there are no DNS servers in this lab, you should use the local routers for DNS lookups.

To understand this task, let's first observe the **show ip ospf database router** command output from R1. Output from this command has been truncated to remove R1's self originated Type-1 router LSAs. The output below shows the router LSAs advertised by R2 and R3 as indicated in the **Advertising Router** field:

On R1:

```
R1#show ip ospf database router | include Advert
```

```
Advertising Router: 1.1.1.1
Advertising Router: 2.2.2.2
Advertising Router: 3.3.3.3
```

The advertising router field for the Type 1 router LSAs is the router ID of the router that generated that router LSA. As seen above, the advertising router field for R2 and R3's LSA shows their RID 2.2.2.2 and

3.3.3.3. OSPF can be configured to perform a DNS lookup to display a more easily identifiable name instead of the router ID in OSPF **show EXEC** command outputs.

DNS lookup can be performed against a stand-alone DNS server, another IOS device that is acting as a DNS server, or from the router's own local name-to-IP mapping table configured using the **ip host** command in global configuration mode. The task specifically states that the local router should be used to perform the resolution. Thus, to complete the task, the **ip host** command is configured mapping the friendly name "Rx" (where "x" is the router number) to the proper router-id. For example, R1's name is mapped to 1.1.1.1 using the command **ip host R1 1.1.1.1**. This configuration is repeated on all routers for all resolutions as follows:

On R1, R2 and R3:

```
Rx(config)#ip host R1 1.1.1.1
Rx(config)#ip host R2 2.2.2.2
Rx(config)#ip host R3 3.3.3.3
```

After configuring the manual lookup entries, the OSPF process is enabled to perform DNS lookup using the **ip ospf name-lookup** command in global configuration mode:

```
Rx(config)#ip ospf name-lookup
```

To verify the configuration:

On R1:

```
R1#show ip ospf database router | include Advert
```

```
Advertising Router: R1
Advertising Router: R2
Advertising Router: R3
```

Note: The DNS lookup is performed for all OSPF **show EXEC** command output. As an example, the **show ip ospf neighbor** command which will display the neighbor as R2 instead of its router ID. This may not work on all IOSes.

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

Neighbor ID	Pri	State	Dead Time	Address	Interface
R2	1	FULL/BDR	00:00:33	12.1.1.2	GigabitEthernet0/2

Task 6

Configure R2 such that if it does not receive an acknowledgment from R3 for a given LSA, it waits 10 seconds before it re-sends that given LSA.

OSPF performs reliable flooding of LSAs. This simply means, for every LS update packet a router sends, the router expects to receive a LS Acknowledge packet from its neighbor. If the router does not receive an LS Acknowledge packet from its neighbor acknowledging it received the LS Update, the router will retransmit the LSA. This retransmission is controlled by a set interval dictated by the retransmit timer. The default retransmit timer is set to 5 seconds as seen below for the G0/3 interface on R2:

On R2:

```
R2#show ip ospf interface g0/3 | include Timer
```

```
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

The task requires configuring R2 such that it waits 10 seconds before retransmitting the unacknowledged LSA to R3. This is achieved by modifying the retransmit value for the G0/3 interface to 10 seconds with the **ip ospf retransmit-interval 10** command in interface configuration mode:

```
R2(config)#interface g0/3
R2(config-if)#ip ospf retransmit-interval 10
```

To verify the configuration:

On completing the above, the **show ip ospf interface g0/3** command output from R2 shows the modified retransmit value of 10 seconds:

```
R2#show ip ospf interface g0/3 | include Timer
```

```
Timer intervals configured, Hello 10 msec, Dead 40, Wait 40, Retransmit  
10
```

Task 7

Configure R1 and R2 such that when there is a topology change in Area 0 for LSA Types 1 and 2, the entire SPT is *not* recomputed. This should only occur for the affected part/s of the tree.

OSPF is a link-state routing protocol. That is to say, an OSPF-speaking router exchanges detailed routing information about the state of its links in the form of a Type-1 router or Type-2 network Link State Advertisement (LSA). These LSAs are advertised in Link State Update packets between all routers in the topology and are stored in the Link State Database (LSDB). When two neighbors come online, they exchange all LSAs contained in their LSDBs with each other. In this way the LSDB is kept synchronized between all routers in a particular OSPF area.

Once the LSDB has been synchronized within an area, the router can construct a graph or map of the area topology. This graph is made up of nodes and the links that connect to the nodes. After complete topology information has been mapped or graphed out, OSPF performs a shortest-path computation to find the shortest path to reach any node on the graph. OSPF uses the Dijkstra SPF algorithm to accomplish this. After running the Dijkstra SPF algorithm against all nodes on the graph, the router has constructed a shortest-path tree from itself (as the root of the tree) to all other nodes on the graph. It is from this tree that the router derives the routes it installs into the routing table.

Any changes to a router's link-state information, such as removing or adding a link connected to this router, results in a change to the graph of the area. The affected router advertises this change in an LS UPDATE packet amending its LSAs. After receiving the update packet containing new Type-1 router or Type-2 network link state information for a particular router, the local router must recompute its own SPT to ensure it is still using the most efficient path to reach all nodes within its area.

There are circumstances, however, where a change in the graph does not affect the local router's resulting SPT or may only affect a portion of the router's SPT. In such cases, it is a waste of effort on the local router's side to recalculate the entire SPT from scratch. Doing so could become CPU intensive if the OSPF area is very large, containing OSPF routers.

For example, in the lab topology, R3, from R1's perspective, is a leaf-node router. That is to say, R3 doesn't connect to any other downstream routers. R2, on the other hand, connects R1 to R3 and as such R2 is a branch router that allows R1 to transit traffic to R3. If R3's link to R2 were to go down, it would not be necessary for R1 to recalculate its entire SPT again only to remove R3 from the very edge of the network.

The example above, is the situation the task is looking to address. It states that if there is a change in Type-1 router and Type-2 network LSAs in Area 0, R1 and R2 should not run a full SPF. Only the affected portions of the SPT should be recalculated. This means that rather than recomputing the entire SPT, R1 can simply remove R3 from the edge of its SPT with minimal consequence, running only a partial SPF algorithm for only the affected part of its SPT (the R2/R3 link). This is called an

incremental SPF calculation.

Incremental SPF calculations optimize SPF calculations by establishing two types of calculations:

1. Full SPF - Where SPF is run against the entirety of the LSDB
2. Incremental SPF - Where only the affected part of the local router's SPT is recalculated

Full SPF calculations occur whenever a change is made to one of the routers own Type-1 or Type-2 LSAs. This is an indication that one of the router's core links have undergone a change in some way. example, if R1 loses its connection with R2, it must run a full SPF to recompute its entire SPT.

Incremental SPF calculations occur in the following:

1. Whenever a leaf node router or link comes up or down
2. Whenever a non-SPT link comes up or down
3. Whenever a branch of the SPT fails

The first, is a direct reference to the R2/R3 link from R1's perspective. If that link goes down, R1 only needs to run an incremental SPF calculation to remove R3 from its SPT. The second references a link in the topology that R1 does not use in its SPT to reach any other node. Running a full calculation in this case would be a waste as there will effectively be no change in the resultant SPT from R1's perspective. The last references a situation where an indirectly-connected branch of the SPT goes down. For example, if R3 were connected to another router, R4 in this case. If the R2/R3 link goes down, R1 still only needs to recalculate that failure and does not need to run a full SPF calculation.

The incremental spf feature is enabled using the **ispf** command in OSPF router configuration mode. To complete the task, this feature needs to be enabled on R1 and R2. As seen in the example output from R1 below, Incremental SPF is turned off by default:

On R1 and R2:

```
Rx#show ip ospf | include Increm
```

```
Incremental-SPF disabled
```

On R3:

Before configuring incremental SPF, let's observe what happens on R1 whenever the R2/R3 link is shut down:

```
R3(config)#interface g0/2
R3(config-if)#shut
```

You should see the following messages:

Process 1, Nbr 2.2.2.2 on GigabitEthernet0/2 from FULL to DOWN, Neighbor Down: Interface down or detached

%LINK-5-CHANGED: Interface GigabitEthernet0/2, changed state to administratively down

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/2, changed state to down

G0/2 is shut down on R3 in the above. In the output below interface G0/2 is brought back up on R3. The **show ip ospf statistics detail** command is used on R1 to see what kind of computation R1 performed:

On R1:

R1#show ip ospf statistics detail

OSPF Router with ID (1.1.1.1) (Process ID 1)

Area 0: SPF algorithm executed 4 times

SPF 1 executed 00:43:35 ago, SPF type Full

SPF calculation time (in msec):

SPT Intra D-Intr Summ D-Summ Ext7 D-Ext7 Total 01000001 LSIDs processed R:3 N:2 Stub:4
SN:0 SA:0 X7:0

Change record 0x0

LSIDs changed 5

Changed LSAs. Recorded is LS ID and LS type: 2.2.2.2(R) 3.3.3.3(R) 12.1.1.2(N)
23.1.1.2(N) 1.1.1.1(R)

SPF 2 executed 00:07:21 ago, SPF type Full

SPF calculation time (in msec):

SPT Intra D-Intr Summ D-Summ Ext7 D-Ext7 Total 00100001 LSIDs processed R:2 N:1 Stub:3
SN:0 SA:0 X7:0

Change record 0x0

LSIDs changed 2

Changed LSAs. Recorded is LS ID and LS type:

2.2.2.2(R) 23.1.1.2(N)

SPF 3 executed 00:05:16 ago, SPF type Full

SPF calculation time (in msec):

SPT Intra D-Intr Summ D-Summ Ext7 D-Ext7 Total 00000000 LSIDs processed R:2 N:2 Stub:2
SN:0 SA:0 X7:0

Change record 0x0

LSIDs changed 3

Changed LSAs. Recorded is LS ID and LS type: 3.3.3.3(R) 2.2.2.2(R) 23.1.1.3(N)

SPF 4 executed 00:05:06 ago, SPF type Full

```
SPF calculation time (in msec):
SPT Intra D-Intr Summ D-Summ Ext7 D-Ext7 Total 00000000 LSIDs processed R:3 N:2 Stub:4
SN:0 SA:0 X7:0
Change record 0x0
LSIDs changed 1
Changed LSAs. Recorded is LS ID and LS type: 3.3.3.3(R)
```

The **show ip ospf statistics detail** output lists OSPF SPF calculation activity. It includes how long it took for a particular SPF calculation, the total number of SPF calculations, what kind of SPF calculation was used, and what LSAs triggered the SPF calculation. The output above reveals that R1 has run SPF 3 times. The most recent runs are SPF 3 and SPF 4. In SPF 2, R1 performed a full SPF update because the Type-2 LSA from R2 changed. This was a result of the R2/R3 link failing. In SPF 3 and 4, R1 ran another full SPF calculation. This was whenever the R2/R3 link was brought down and back up.

Next, incremental SPF will be configured using the **ispf** command under the OSPF router configuration mode on R1 and R2:

On R1, and R2:

```
Rx(config)#router ospf 1
Rx(config-router)#ispf
```

To verify the configuration:

On R1:

The **show ip ospf** command output from R1 and R2 now reveal this feature is enabled:

```
R1#show ip ospf | include Incremental
```

```
Incremental-SPF enabled
```

On R3:

Once again, the R2/R3 link is brought down and back up:

```
R3(config)#interface g0/2
R3(config-if)#no shut
R3(config-if)#shut
```

On R1:

The **show ip ospf statistics detail** command now lists the following:

```
R1#show ip ospf statistics detail
```

```

      OSPF Router with ID (1.1.1.1) (Process ID 1)
    Area 0: SPF algorithm executed 7 times
[--- omitted ---]
```

```
SPF 6 executed 00:01:26 ago, SPF type Incremental
```

```
SPF calculation time (in msec):
```

```
SPT Intra D-Intr Summ D-Summ Ext7 D-Ext7 Total 00000000 LSIDs processed R:2 N:2 Stub:2 SN:0 SA:0 X7:0
```

```
Change record 0x0
```

```
LSIDs changed 4
```

```
Changed LSAs. Recorded is LS ID and LS type: 3.3.3.3(R) 23.1.1.3(N) 2.2.2.2(R) 23.1.1.2(N)
```

```
SPF 7 executed 00:01:16 ago, SPF type Incremental
```

```
SPF calculation time (in msec):
```

```
SPT Intra D-Intr Summ D-Summ Ext7 D-Ext7 Total 00000000 LSIDs processed R:3 N:2 Stub:4 SN:0 SA:0 X7:0
```

```
Change record 0x0
```

```
LSIDs changed 1
```

```
Changed LSAs. Recorded is LS ID and LS type: 3.3.3.3(R)
```

The truncated output above shows that R1 used incremental calculations for its last two SPF calculations as required by the task.

Task 8

Erase the startup configuration and reload the routers before proceeding to the next lab.