INTRODUCTION TO CISCO IOS XR

Technical Workbook

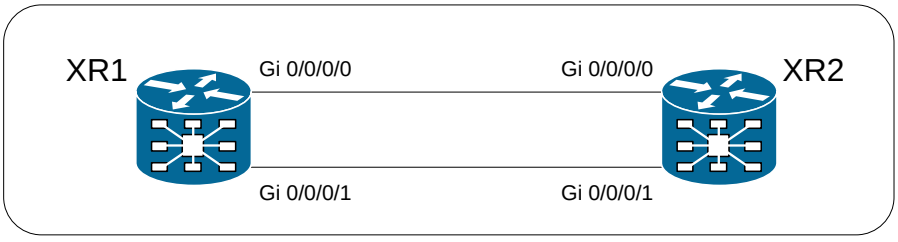
February 2014

This workbook provides a brief introductory tour to Cisco's IOS XR operating system as a supplement to the Cisco Configuration Guides found on here:

Cisco IOS XR Software Support and Documentation

The following tasks are based off a topology with 2 IOS XRv routers connected via two Ethernet connections as depicted in Figure 1.

Figure 1 Basic XRv Topology



The workbook is structured so that tasks are performed only on XR1. XR2 is pre-configured by copying the initial configuration found in **Appendix A: XR2 Configuration** in its entirety and pasting into the console of XR2.

By Brad Edgeworth and Vinit Jain

For More Information

Read more about the Cisco IOS XR Software, or contact your local account representative. The Cisco Press book *Cisco IOS XR Fundamentals* by Mobeen Tahir, et al. Cisco Press, 2009. Print is an additional resource on Cisco's IOS XR architecture.

Document Authors:

Brad Edgeworth

Vinit Jain

Tasks:

System Administration

Task 1.1: Identify Software Version

- Identify the Software Version on the router

Task 1.2: Identify Interfaces

- Identify the Physical Interfaces

Task 1.3: Configure Ipv4 Interfaces

- Configure IPv4 address 10.12.1.1/24 on interface Gi0/0/0/0
- Configure IPv4 address 192.168.1.1/32 on interface Loopback 0
- Verify the IPv4 addresses and connectivity with 10.12.1.2

Task 1.4: Configure Ipv6 Interfaces

- Configure IPv6 address fec0:12::1/64 on interface Gi0/0/0/0
- Configure IPv6 address fec0::1/128 on interface Loopback 0
- Verify the IPv6 addresses and connectivity to fec0:12::1

Task 1.5: View the Configuration

- Display the running-configuration

Task 1.6: Pre-staging changes

- Change the hostname to XR1
- Verify the configuration change
- Do not commit the change but save it to the text-file XR1hostname.txt
- Exit configuration Mode

Task 1.7: Loading staged changes

- Load the text-file XR1hostname.txt
- Verify the configuration
- Commit the change, use the label HOSTNAME

Task 1.7: View Configuration Changes

- Display all the changes that have occurred on the router

Task 1.8: View a Specific Change

- View a list of the configuration commits
- View the last two changes

- View a specific change

Task 1.9: Perform a Rollback

- Change the hostname to XR-Rollback, and commit the change.
- Rollback the configuration to the change label of HOSTNAME

Task 1.10: Assign IPv4 Addresses to Sub-Interfaces

- Set the hostname of the router to XR1
- Assign IPv4 Addresses to the following interfaces:

Interface	VLAN	IP Address and Subnet Mask	Remote Endpoint
GigabitEtherent 0/0/0/1.13	13	10.13.1.1/24	10.13.1.3
GigabitEtherent 0/0/0/1.14	14	172.16.14.1/24	172.16.14.4
GigabitEtherent 0/0/0/1.13	15	172.16.15.1/24	172.16.15.1.5
Loopback 1	N/A	192.168.11.11/32	N/A

- Verify connectivity to the remote endpoints

Static Routes

Task 2.1: IPv4 Static Route

- Create a static route to the 10.33.33.0/24 network with a next-hop of 10.13.1.3
- Verify that you can ping the 10.33.33.33 IPv4 address

Task 2.2: IPv6 Static Route

- Create a static route to the FEC0:22::/64 network with a next-hop of FEC0:12::2
- Verify that you can ping the FECO:22:22 IPv6 address

OSPF

Task 3.1: Basic OSPF

- Create OSPF process 1
- Enable interface Gi0/0/0/0, Gi0/0/0/1.13, and Loopback 0 for Area 0
- Enable interface Loopback 1 for Area 1
- Verify OSPF interfaces
- Verify OSPF Neighbors

Task 3.2: Global Inheritance

- Configure OSPF so that all interfaces will have a cost of 100
- Verify the OSPF cost for all interfaces

Task 3.3: Area Inheritance

- Configure OSPF so that all interfaces in Area 0 will have a cost of 50
- Verify the OSPF cost for all interfaces

Task 3.4: Interface Inheritance

- Configure OSPF so that Loopback 0 has a cost of 10
- Verify the OSPF cost for all interfaces

BGP

Task 4.1: iBGP

- Create the BGP process with the **router bgp** *ASN* command.
- Configure BGP using the Autonomous System (AS) number 100
- Configure iBGP neighbor with 192.168.2.2 sourcing the session from loopback0 interface
- Verify iBGP neighborship
- Verify BGP routes in BGP table

Task 4.2: eBGP

- Configure an eBGP session between XR1 and 172.16.14.4 using remote-as 200
- Make XR1 appear like it is AS 150 with the '**local-as**' feature
- Verify eBGP neighborship
- Verify that routes are passing

Task 4.3: Basic Route-Policy

- Create the route-policy PASS-ALL permitting all prefixes
- Apply the route-policy pass-all in inbound and outbound direction on the eBGP neighbor 172.16.14.4 for address-family ipv4 unicast.
- Verify the routes exchanged between eBGP peers

Task 4.4: Route-Policy with If-Else

- Create a route-policy called BLOCK-AS65000 to filter routes with AS 65000 in the AS_PATH
  Use IOS regex processing for this task
- Apply the route-policy under BGP neighbor 172.16.14.4 address-family ipv4 unicast
- Verify the policy is working

Solutions:

System Administration

Task 1.1: Identify Software Version

- Identify the software version on the router

Example 1-1 provides output from the familiar **show version** command found on other Cisco Operating Systems. Notice that the major software revision is provided and is followed by the versions of the other software packages. Notice that the platform is XRv running the 5.1.1 image.

Example 1-1 *Show version*

```
RP/0/0/CPU0:ios#show version

Cisco IOS XR Software, Version 5.1.1.12C[Default]
Copyright (c) 2013 by Cisco Systems, Inc.

ROM: GRUB, Version 1.99(0), DEV RELEASE

ios uptime is 32 minutes
System image file is "bootflash:disk0/xrvr-os-mbi-5.1.1.12C/mbixrvr-rp.vm"

cisco IOS XRv Series (Intel 686 F6M2S3) processor with 3169905K bytes of memory.
Intel 686 F6M2S3 processor at 3002MHz, Revision 2.174
IOS XRv Chassis

1 Management Ethernet
2 GigabitEthernet
96050k bytes of non-volatile configuration memory.
866M bytes of hard disk.
1866736k bytes of disk0: (Sector size 512 bytes).

Configuration register on node 0/0/CPU0 is 0x2102
Boot device on node 0/0/CPU0 is disk0:
Package active on node 0/0/CPU0:
iosxr-infra, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-infra-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-fwding, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-fwding-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-routing, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-routing-
5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-ce, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-ce-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

xrvr-os-mbi, V 5.1.1.12C[Default], Cisco Systems, at disk0:xrvr-os-mbi-5.1.1.12C
    Built on Wed Oct 23 16:26:28 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

xrvr-base, V 5.1.1.12C[Default], Cisco Systems, at disk0:xrvr-base-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie
```

```
xrvr-fwding, V 5.1.1.12C[Default], Cisco Systems, at disk0:xrvr-fwding-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-mpls, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-mpls-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-mgbl, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-mgbl-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-mcast, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-mcast-5.1.1.12C
    Built on Wed Oct 23 16:25:48 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

xrvr-mcast-supp, V 5.1.1.12C[Default], Cisco Systems, at disk0:xrvr-mcast-supp-
5.1.1.12C
    Built on Wed Oct 23 16:25:49 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-bng, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-bng-5.1.1.12C
    Built on Wed Oct 23 16:25:50 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

xrvr-bng-supp, V 5.1.1.12C[Default], Cisco Systems, at disk0:xrvr-bng-supp-
5.1.1.12C
    Built on Wed Oct 23 16:25:51 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

iosxr-security, V 5.1.1.12C[Default], Cisco Systems, at disk0:iosxr-security-
5.1.1.12C
    Built on Wed Oct 23 16:25:51 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie

xrvr-fullk9-x, V 5.1.1.12C[Default], Cisco Systems, at disk0:xrvr-fullk9-x-
5.1.1.12C
    Built on Wed Oct 23 16:26:32 UTC 2013
    By rtp-ads-066 in /nobackup/glmatthe/xrvr-eft for pie
```

Task 1.2: Identify Physical Interfaces

- Identify the physical interfaces

The command **show interfaces** [**brief**] found on other Cisco Operating Systems works in a similar fashion. Example 1-2 demonstrates the brief version of the output. Notice that all the interfaces start in admin-down state.

Example 1-2 *Show version*

```
RP/0/0/CPU0:ios#show interfaces brief

                Intf       Intf        LineP              Encap  MTU        BW
                Name       State       State               Type (byte)    (Kbps)
--------------------------------------------------------------------------------
                 Nu0          up          up               Null  1500          0
       Mg0/0/CPU0/0  admin-down  admin-down               ARPA  1514          0
          Gi0/0/0/0  admin-down  admin-down               ARPA  1514    1000000
          Gi0/0/0/1  admin-down  admin-down               ARPA  1514    1000000
```

Note: The Mg0/0/CPU0/0 interface is an *out-of-band (OOB)* management interface. This will always be the first interface associated with the XRv VM as indicated in the XRv installation guide.

Task 1.3: Configure Ipv4 Interfaces

- Configure IPv4 address 10.12.1.1/24 on interface Gi0/0/0/0
- Configure IPv4 address 192.168.1.1/32 on interface Loopback 0
- Verify the IPv4 addresses and connectivity with 10.12.1.2

IOS XR uses a *two-stage commit* for processing of configuration changes.

In the first stage, a *target configuration* is created where changes are entered. The target configuration only contains changes that have been entered, and does not contain a full copy of the running-configuration. Upon entering a command, IOS XR parses the configuration for syntax and then applies the command to the target configuration. Multiple configuration commands can be made to the target configuration as needed.

In the second stage, the target configuration processes the changes into the running-configuration after applying the **commit** command.

In addition, configuring IPv4 address use the command **ipv4 address** *ip-address* [*subnet-mask*|*/prefix_length*].

Example 1-3 provides sample configuration of this task. Notice that the IP address can be configured either with the subnet mask or prefix length.

Example 1-3 *IP Address Configuration*

```
RP/0/0/CPU0:ios#configure terminal
RP/0/0/CPU0:ios(config)#interface GigabitEthernet 0/0/0/0
RP/0/0/CPU0:ios(config-if)#ipv4 address 10.12.1.1/24
RP/0/0/CPU0:ios(config-if)#no shut
RP/0/0/CPU0:ios(config-if)#interface Loopback 0
RP/0/0/CPU0:ios(config-if)#ipv4 address 192.168.1.1 255.255.255.255
RP/0/0/CPU0:ios(config-if)#no shut
RP/0/0/CPU0:ios(config-if)#commit
!  The commit command processes the target configuration into the running-config
RP/0/0/CPU0:Jan 15 16:52:02.056 : config[65708]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'cisco'. Use 'show configuration commit changes
1000000001' to view the changes.
RP/0/0/CPU0:Jan 15 16:52:01.456 : ifmgr[224]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/0, changed state to Down
RP/0/0/CPU0:Jan 15 16:52:01.506 : ifmgr[224]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/0, changed state to Up
```

The target configuration is shown with the command **show configuration** from the configuration mode. Example 1-4 demonstrates the command

Example 1-4 *Viewing Target Configuration*

```
RP/0/0/CPU0:ios(config-if)#show configuration
Building configuration...
```

```
!! IOS XR Configuration 5.1.1.12C
interface Loopback0
 ipv4 address 192.168.1.1 255.255.255.255
 no shutdown
!
interface GigabitEthernet0/0/0/0
 ipv4 address 10.12.1.1 255.255.255.0
 no shutdown
!
end
```

The target configuration is commited in Example 1-5. Notice the commit-ID of 10000000001

Example 1-5 *Commiting the Change*

```
RP/0/0/CPU0:ios(config-if)#commit
!  The commit command processes the target configuration into the running-config
RP/0/0/CPU0:Jan 15 16:52:02.056 : config[65708]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'cisco'. Use 'show configuration commit changes
1000000001' to view the changes.
RP/0/0/CPU0:Jan 15 16:52:01.456 : ifmgr[224]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/0, changed state to Down
RP/0/0/CPU0:Jan 15 16:52:01.506 : ifmgr[224]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/0, changed state to Up
```

Note: Some output may not be shown unless you enable terminal monitoring with the command **term mon**.

Example 1-6 shows verification of the IPv4 addressing on an router. IOS XR uses the command **show ipv4 interface** [*interface-type interface-number*] [**brief**].

Example 1-6 *IP Address Verification*

```
RP/0/0/CPU0:ios#show ipv4 interface brief

Interface                      IP-Address      Status                Protocol
Loopback0                      192.168.1.1     Up                    Up
MgmtEth0/0/CPU0/0              unassigned      Shutdown              Down
GigabitEthernet0/0/0/0         10.12.1.1       Up                    Up
GigabitEthernet0/0/0/1         unassigned      Shutdown              Down
```

Example 1-7 verifies connectivity to XR2's 10.12.1.2 IP address.

Example 1-7 *IPv4 Connectivity Verification*

```
RP/0/0/CPU0:ios#ping 10.12.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.12.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/19 ms
```

Task 1.4: Configure Ipv6 Interfaces

- Configure IPv6 address fec0:12::1/64 on interface Gi0/0/0/0
- Configure IPv6 address fec0::1/128 on interface Loopback 0
- Verify the IPv6 addresses and connectivity to fec0:12::1

Configuring IPv6 addresses use the command **ipv6 address** *ip-address subnet-mask*

Example 1-8 provides sample configuration of this task.

Example 1-8 *IP Address Configuration*

```
RP/0/0/CPU0:ios(config)#interface GigabitEthernet 0/0/0/0
RP/0/0/CPU0:ios(config-if)#ipv6 address fec0:12::1/64
RP/0/0/CPU0:ios(config-if)#interface loopback 0
RP/0/0/CPU0:ios(config-if)#ipv6 address fec0::1/128
RP/0/0/CPU0:ios(config-if)#commit
```

Example 1-9 shows verification of the IPv6 addressing. IOS XR uses the command **show ipv6 interface** [*interface-type interface-number*] [**brief**]. Notice that the IPv6 Link-Local Address is automatically configured.

Example 1-9 *IP Address Verification*

```
RP/0/0/CPU0:ios(config-if)#do show ipv6 interface brief
Loopback0              [Up/Up]
    fe80::c8bf:f1ff:fedb:e1e2
    fec0::1
MgmtEth0/0/CPU0/0      [Shutdown/Down]
    unassigned
GigabitEthernet0/0/0/0 [Up/Up]
    fe80::f816:3eff:fe3f:953
    fec0:12::1
GigabitEthernet0/0/0/1 [Shutdown/Down]
    unassigned
```

Note: IOS XR supports the '**do**' command that allows executive commands to be entered while in configuration mode. This is illustrated in Example 1-9.

Example 1-10 verifies connectivity to XR2's fec0:12::2 IPv6 address.

Example 1-10 *IPv6 Connectivity Verification*

```
RP/0/0/CPU0:ios#ping fec0:12::2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to fec0:12::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

Task 1.5: View the Configuration

- Display the running-configuration

The command **show running-configuration** displays IOS XR's running-config as shown in Example 1-11.

Example 1-11 *Show running-config*

```
RP/0/0/CPU0:ios#show running-config
Building configuration...
!! IOS XR Configuration 5.1.1.12C
!! Last configuration change at Wed Jan 15 17:01:30 2014 by cisco
!
interface Loopback0
 ipv4 address 192.168.1.1 255.255.255.255
 ipv6 address fec0::1/128
!
interface MgmtEth0/0/CPU0/0
 shutdown
!
interface GigabitEthernet0/0/0/0
 ipv4 address 10.12.1.1 255.255.255.0
 ipv6 address fec0:12::1/64
!
interface GigabitEthernet0/0/0/1
 shutdown
!
end
```

Note: A nice feature of the **show running-configuration** command is that you can see the configuration for specific processes without having to parse it with a "**| begin**" or "**| section**" command. This functionality is shown in the static router section of this workbook.

Task 1.6: Pre-staging Changes

- Change the hostname to XR1
- Do not commit the change but save it to the text-file *XR1hostname.txt*
- Exit configuration Mode

Instead of committing the target configuration, the change can be saved to the local storage in preparation for a future change. The command **save configuration** *URL* saves the target configuration in IOS XR.

Example 1-12 demonstrates the hostname being changed and saved as '*XR1hostname.txt*' for another time. Notice that IOS XR notifies you if an uncommitted change exists in the target configuration when leaving the configuration mode.

Example 1-12 *Saving Target Configuration*

```
RP/0/0/CPU0:ios(config)#hostname XR1
RP/0/0/CPU0:ios(config)#save configuration disk0:XR1hostname.txt
Destination file name (control-c to abort): [/XR1hostname.txt]?
Building configuration.
10 lines built in 1 second
[OK]
RP/0/0/CPU0:ios(config)#end
Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:no
RP/0/0/CPU0:ios#
```

Task 1.7: Loading Staged Changes

- Load the text-file XR1hostname.txt
- Verify the configuration
- Commit the change, use the label HOSTNAME

Files can be loaded into the target configuration with the command **load** *URL* as shown in Example 1-13

Example 1-13 *Load Configuration File on Storage*

```
RP/0/0/CPU0:ios#conf t
RP/0/0/CPU0:ios(config)#load disk0:XR1hostname.txt
Loading.
214 bytes parsed in 1 sec (207)bytes/sec
```

Example 1-14 verifies the target configuration.

Example 1-14 *Verify Target Configuration*

```
RP/0/0/CPU0:ios(config)#show configuration
Building configuration...
!! IOS XR Configuration 5.1.1.12C
hostname XR1
end
```

Note: Staging configurations into text files that can be loaded reduces the chance of typographical errors during changes to a configuration.

The **commit** command has various options. One option includes a user-friendly label to help identify a specific change in the event it needs to be referenced later.

The command **commit label** *unique-label-string* adds a label to the commit-id as shown in Example 1-15. Notice that the commit-ID is still shown.

Example 1-15 *Commit Label*

```
RP/0/0/CPU0:ios(config)#commit label HOSTNAME
RP/0/0/CPU0:Jan 15 18:04:30.868 : config[65708]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'cisco'. Use 'show configuration commit changes
1000000003' to view the changes.
```

Task 1.8: View a Specific Change

- View a list of the configuration commits
- View the last two changes
- View a specific change

IOS XR stores the running-configuration within a database referred to as SysDB. As changes are committed to the system, a commit-ID is associated to the configuration modification and entered into the SysDB.

The SysDB can be queried with the command **show configuration commit list**, which provides a complete listing of the changes in the SysDB. Changes are listed with the most recent being first, along with the commit-ID (or commit label when used), the username, method, and timestamp for the commit. Example 1-16 shows the output of this command.

Example 1-16 *Show Configuration Commit List*

```
RP/0/0/CPU0:XR1#show configuration commit list
SNo. Label/ID    User      Line               Client      Time Stamp
~~~~ ~~~~~~~~    ~~~~      ~~~~               ~~~~~~      ~~~~~~~~~~
1    HOSTNAME    cisco     con0_0_CPU0        CLI         Wed Jan 15 18:04:30 2014
2    1000000002  cisco     con0_0_CPU0        CLI         Wed Jan 15 17:01:30 2014
3    1000000001  cisco     con0_0_CPU0        CLI         Wed Jan 15 16:52:01 2014
```

To see the specific changes over multiple commits, the command **show configuration commit changes last** *number* is used. Example 1-17 demonstrates looking at the configuration changes over the last two commits.

Example 1-17 *Show Last Two Configuration Changes*

```
RP/0/0/CPU0:XR1#show configuration commit changes last 2
Building configuration...
!! IOS XR Configuration 5.1.1.12C
hostname XR1
interface Loopback0
 ipv6 address fec0::1/128
!
interface GigabitEthernet0/0/0/0
 ipv6 address fec0:12::1/64
!
end
```

A specific change can be viewed with the command **show configuration commit changes** { *commit-ID* | *label* }. Example 1-18 demonstrates both techniques.

Example 1-18 *Show Configuration Changes for a Specific Commit*

```
RP/0/0/CPU0:XR1#show configuration commit changes 1000000003
Building configuration...
!! IOS XR Configuration 5.1.1.12C
hostname XR1
end
```

```
RP/0/0/CPU0:XR1#show configuration commit changes HOSTNAME
Building configuration...
!! IOS XR Configuration 5.1.1.12C
hostname XR1
end
```

Note: The usage of labels helps in identifying a specific change.

Task 1.9: Perform a Rollback

- Change the hostname to *XR-Rollback*, and commit the change.
- Rollback the configuration to the change label of *HOSTNAME*

Example-1-19 demonstrates the change of the hostname.

Example 1-19 *Change of Hostname*

```
RP/0/0/CPU0:ios(config)#hostname XR-Rollback
RP/0/0/CPU0:ios(config)#commit
RP/0/0/CPU0:Jan 15 19:08:12.936 : ike[226]: %SECURITY-IKE-4-WARNING : You may want
to configure a domain-name
RP/0/0/CPU0:Jan 15 19:08:13.026 : config[65708]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'cisco'. Use 'show configuration commit changes
1000000004' to view the changes.
```

IOS XR allows one or multiple changes to be rolled back through the usage of the SysDB. Upon rolling back a configuration change, the operator can specify the number of committed changes to rollback or to the configuration before a specific commit-ID.

The rollback command follows the syntax **rollback configuration** {**last** *number_of_changes* | **to** *commit-ID/Change-Label* }. Example 1-20 shows the rollback to the commit-label HOSTNAME. Notice that the hostname changed to the configuration before the HOSTNAME commit was performed.

Example 1-20 *Rollback*

```
RP/0/0/CPU0:XR-Rollback#rollback configuration to HOSTNAME
Loading Rollback Changes.
Loaded Rollback Changes in 1 sec
Committing.
RP/0/0/CPU0:Jan 15 19:08:33.975 : config_rollback[65708]: %MGBL-CONFIG-6-DB_COMMIT
: Configuration committed by user 'cisco'. Use 'show configuration commit changes
1000000005' to view the changes.
2 items committed in 1 sec (1)items/sec
Updating.
Updated Commit database in 1 sec
Configuration successfully rolled back to 'HOSTNAME'.
RP/0/0/CPU0:ios#
```

IOS XR considers a rollback as a change and is provided a commit-ID as shown in Example 1-20. Example 1-21 verifies this concept as the commit-ID of 1000000005 shows a client of 'Rollback'.

Example 1-21 *Show Configuration Commit List*

```
RP/0/0/CPU0:ios#show configuration commit list
SNo. Label/ID    User      Line                 Client      Time Stamp
~~~~ ~~~~~~~~    ~~~~      ~~~~                 ~~~~~~      ~~~~~~~~~~
1    1000000005  cisco     con0_0_CPU0          Rollback    Wed Jan 15 19:08:32 2014
2    1000000004  cisco     con0_0_CPU0          CLI         Wed Jan 15 19:06:10 2014
3    HOSTNAME    cisco     con0_0_CPU0          CLI         Wed Jan 15 18:04:30 2014
4    1000000002  cisco     con0_0_CPU0          CLI         Wed Jan 15 17:01:30 2014
5    1000000001  cisco     con0_0_CPU0          CLI         Wed Jan 15 16:52:01 2014
```

Task 1.10: Assign IPv4 Addresses to Sub-Interfaces

- Set the hostname of the router to XR1
- Assign IPv4 Addresses to the following interfaces:

Interface	VLAN	IP Address and Subnet Mask	Remote Endpoint
GigabitEtherent 0/0/0/1.13	13	10.13.1.1/24	10.13.1.3
GigabitEtherent 0/0/0/1.14	14	172.16.14.1/24	172.16.14.4
GigabitEtherent 0/0/0/1.13	15	172.16.15.1/24	172.16.15.1.5
Loopback 1	N/A	192.168.11.11/32	N/A

Sub-interface configuration requires creating the sub-interface and then specifying the 802.1Q tag used with the command **encapsulation dot1q** *vlan-number*. Example 1-22 provides example configuration.

Example 1-22 *Configuration of Sub-Interfaces*

```
RP/0/0/CPU0:ios(config)#hostname XR1
RP/0/0/CPU0:ios(config)#interface gi0/0/0/1
RP/0/0/CPU0:ios(config-if)#no shut
RP/0/0/CPU0:ios(config-if)#interface gi0/0/0/1.13
RP/0/0/CPU0:ios(config-subif)#encapsulation dot1q 13
RP/0/0/CPU0:ios(config-subif)#ipv4 add 10.13.1.1/24
RP/0/0/CPU0:ios(config-subif)#interface gi0/0/0/1.14
RP/0/0/CPU0:ios(config-subif)#encapsulation dot1q 14
RP/0/0/CPU0:ios(config-subif)#ipv4 add 172.16.14.1/24
RP/0/0/CPU0:ios(config-subif)#interface gi0/0/0/1.15
RP/0/0/CPU0:ios(config-subif)#encapsulation dot1q 15
RP/0/0/CPU0:ios(config-subif)#ipv4 add 172.16.15.1/24
RP/0/0/CPU0:ios(config-subif)#interface loopback1
RP/0/0/CPU0:ios(config-if)#ipv4 add 192.168.11.11/32
RP/0/0/CPU0:ios(config-if)#commit
```

Example 1-23 provides verification of the newly configured interfaces.

Example 1-23 *Verification of Sub-Interfaces*

```
RP/0/0/CPU0:XR1#show ipv4 interface brief

Interface                  IP-Address      Status          Protocol
Loopback0                  192.168.1.1     Up              Up
Loopback1                  192.168.11.11   Up              Up
MgmtEth0/0/CPU0/0          unassigned      Shutdown        Down
GigabitEthernet0/0/0/0     10.12.1.1       Up              Up
GigabitEthernet0/0/0/1     unassigned      Up              Up
GigabitEthernet0/0/0/1.13  10.13.1.1       Up              Up
GigabitEthernet0/0/0/1.14  172.16.14.1     Up              Up
GigabitEthernet0/0/0/1.15  172.16.15.1     Up              Up
```

Example 1-24 provides verification of connectivity to remote end-points.

Example 1-24 *Remote End Point Connectivity Check*

```
RP/0/0/CPU0:XR1#ping 10.13.1.1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.13.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

RP/0/0/CPU0:XR1#ping 172.16.14.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.14.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/6/19 ms

RP/0/0/CPU0:XR1#ping 172.16.15.5
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.15.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/7/9 ms
```

Static Routes

Task 2.1: IPv4 Static Route

- Create a static route to the 10.33.33.0/24 network with a next-hop of 10.13.1.3
- Verify that you can ping the 10.33.33.33 IPv4 address

IOS XR maintains a process specifically for static routing. The process initializes with the command **router static**. Static routes are then associated with the correct address-family, so IPv4 routes are placed under the **address-family ipv4 unicast** section. The command for the static route is *network* {*subnet_mask* | */prefix_length*} *next-hop_IP* as shown below in Example 2-1.

Example 2-1 *IPv4 Static Route Configuration*

```
RP/0/0/CPU0:XR1(config)#router static
RP/0/0/CPU0:XR1(config-static)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-static-afi)#10.33.33.0/24 10.13.1.3
RP/0/0/CPU0:XR1(config-static-afi)#commit
```

Example 2-2 verifies the entry of the static route into the routing table with the command **show route** [**ipv4**].

Example 2-2 *Verification of IPv4 Route Table*

```
RP/0/0/CPU0:XR1#show route

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       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, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR
       A - access/subscriber, a - Application route, (!) - FRR Backup path

Gateway of last resort is not set

C    10.12.1.0/24 is directly connected, 03:01:16, GigabitEthernet0/0/0/0
L    10.12.1.1/32 is directly connected, 03:01:16, GigabitEthernet0/0/0/0
C    10.13.1.0/24 is directly connected, 00:32:56, GigabitEthernet0/0/0/1.13
L    10.13.1.1/32 is directly connected, 00:32:56, GigabitEthernet0/0/0/1.13
S    10.33.33.0/24 [1/0] via 10.13.1.3, 00:00:30
C    172.16.14.0/24 is directly connected, 00:22:05, GigabitEthernet0/0/0/1.14
L    172.16.14.1/32 is directly connected, 00:22:05, GigabitEthernet0/0/0/1.14
C    172.16.15.0/24 is directly connected, 00:22:05, GigabitEthernet0/0/0/1.15
L    172.16.15.1/32 is directly connected, 00:22:05, GigabitEthernet0/0/0/1.15
L    192.168.1.1/32 is directly connected, 03:01:16, Loopback0
L    192.168.11.11/32 is directly connected, 00:32:56, Loopback1
```

Example 2-3 verifies that XR1 has connectivity to the 10.33.33.33 IP address.

Example 2-3 *Remote End Point Connectivity Check*

```
RP/0/0/CPU0:XR1#ping 10.33.33.33
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.33.33.33, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

Task 2.2: IPv6 Static Route

- Create a static route to the FEC0:22::/64 network with a next-hop of FEC0:12::2
- Verify that you can ping the FECO:22:22 IPv6 address

IPv6 routes are placed under the **address-family ipv6 unicast** of the static routing process. The command structure is the same for the static route. Example 2-4 provides verification.

Example 2-4 *Remote End Point Connectivity Check*

```
RP/0/0/CPU0:XR1(config)#router static
RP/0/0/CPU0:XR1(config-static)#address-family ipv6 unicast
RP/0/0/CPU0:XR1(config-static-afi)#fec0:22::/64 fec0:12::2
RP/0/0/CPU0:XR1(config-static-afi)#commit
```

Example 2-5 shows the IPv6 static route entry in the IPv6 routing table with the command **show route ipv6**.

Example 2-5 *Verification of IPv4 Route Table*

```
RP/0/0/CPU0:XR1#show route ipv6

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       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, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR
       A - access/subscriber, a - Application route, (!) - FRR Backup path

Gateway of last resort is not set

L    fec0::1/128 is directly connected,
      03:15:22, Loopback0
C    fec0:12::/64 is directly connected,
      03:15:22, GigabitEthernet0/0/0/0
L    fec0:12::1/128 is directly connected,
      03:15:22, GigabitEthernet0/0/0/0
S    fec0:22::/64
      [1/0] via fec0:12::2, 00:02:05
```

Example 2-6 verifies connectivity to the FEC0:22::22 host.

Example 2-6 *Remote End Point Connectivity Check*

```
RP/0/0/CPU0:XR1#ping fec0:22::22
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to fec0:22::22, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/9 ms
```

Note: The command **show run router static** will show you the entire static routing configuration. You can append an address-family to the command to limit the configuration to the specific address-family.

OSPF

Task 3.1: Basic OSPF Configuration

- Create OSPF process 1
  - Enable interface Gi0/0/0/0, Gi0/0/0/1.13, and Loopback 0 for Area 0
  - Enable interface Loopback 1 for Area 1
- Verify OSPF interfaces
- Verify OSPF neighbors

The command **router ospf** *process-id* initializes the OSPF process on an IOS XR router. After initializing the process, OSPF areas are defined with the command **area** *area-id*. Interfaces are then associated with the command **interface** *interface-type interface-number* to the area therefore enabling OSPF on those interfaces. Example 3-1 shows the configuration of OSPF.

Example 3-1 *OSPF Router Configuration*

```
RP/0/0/CPU0:XR1(config)#router ospf 1
RP/0/0/CPU0:XR1(config-ospf)#area 0
RP/0/0/CPU0:XR1(config-ospf-ar)#interface Loopback 0
RP/0/0/CPU0:XR1(config-ospf-ar-if)#interface Gigabit0/0/0/0
RP/0/0/CPU0:XR1(config-ospf-ar-if)#interface Gigabit0/0/0/1.13
RP/0/0/CPU0:XR1(config-ospf-ar-if)#area 1
RP/0/0/CPU0:XR1(config-ospf-ar)#interface Loopback 1
```

The command **show ospf interface** [**brief**] shows relevant OSPF interface information. Example 3-2 shows output of the command with the brief option.

Example 3-2 *Verification of OSPF Interfaces*

```
RP/0/0/CPU0:XR1#show ospf interface brief

* Indicates MADJ interface, (P) Indicates fast detect hold down state

Interfaces for OSPF 1

Interface          PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Lo0                1     0               192.168.1.1/32     1     LOOP  0/0
Gi0/0/0/0          1     0               10.12.1.1/24       1     BDR   1/1
Gi0/0/0/1.13       1     0               10.13.1.1/24       1     BDR   1/1
Lo1                1     1               192.168.11.11/32   1     LOOP  0/0
                                                      o
```

The command **show ospf neighbors** provides the router's OSPF neighbors. Example 3-3 provides sample output.

Example 3-3 *Verification of OSPF Neighbors*

```
RP/0/0/CPU0:XR1#show ospf neighbor

* Indicates MADJ interface

Neighbors for OSPF 1

Neighbor ID     Pri   State           Dead Time   Address    Interface
192.168.2.2     1   FULL/BDR          00:00:37  10.12.1.2   GigabitEthernet0/0/0/0
    Neighbor is up for 00:02:31
192.168.3.3     1   FULL/BDR          00:00:39  10.13.1.3  GigabitEthernet0/0/0/1.13
    Neighbor is up for 00:01:23
Total neighbor count: 2
```

The routing table output for OSPF based routes is shown with the command **show route ospf** as illustrated in Example 3-4. Notice that the routes are intra-area and inter-area routes.

Example 3-4 *Verification of OSPF Route Table*

```
RP/0/0/CPU0:XR1#show route ospf

O IA 10.2.2.2/32 [110/51] via 10.12.1.2, 00:03:38, GigabitEthernet0/0/0/0
O IA 10.3.3.3/32 [110/51] via 10.13.1.3, 00:02:34, GigabitEthernet0/0/0/1.13
O    192.168.2.2/32 [110/51] via 10.12.1.2, 00:03:42, GigabitEthernet0/0/0/0
O    192.168.3.3/32 [110/51] via 10.13.1.3, 00:02:34, GigabitEthernet0/0/0/1.13
```

Task 3.2: Global Inheritance

- Configure OSPF so that all interfaces will have a cost of 100
- Verify the OSPF cost for all interfaces

IOS XR's configuration is 100% hierarchical. For example, all features and options for a routing protocol reside within the routing protocol configuration. Parameters that are set at the global process for the routing protocol are inherited by the more specific components of the protocol. Setting the OSPF cost parameter globally will set the cost for all interfaces on a router.

The command **cost** *value* sets the OSPF cost as shown in Example 3-5.

Example 3-5 *Global Cost Setting*

```
RP/0/0/CPU0:XR1(config)#router ospf 1
RP/0/0/CPU0:XR1(config-ospf)#cost 100
RP/0/0/CPU0:XR1(config-ospf)#commit
```

Example 3-6 verifies that all interfaces now have the cost of 100 because it was set globally in the OSPF process.

Example 3-6 *Verification of OSPF Cost Settings*

```
RP/0/0/CPU0:XR1#show ospf interface brief

* Indicates MADJ interface, (P) Indicates fast detect hold down state

Interfaces for OSPF 1

Interface          PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Lo0                1     0               192.168.1.1/32     100   LOOP  0/0
Gi0/0/0/0          1     0               10.12.1.1/24       100   BDR   1/1
Gi0/0/0/1.13       1     0               10.13.1.1/24       100   BDR   1/1
Lo1                1     1               192.168.11.11/32   100   LOOP  0/0
```

Task 3.3: Area Inheritance

- Configure OSPF so that all interfaces in Area 0 will have a cost of 50
- Verify the OSPF cost for all interfaces

Example 3-7 demonstrates setting the cost at the area context within OSPF.

Example 3-7 *Area Cost Configuration*

```
RP/0/0/CPU0:XR1(config)#router ospf 1
RP/0/0/CPU0:XR1(config-ospf)#area 0
RP/0/0/CPU0:XR1(config-ospf-ar)#cost 50
RP/0/0/CPU0:XR1(config-ospf-ar)#commit
```

Hierarchical configurations allow for exceptions, by allowing lower-level components to pre-empt inheritance. In Example 3-8, all interfaces within Area 0 now have an OSPF cost of 50 versus 100 that was set globally.

Example 3-8 *Verification of OSPF Cost Settings*

```
RP/0/0/CPU0:XR1#show ospf interface brief

* Indicates MADJ interface, (P) Indicates fast detect hold down state

Interfaces for OSPF 1

Interface          PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Lo0                1     0               192.168.1.1/32     50    LOOP  0/0
Gi0/0/0/0          1     0               10.12.1.1/24       50    BDR   1/1
Gi0/0/0/1.13       1     0               10.13.1.1/24       50    BDR   1/1
Lo1                1     1               192.168.11.11/32   100   LOOP  0/0
```

Task 3.4: Interface Inheritance

- Configure OSPF so that Loopback 0 has a cost of 10
- Verify the OSPF cost for all interfaces

Example 3-9 shows the OSPF cost set to 10 on the interface Loopback 0.

Example 3-9 *Interface Cost Configuratoin*

```
RP/0/0/CPU0:XR1(config)#router ospf 1
RP/0/0/CPU0:XR1(config-ospf)#area 0
RP/0/0/CPU0:XR1(config-ospf-ar)#interface loopback 0
RP/0/0/CPU0:XR1(config-ospf-ar-if)#cost 10
RP/0/0/CPU0:XR1(config-ospf-ar-if)#commit
```

Example 3-10 re-iterates the concept of pre-emption down to the interface level. Notice that Loopback 0 now has a cost of 10, while other interfaces in Area 0 have a cost of 50; and all other interfaces on the router have a cost of 100.

Inheritance within OSPF can save a lot of time with specific parameters such timers, authentication, etc.

Example 3-10 *Verification of OSPF Cost Settings*

```
RP/0/0/CPU0:XR1#show ospf interface brief

* Indicates MADJ interface, (P) Indicates fast detect hold down state

Interfaces for OSPF 1

Interface          PID   Area            IP Address/Mask    Cost  State Nbrs F/C
Lo0                1     0               192.168.1.1/32     10    LOOP  0/0
Gi0/0/0/0          1     0               10.12.1.1/24       50    BDR   1/1
Gi0/0/0/1.13       1     0               10.13.1.1/24       50    BDR   1/1
Lo1                1     1               192.168.11.11/32   100   LOOP  0/0
```

BGP

Task 4.1: iBGP

- Create the BGP process with the **router bgp** *ASN* command.
- Configure BGP using the Autonomous System (AS) number 100
- Configure iBGP neighbor with 192.168.2.2 sourcing the session from loopback0 interface
- Verify iBGP neighborship
- Verify BGP routes in BGP table

IOS XR maintains a process specifically for BGP. The process initializes with the command **router BGP**. IPv4 routes are advertised under the **address-family ipv4 unicast** section and the attributes for the neighbors are defined under the **address-family ipv4 unicast** section of the neighbor configuration as shown below in Example 4-1.

Example 4-1 *IPv4 Static Route Configuration*

```
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-bgp-af)#network 192.168.1.1/32
RP/0/0/CPU0:XR1(config-bgp-af)#exit
RP/0/0/CPU0:XR1(config-bgp)#neighbor 192.168.2.2
RP/0/0/CPU0:XR1(config-bgp-nbr)#remote-as 100
RP/0/0/CPU0:XR1(config-bgp-nbr)#update-source loopback 0
RP/0/0/CPU0:XR1(config-bgp-nbr)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#next-hop-self
```

Example 4-2 verifies the BGP IPv4 neighborship using **show bgp ipv4 unicast summary**. Notice that the BGP process on XR1 is running as AS 100, and that we have received one prefix from 192.168.2.2.

Example 4-2 *Verification of iBGP Neighborship*

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast summary
BGP router identifier 192.168.1.1, local AS number 100
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 4
BGP main routing table version 4
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               4          4          4          4           4           4

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
192.168.2.2       0   100       9       9        4    0    0 00:05:07          1
```

Example 4-3 displays the IPv4 BGP table with the command **show bgp ipv4 unicast.** Notice that the route 192.168.1.1/32 is the locally advertised route from XR1 along with the 192.168.2.2/32 route learned from XR2.

Example 4-3 *Verification of BGP Table for Routes learnt*

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
Wed Jan 29 00:08:05.627 UTC
BGP router identifier 192.168.1.1, local AS number 100
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 2
BGP main routing table version 2
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*  192.168.1.1/32     0.0.0.0                  0         32768 i

Processed 1 prefixes, 1 paths
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
Wed Jan 29 00:12:23.239 UTC
BGP router identifier 192.168.1.1, local AS number 100
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 4
BGP main routing table version 4
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 192.168.1.1/32     0.0.0.0                  0         32768 i
*>i192.168.2.2/32     192.168.2.2              0    100      0 i

Processed 2 prefixes, 2 paths
```

Task 4.2: eBGP

- Configure an eBGP session between XR1 and 172.16.14.4 using remote-as 200
- Make XR1 appear like it is AS 150 with the '**local-as**' feature
- Verify eBGP neighborship
- Verify that routes are passing

eBGP configuration is similar to IOS nodes except that a different remote-as is used. By default XR will not transfer routes between external BGP sessions as we will show you below in Example 4-4.

Example 4-4 *eBGP Configuration*

```
RP/0/0/CPU0:XR1(config)# router bgp 100
RP/0/0/CPU0:XR1(config-bgp)# neighbor 172.16.14.4
RP/0/0/CPU0:XR1(config-bgp-nbr)# remote-as 200
RP/0/0/CPU0:XR1(config-bgp-nbr)# local-as 150
RP/0/0/CPU0:XR1(config-bgp-nbr)# address-family ipv4 unicast
```

Example 4-5 verifies the eBGP IPv4 neighborship using **show bgp ipv4 unicast summary**. Notice that there are no prefixes exchanged by 172.16.4.4 and that an exclamation is present indicating a problem.

Example 4-5 *Verifying eBGP neighbors*

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast summary
Wed Jan 29 01:21:00.037 UTC
BGP router identifier 192.168.1.1, local AS number 100
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 4
BGP main routing table version 4
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               4          4          4          4           4           4

Some configured eBGP neighbors (under default or non-default vrfs)
do not have both inbound and outbound policies configured for IPv4 Unicast
address family. These neighbors will default to sending and/or
receiving no routes and are marked with '!' in the output below.
Use the 'show bgp neighbor <nbr_address>' command for details.

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
172.16.14.4       0   200       2       2        0    0    0 00:00:02          0!
192.168.2.2       0   100      77      77        4    0    0 01:13:38          1
```

Note: With the above configuration, you will be able to bring the eBGP neighborship up but there will be no route exchange between the two peers. The reason behind this is, In IOS XR External BGP (eBGP) neighbors must have an inbound and outbound policy configured. If no policy is configured, no routes are accepted from the neighbor, nor are any routes advertised to it. This added security measure ensures that routes cannot accidentally be accepted or advertised in the case of a configuration omission error.

Let's examine the neighbor status to see exactly what is happening. Example 4-6 shows attributes with the BGP neighbor 172.16.14.4 with the command **show bgp neighbor** *IP-address*. Notice that no route-policy exists and NLRIs will be dropped. A route-policy needs to be associated under the IPv4 Unicast Address family for this neighbor so that routes can be exchanged.

Example 4-6 *Verifying eBGP neighbors*

```
RP/0/0/CPU0:XR1#show bgp neighbor 172.16.14.4

BGP neighbor is 172.16.14.4
 Remote AS 200, local AS 150, external link
 Remote router ID 192.168.2.2
  BGP state = Established, up for 1d05h
  Last read 00:00:14, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:14, attempted 33, written 33
  Second last write 00:00:16, attempted 23, written 23
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Jan 31 15:43:44.348 last full not set pulse count 3561
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Enforcing first AS is enabled
  Multi-protocol capability received
  Neighbor capabilities:
    Route refresh: advertised (old + new) and received (old + new)
    4-byte AS: advertised and received
    Address family IPv4 Unicast: advertised and received
  Received 1780 messages, 0 notifications, 0 in queue
  Sent 1787 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 30 secs

 For Address Family: IPv4 Unicast
  BGP neighbor version 6
  Update group: 0.2 Filter-group: 0.2  No Refresh request being processed
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 2
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 2.
    No policy: 1, Failed RT match: 0
    By ORF policy: 0, By policy: 1
  Prefix advertised 6, suppressed 0, withdrawn 2
  Maximum prefixes allowed 1048576
  Threshold for warning message 75%, restart interval 0 min
  An EoR was received during read-only mode
  Last ack version 0, Last synced ack version 0
  Outstanding version objects: current 1, max 2
  Additional-paths operation: None

  Connections established 1; dropped 0
  Local host: 172.16.14.1, Local port: 63387
  Foreign host: 172.16.14.4, Foreign port: 179
  Last reset 00:00:00
```

Task 4.3: Basic Route-Policy

- Create the route-policy PASS-ALL permitting all prefixes
- Apply the route-policy pass-all in inbound and outbound direction on the eBGP neighbor 172.16.14.4 for *address-family ipv4 unicast*.
- Verify the routes exchanged between eBGP peers

Route-Policies can be created using the configuration **route-policy** *Name*. The route-policy is then associated to the BGP neighbor with in and out policies. Creation of a route-policy and linking to the BGP peer is shown in Example 4-7.

Example 4-7 *Route-Policy Configuration*

```
RP/0/0/CPU0:XR1(config)#route-policy PASS-ALL
RP/0/0/CPU0:XR1(config-rpl)#pass
RP/0/0/CPU0:XR1(config-rpl)#end-policy
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#neighbor 172.16.14.4
RP/0/0/CPU0:XR1(config-bgp-nbr)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#route-policy PASS-ALL in
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#route-policy PASS-ALL out
```

Example 4-8 verifies t that XR1 is receiving routes from it's eBGP peer 172.16.14.4.

Example 4-8 *Verifying eBGP neighor with Route-Policy*

```
RP/0/0/CPU0:XR1#show bgp
BGP router identifier 192.168.1.1, local AS number 100
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 9
BGP main routing table version 9
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 172.16.14.0/24     172.16.14.4              0             0 150 200 ?
*> 172.16.44.0/24     172.16.14.4              0             0 150 200 65000 ?
*> 192.168.1.1/32     0.0.0.0                  0         32768 i
*>i192.168.2.2/32     192.168.2.2              0    100      0 i
`*> 192.168.4.4/32     172.16.14.4              0             0 150 200 i

Processed 5 prefixes, 5 paths
```

Task 4.4: Route-Policy with If-Else

- Create a route-policy called BLOCK-AS65000 to filter routes with AS 65000 in the AS_PATH
  Use IOS regex processing for this task
- Apply the route-policy under BGP neighbor 172.16.14.4 address-family ipv4 unicast
- Verify the policy is working

IOS XR Route-Policy allows you to perform conditional filtering to help filter the routes. This can be done using the **If-Else** statement under the route-policy. The if statement matches if there is any prefix which has AS 65000 in the AS-PATH, then it will drop the prefix else will allow the prefix. Example 4-9 provides the relevant configuration.

Example 4-9 *Route-Policy with If-Else*

```
RP/0/0/CPU0:XR1(config)#route-policy BLOCK-AS65000
RP/0/0/CPU0:XR1(config-rpl)# if as-path in (ios-regex '_65000_') then
RP/0/0/CPU0:XR1(config-rpl-if)# drop
RP/0/0/CPU0:XR1(config-rpl-if)# else
RP/0/0/CPU0:XR1(config-rpl-else)# pass
RP/0/0/CPU0:XR1(config-rpl-else)# endif
RP/0/0/CPU0:XR1(config-rpl)#end-policy
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#neighbor 172.16.14.4
RP/0/0/CPU0:XR1(config-bgp-nbr)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#route-policy BLOCK-AS65000 in
```

Example 4-10 verifies that the route-policy is working as intended because there are no entries in the BGP table that have AS 65000 in the AS_Path anymore.

Example 4-10 *Verifying Route-Policy is working fine*

```
RP/0/0/CPU0:XR1#show bgp
Sat Feb  1 14:45:08.760 UTC
BGP router identifier 192.168.1.1, local AS number 100
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 10
BGP main routing table version 10
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 172.16.14.0/24     172.16.14.4              0             0 150 200 ?
*> 192.168.1.1/32     0.0.0.0                  0         32768 i
*>i192.168.2.2/32     192.168.2.2              0    100      0 i
*> 192.168.4.4/32     172.16.14.4              0             0 150 200 i

Processed 4 prefixes, 4 paths
RP/0/0/CPU0:XR1#
```

Appendix A: XR2 Initial Configuration

After logging into the second XRv router, copy and paste the entire contents of the cell below into XR2's console.

```
Configure terminal

hostname XR2
cdp timer 5
cdp
vrf R3
 address-family ipv4 unicast
  import route-target
   100:3
  !
  export route-target
   100:3
  !
 !
 address-family ipv6 unicast
  import route-target
   100:3
  !
  export route-target
   100:3
  !
 !
vrf R4
 address-family ipv4 unicast
  import route-target
   100:4
  !
  export route-target
   100:4
  !
 !
 address-family ipv6 unicast
  import route-target
   100:4
  !
  export route-target
   100:4
  !
 !
vrf R5
 address-family ipv4 unicast
  import route-target
   100:5
  !
  export route-target
   100:5
  !
 !
 address-family ipv6 unicast
  import route-target
   100:5
  !
  export route-target
   100:5
  !
 !
!
interface Loopback0
 description BGP-Peering
 ipv4 address 192.168.2.2 255.255.255.255
 ipv6 address fec0::2/128
!
```

```
interface Loopback1
 description OSPF-Route
 ipv4 address 10.2.2.2 255.255.255.0
 ipv6 address fec0:2::2/64
!
interface Loopback2
 description Static-Route
 ipv4 address 10.22.22.22 255.255.255.0
 ipv6 address fec0:22::22/64
!
interface Loopback30
 description BGP-Peering
 vrf R3
 ipv4 address 192.168.3.3 255.255.255.255
 ipv6 address fec0::3/128
!
interface Loopback31
 description OSPF-Route
 vrf R3
 ipv4 address 10.3.3.3 255.255.255.0
 ipv6 address fec0:3::3/64
!
interface Loopback32
 description Static-Route
 vrf R3
 ipv4 address 10.33.33.33 255.255.255.0
 ipv6 address fec0:33::3/64
!
interface Loopback40
 description BGP-Peering
 vrf R4
 ipv4 address 192.168.4.4 255.255.255.255
 ipv6 address fec0::4/128
!
interface Loopback44
 vrf R4
 ipv4 address 172.16.44.44 255.255.255.0
!
interface Loopback50
 description BGP-Peering
 vrf R5
 ipv4 address 192.168.5.5 255.255.255.255
 ipv6 address fec0::5/128
!
interface GigabitEthernet0/0/0/0
 cdp
 ipv4 address 10.12.1.2 255.255.255.0
 ipv6 address fec0:12::2/64
!
interface GigabitEthernet0/0/0/1
 cdp
!
interface GigabitEthernet0/0/0/1.13
 vrf R3
 ipv4 address 10.13.1.3 255.255.255.0
 ipv6 address fec0:13::3/64
 encapsulation dot1q 13
!
interface GigabitEthernet0/0/0/1.14
 vrf R4
 ipv4 address 172.16.14.4 255.255.255.0
 ipv6 address fec0:14::4/64
 ipv6 address fec0:172:14::4/64
 encapsulation dot1q 14
!
interface GigabitEthernet0/0/0/1.15
 vrf R5
 ipv4 address 172.16.15.5 255.255.255.0
 ipv6 address fec0:15::5/64
 ipv6 address fec0:172:15::5/64
```

```
 encapsulation dot1q 15
!
route-policy PREPEND
  if destination in (172.16.44.0/24) then
    prepend as-path 65000
  endif
  pass
end-policy
!
route-policy PASS-ALL
  pass
end-policy
!
router ospf 1
 area 0
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/0
  !
 !
 area 2
  interface Loopback1
  !
 !
 vrf R3
  router-id 192.168.3.3
  area 0
   interface Loopback30
   !
   interface GigabitEthernet0/0/0/1.13
   !
  !
  area 3
   interface Loopback31
   !
   interface GigabitEthernet0/0/0/1.14
   !
  !
 !
router bgp 100
 address-family ipv4 unicast
  network 192.168.2.2/32
 !
 address-family vpnv4 unicast
 !
 neighbor 192.168.1.1
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
   next-hop-self
  !
 !
 vrf R4
  rd 100:4
  address-family ipv4 unicast
   network 192.168.4.4/32
   redistribute connected
  !
  neighbor 172.16.14.1
   remote-as 150
   local-as 200 no-prepend replace-as
   address-family ipv4 unicast
    route-policy PASS-ALL in
    route-policy PREPEND out
   !
  !
 Commit replace
```

At the prompt for the commit replace, please type in yes.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**