Table of Contents

31. august 2023 10:04

Password Recovery, IOS

13. november 2015 09:06

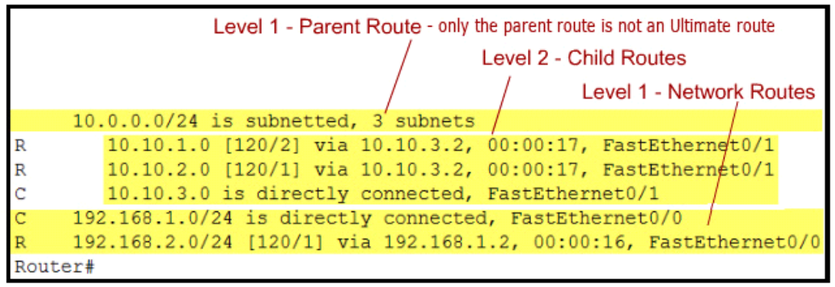
```
!-------------------------------------
! Password Recovery - Router
!-------------------------------------
config-register|confreg
0x2102|0x2142
reset
alt+b in Tera Term -> ROMMON

To recover or reset the enable password on the
Cisco 1900 series router, complete the following
steps:
  1. Establish a terminal connection with the
     router using Tera Term or other terminal
     emulator.
  2. Boot to ROMMON by either removing flash and
     rebooting or selecting Alt-b during a
     reboot.
  3. Type confreg 0x2142 at the rommon prompt.
  4. Type reset at the next rommon prompt.
  5. Type no at the initial configuration dialog.
  6. Type enable at the router prompt.
  7. Type copy startup-config running-config to
     load the startup configuration.
  8. Type show running-config.
  9. Record an unencrypted enable password. Reset
     an encrypted enable password.
 10. In configuration mode, type config-register
     0x2102.
 11. In privilege mode, type copy running-config
     startup-config in order to save
     configuration.
 12. Use the show version command to verity
     configuration register settings.
```

```
!-------------------------------------------------------
! Password Recovery - Switch
!-------------------------------------------------------
press MODE button (within 15 seconds after power on)
 - LED flashing green
 - LED turns briefly amber and then solid green
 - release the Mode button

 follow instruction

  1. flash_init
  2. [load_helper]
  3. rename flash:config.text flash:config.text.old
  4. boot
  5. rename flash:config.text.old flash:config.text

!-------------------------------------------------------
! Manage IOS and Config on Switch
!-------------------------------------------------------
sh version
sh flash
cd flash:/<dir>

copy flash tftp
copy tftp flash

copy startup-config tftp
erase nvram
copy tftp startup-config

sh bootvar (old cmd sh boot)
rename flash:/… flash:/
dir flash:/

delete flash:/…/html/*

archive tar /x tftp://<ip-of-tftp>/<ios-name> flash:/<dir>

boot system flash:/path/IOSname
```

Basics

16. december 2015 10:03

```
!-------------------------------------
! Connect to the router/switch:
!-------------------------------------
Console settings: 9600,8,n,1,n

!-------------------------------------
! Interface
!-------------------------------------
int <>
 description <>
 bandwidth <Kbps>
 ip address <net> <sm>

!-------------------------------------
! DCE interface
!-------------------------------------
int serial <>
 clock rate [speed]

!-------------------------------------
! Basic setup
!-------------------------------------
hostname <>
service password-encryption
security passwords min-length <no>
enable secret <pw>
no ip domain-lookup
banner motd #<text>#
!
line con 0
 logging synchronous
 exec-timeout <min> <sec>
 password <>
 login
 exit
!
line vty 0 15
 logging synchronous
 exec-timeout <min> <sec>
 password <>
 login
end

!-------------------------------------
! Initializing and Reloading a Router and Switch
!-------------------------------------
enable
erase startup-config
[delete vlan.dat]
reload

!-------------------------------------
! Pipe commands
!-------------------------------------
sh ver | include <word>
sh startup-config | begin vty
sh run | section ip route
```



In the Cisco CCNA you will need to be able to identify the following types of routes in the routing table:

Level 1 route
is a **network route**, a **default route**, a supernet route or an **ultimate route**.

Network Route
is an ultimate route with an exit interface.

Ultimate route
an ultimate route is a route that ***has an exit interface or a next hop IP address.***

Parent route
is a classful route, but it is not an ulltimate route. A parent route has subnetted child routes. If there are no child routes there is no parent route. Parent routes ***do not have an exit interface or next hop IP address***. A parent route is also called a level 1 route.

Default Route
also known as a "gateway of last resort," is a route configured to the 0.0.0.0 /0 network and mask. This route does not have to qualify or "match" the destination network therefore it is a match for all destinations.

Level 2 route
is a subnetted route with a greater than classful subnet mask, it is also a called a child route.

Child route
A child route is a subnetted route, where the subnet mask is greater than the classful subnet mask (eg. /27 versus /24). A child route is a level 2 route.

```
!-------------------------------------
! Basic Show commands
!-------------------------------------
show version
show running-config
show ip interface brief
show clock

sh interfaces <>
sh interfaces description
sh protocols
sh arp | sh ip arp
sh mac address-table

sh flash or dir flash:
sh ver
sh ip int brief
sh ip route
sh arp
sh cdp neighbors detail
sh run
sh mac address-table
sh file systems
```

```
!-------------------------------------
! On pc:
!-------------------------------------
arp -a       (show all entries)
arp -d *     (delete all ARP entries)
arp -d <IP>  (delete specific entry)
arp -s <IP> <MAC> (insert entry)

netstat -r or
route print (display host routing table)

ipconfig /all
ipconfig /flushdns

!-------------------------------------
! extended ping on PC
!-------------------------------------
ping -t <IP>

!-------------------------------------
! Domain lookup
!-------------------------------------
nslookup <domain>
or
nslookup <ENTER>
> www.google.com
> set type=mx
> cisco.com

!-------------------------------------
! allow icmp traffic through a
! windows firewall
!-------------------------------------
windows firewall
 -> Advanced settings
Inbound Rules
 -> New Rule
Rule Type
 -> Custom
Protocol and Ports
 -> Protocol type = ICMPv4 or ICMPv6
Scope
 -> Wich local IP..  = Any IP address
 -> Wich remote IP.. = Any IP address
Name
 -> <Give rule a name>
```

```
!-------------------------------------
! extended ping in CLI
!-------------------------------------
ping <ENTER>
ping <IP> repeat 500

ping -n 25 www.afrinic.net > afrinic.txt
more afrinic.txt
dir *.txt
```

Basic IPv6

24. maj 2016 13:34

```
!---------------------------------------
! IPv6 on interface
!---------------------------------------
int <>
 description <>
 ipv6 address <>/64
 ipv6 address <fe80::x> link-local

!-----------------------------------
! change the link-local address
!-----------------------------------
interface s1/0
 ipv6 address FE80::1 link-local

ping FE80::1
show ipv6 interface serial 1/0

!-----------------------------------
! Configure EUI-64 address
!-----------------------------------
interface fa 0/0
 ipv6 address FEC0:23::/64 eui-64

show ipv6 interface fa 0/0
sh ipv6 interface brief

cisco best practice is to manualy assign the
link-local address
    -FE80::1 link-local

!-----------------------------------
! Configure IPv6 address
!-----------------------------------
int <eth>
 ipv6 address <>/64 eui-64
 no shut

!unique-local addres recommended for p2p ser
int <ser>
 ipv6 address FC00::1/64
 no shut

IPv6 SLAAC
(Stateless Address Autoconfiguration)

By default, Cisco routers send RA messages
every 200 seconds. RA messages are always
sent to the IPv6 all-nodes multicast address
FF02::1

Router Solicitation (RS)- Search
Router Adverticement (RA)- Inform
Neighbor Solicitation (NS)- Search
Neighbor Adverticement (NA)- Inform

ping -6 <> !ipv6 ping
```

IPv6:
128bit
8 hextets
::1 (**loopback address**)
FE80:… (**link-local address**)

all routers multicast group (FF02::2**)**
This will allow the PCs to obtain their IP address and default gateway information automatically using Stateless Address Autoconfiguration (SLAAC).

anycast address
used by routers to facilitate load sharing and provide alternate path flexibility if a router becomes unavailable. Only routers should respond to an anycast address.

Site-local address
site-local addresses have been deprecated and replaced by unique-local addresses. Site-local addresses were identified by the numbers **FECO** in the initial hextet.

First Hextet (Far Left)	Type of IPv6 Address
0000 to 00FF	Loopback address, any address, unspecified address, or IPv4-compatible
2000 to 3FFF	Global unicast address (a routable address in a range of addresses that is currently being handed out by the Internet Assigned Numbers Authority [IANA])
FE80 to FEBF	Link-local (a unicast address which identifies the host computer on the local network)
FC00 to FCFF	Unique-local (a unicast address which can be assigned to a host to identify it as being part of a specific subnet on the local network)
FF00 to FFFF	Multicast address

The IPv6 global prefix 2001:DB8::/32 is a reserved prefix for use in documentation, as described in RFC 3849.

The default bias template used by the Switch Database Manager (SDM) does not provide IPv6 address capabilities. Verify that SDM is using either the '**dual-ipv4-and-ipv6**' template or the '**lanbaserouting**' template. The new template will be used after reboot even if the config is not saved.

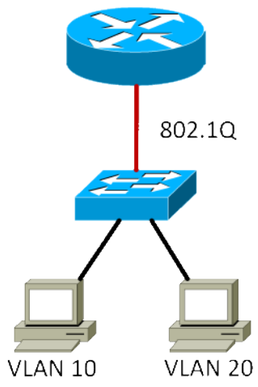
```
S1# show sdm prefer
```

Follow these steps to assign the dual-ipv4-and-ipv6 template as the default SDM template:

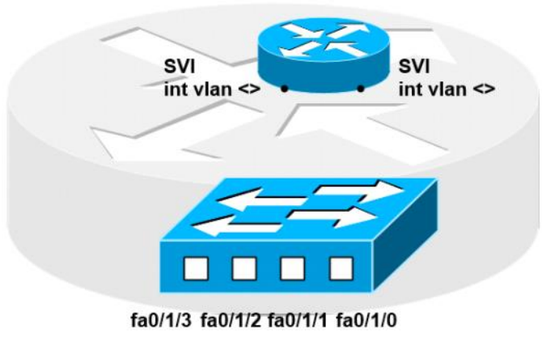
```
S1# configure terminal
S1(config)# sdm prefer dual-ipv4-and-ipv6 default
S1(config)# end
S1# reload


sdm prefer [default|dual-ipv6-and-ipv6|lanbase-routing|qos]

2960 supports up to 750 static routes
sdm prefer lanbase-routing
!remember to reload switch
```

Inter-VLAN Routing

28 August 2017 11:28




```
!--------------------------------------
! Inter-VLAN routing (router setup)
! Router-on-a-stick
!--------------------------------------
int fa 0/0
 no shutdown
!
int fa 0/0.<vlan-id>
 encapsulation dot1q <vlan-id> [native]
 ip address <IP> <SM>
```




```
!--------------------------------------
! Inter-VLAN routing
! (4 port switchmodul in router)
!--------------------------------------
ip routing

int vlan <x>
 description GW FOR VLAN X
 ip address <NET+SM>
!
int vlan <y>
 description GW FOR VLAN Y
 ip address <NET+SM>
!
int fa 0/1/0
 switchport mode access
 switchport access vlan <x>
!
int fa 0/1/1
 switchport mode access
 switchport access vlan <y>
```

NAT

31. august 2020 13:50

```
!-------------------------------------------------------
! Traditional NAT
! if port nat inside and NAT translation match in
! table -> route to outside int -> translate -> send
!-------------------------------------------------------
!
!-------------------------------------------------------
! Enable interface for NAT(Network Address Translation)
!-------------------------------------------------------
int <outside-int>
 ip nat outside
int <inside-int>
 ip nat inside

!-------------------------------------------------------
! Static NAT
!-------------------------------------------------------
ip nat inside source static <inside-IP> <outside-IP>

! Static NAT with port forwarding
ip nat inside source static {tcp | udp} <local-ip local-port> <global-ip global-port>

!-------------------------------------------------------
! Dynamic NAT - NAT with pool
!-------------------------------------------------------
ip access-list standard <ACL-NAME> !(or use numbered ACL)
 permit <source-net> <wm>

ip nat pool <Pool-NAME> <start-ip> <end-ip> prefix-length <>
ip nat inside source list <ACL-NAME> pool <POOL-NAME>

!-------------------------------------------------------
! NAT overload - PAT (Port Address Translation)
!-------------------------------------------------------
ip access-list standard <ACL-NAME> !(or use numbered ACL)
 permit <source-net> <wm>

ip nat inside source list <ACL-NAME> int <outside-int> overload


!-------------------------------------------------------
! If needed switch standard ACL with extended ACL
!-------------------------------------------------------
ip access-list extended <NAME>
 remark Permit Local LAN to use NAT
 permit ip <LAN-net> <wm> any
 exit


sh ip nat translations
sh ip nat statistics
debug ip nat
clear ip nat translation *
clear ip nat statistics
```

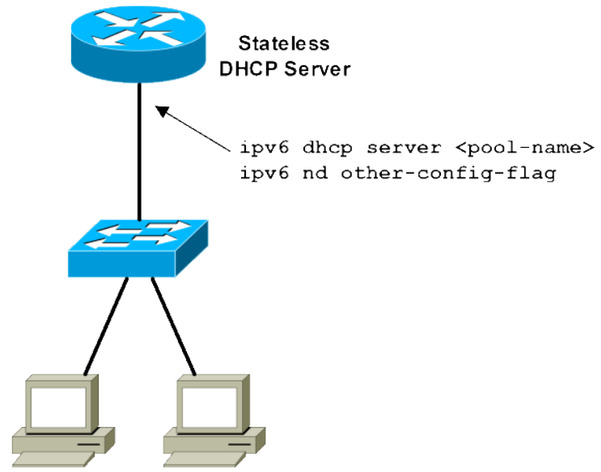
```
!-------------------------------------------------------
!
!-------------------------------------------------------
```

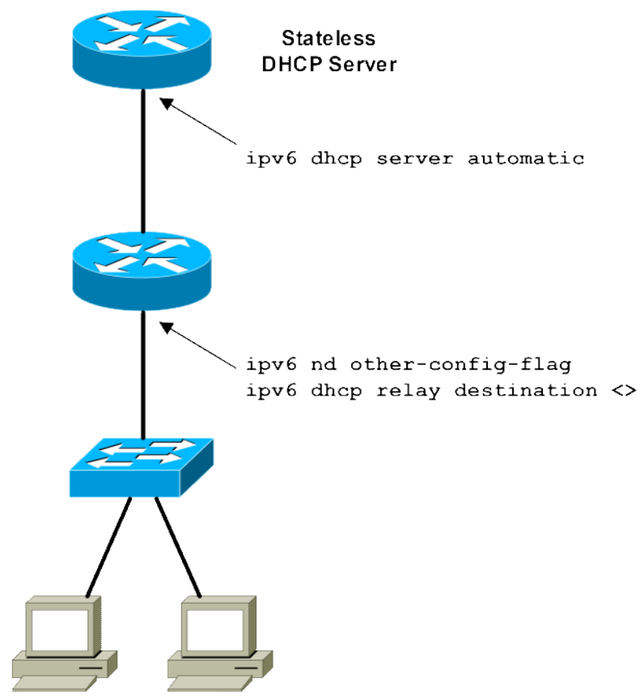
```
!-------------------------------------------------------
! NAT with NVI interface
! - NVI is used for NAT between different VRFs
!
! if port enabled for nat and NAT translation match in table
! -> route to NVI -> translate -> route
! (route-translate-route)
!-------------------------------------------------------
ip access-list extended <NAME>
 remark Permit Local LAN to use NAT
 deny ip <LAN-net> <wm> <LAN-net> <wm>
 permit ip <LAN-net> <wm> any
 exit
!
ip nat pool <Pool-NAME> <start-ip> <end-ip> prefix-length <> [add-route]
ip nat source list <ACL> pool <pool-name>
ip nat source static [tcp…] <inside-ip> [port] <outside-ip> [port]


!-------------------------------------------------------
! Enable interface for NAT
!-------------------------------------------------------
int <>
 ip nat enable

sh ip nat nvi statistics
sh ip nat nvi translations
```

DHCP (IPv4 and IPv6)

25. november 2019 14:10

```
!-----------------------------------------
! dhcp server (IPv4)
!-----------------------------------------
!service dhcp is enabled by default
service dhcp !also enables relay agent function

ip dhcp excluded-address <start-IP> <end-IP>
ip dhcp pool <NAME>
 network <net> <sm>
 dns-server <dns-IP>
 default-router <gw-IP>
 domain-name <>
 lease <days>

!-----------------------------------------
! relay agent
!-----------------------------------------
interface fa 0/0
 ip helper-address <dhcp-IP>

!-----------------------------------------
! Provider Assigned IPv4 address with DHCP
!-----------------------------------------
interface fa 0/0
 ip address dhcp

sh ip dhcp binding
sh ip dhcp pool
debug ip dhcp server events
sh ip dhcp server statistics
sh ip dhcp conflict
sh run | section dhcp
```






```
!-----------------------------------
! Stateless DHCP Server (IPv6)
!-----------------------------------
ipv6 unicast-routing
!
ipv6 dhcp pool <name>
 [link-address <relay-agent-IP>]
 dns-server <ip>
 domain-name <>
!
int <>
 ipv6 dhcp server <pool-name>
 ipv6 nd other-config-flag ! O flag = 1

!-----------------------------------
! use router as Stateless DHCPv6 Client
!-----------------------------------
int <>
 ipv6 enable
 ipv6 address autoconfig !(default)

sh ipv6 dhcp pool
sh ipv6 interface
debug ipv6 dhcp detail

!-----------------------------------
! Stateful DHCP Server
!-----------------------------------
ipv6 unicast-routing
ipv6 dhcp pool <name>
 address prefix <ipv6-prefix> [lifetime {<time>|infinite}]
 dns-server <ip>
 domain-name <>
int <>
 ipv6 dhcp server <pool-name>
 ipv6 nd managed-config-flag ! M flag = 1

sh ipv6 dhcp pool
sh ipv6 dhcp binding
sh ipv6 int <>

!-----------------------------------
! use router as Stateful DHCPv6 Client
!-----------------------------------
int <>
 ipv6 enable
 ipv6 address dhcp

!-----------------------------------
! DHCPv6 relay agent
!-----------------------------------
!int <>
 ipv6 dhcp relay destination <>
 end

sh ipv6 dhcp int <>
```

M	O	Function
0	0	SLAAC
0	1	Stateless DHCPv6
1	-	Statefull DHCPv6

```
!reset M & O flag to default
no ipv6 nd managed-config-flag
no ipv6 nd other-config-flag
```

ACL

19. marts 2015 09:41

```
!-------------------------------------------------------------------------
! Standard ACL (only look at source address)
! - place close to destination as default
!-------------------------------------------------------------------------
ip access-list standard <NAME>
 deny <NET> <WM> log
 permit any

int <int>
 ip access-group <NAME> {in|out}

!-------------------------------------------------------------------------
! Extended ACL (look at protocol, source-, dest-address and portno)
! - place close to source as default
!-------------------------------------------------------------------------
ip access-list extended <NAME>
 deny <protocol> <source-NET> <WM> <dest-NET> <WM> [eq <dest-port>] [log]
 permit ip any any

int <int>
 ip access-group <NAME> {in|out}

protocol  = ip, icmp, tcp...
host <IP> = <IP> 0.0.0.0
any       = 0.0.0.0 255.255.255.255

!-------------------------------------------------------------------------
! VTY ACL
!-------------------------------------------------------------------------
ip access-list standard VTY
 permit <NET> <WM>
 permit host <IP>
 deny any [log]

line vty 0 4
 access-class VTY in
!-------------------------------------------------------------------------
! ACL (numbered)
!-------------------------------------------------------------------------
!Standard access-list (1-99, 1300-1999)
access-list <1-99> permit/deny <IPno> WM

!Extended access-list (100-199, 2000-2699)
access-list <100-199> permit/deny <protocol> <sourceIP> WM <destIP> WM [eq portno]

interface ...
  ip access-group <no> in/out

sh access-lists
sh ip access-lists
sh ip int <int>
```

```
!-------------------------------------
! Well Known Port Numbers
!-------------------------------------
20-21 FTP
22    SSH
23    Telnet
25    SMTP
53    DNS
67/68 DHCP
80    HTTP
110   POP3
123   NTP
143   IMAP4
443   HTTPS
```

```
!-------------------------------------
! Bit Calculator
!-------------------------------------
```

192	224	240	248	252	254	255

128	64	32	16	8	4	2	1

```
!-------------------------------------
! IPv6 ACL - now only extended named ACL
!-------------------------------------
The default bias template used by the Switch
Database Manager (SDM) does not provide IPv6
address capabilities. Verify that SDM is using
either the dual-ipv4-and-ipv6 template or the
lanbase-routing template. The new template will
be used after reboot even if the configuration
is not saved.

!-------------------------------------
! assign the dual-ipv4-and-ipv6
! template as the default SDM template
!-------------------------------------
sh sdm prefer
sdm prefer dual-ipv4-and-ipv6 default
reload

!-------------------------------------
! IPv6 ACL
!-------------------------------------
ipv6 access-list <NAME>
 permit…/x…
 deny…

int <>
 ipv6 traffic-filter <NAME> {in|out}

line vty 0 15
 ipv6 access-class <> in
```

Each IPv6 ACL contains *implicit* permit rules to enable IPv6 neighbor discovery.
These rules can be overridden by the user by placing a `deny ipv6 any any` statement at the end of the ACL. If, however, this statement is used, then the administrator must also specifically permit the neighbor discovery process as follows:

```
permit icmp any any nd-na
permit icmp any any nd-ns
deny ipv6 any any

!-------------------------------------
! reset match counters
!-------------------------------------
clear ipv6 access-list <NAME>

sh access-lists [NAME]

!-------------------------------------
! how to edit?
!-------------------------------------
remove from int
!insert new ACE
ipv6 access-list <NAME>
 permit…… sequence <no>
!append
 new ACE without sequence
!delete
 no ACE…
```

RIP, RIPng, Static

3. maj 2016 12:47

```
!---------------------------------------------
!
!---------------------------------------------
```

```
!---------------------------------------
! RIP protocol
! adm distance: 120
! v1-255.255.255.255 v2-224.0.0.9
! UDP 520
! Algorithm: Bellman-Ford
! metric: hop (max 15 hop)
!---------------------------------------
router rip
 version 2
 no auto-summary
 passive-interface default
 no passive-interface <int>
 network <network>

!-------------------------------------
! to originate default route
! remember to make ip route 0.0.0.0/0
! often done on gateway router -> ISP
!-------------------------------------
 default-information originate
```

Note: Sometimes it is necessary to clear the RIP routing process before the default-information originate command will work.
First, try the command clear ip route * . Sometimes this does not work with RIP.
If still not working, save the configuration and then ***reload routers***. Doing this will reset the hardware and both routers will restart the RIP routing process.

```
sh ip protocols
sh ip route
sh ip rip database
clear ip route *
```

RIP	Bellman-Ford	hop
IGRP/**EIGRP**	DUAL	bw+dly
OSPF	Dijkstra SPF	cost

```
!---------------------------------------
! RIPng protocol
! adm distance: 120
! Multicast: FF02::9
! UDP 521
! Algorithm: Bellman-Ford
! metric: hop (max 15 hop)
!---------------------------------------
!
!-----------------------------------------------
! enable IPv6 routing
!-----------------------------------------------
ipv6 unicast-routing

!-----------------------------------------------
! enable rip on interface
!-----------------------------------------------
int <>
 ipv6 rip <process-name> enable

sh ipv6 protocols
sh ipv6 route
 - a RIPng router includes itself in the hop count
   ie. hop=2 is neighbour router
   a connected interface is 0 hop

!-----------------------------------------------
! IPv6 Static route
!-----------------------------------------------
ipv6 route <destination-net> <next-hop-ip>


!-----------------------------------------------
! IPv6 Static default route
!-----------------------------------------------
ipv6 route ::/0 <next-hop-ip>

!-----------------------------------------------
! Originate default route
!-----------------------------------------------
! on interface pointing to internal network
! The originate keyword propagates the default route in addition
! to other routes in R2's routing table
int <>
 ipv6 rip <process-name> default-information {originate|only}

!-----------------------------------------------
! show commands
!-----------------------------------------------
sh ipv6 interface brief
clear ipv6 rip <process-name>
sh ipv6 route rip
sh ipv6 rip
sh ipv6 rip database
sh ipv6 rip <process-name> next-hops
```

```
!-----------------------------------------------
! Static IPv4 routes
!-----------------------------------------------
ip route <network> <sm>  <exit-int> <next-hop-IP>

! default static route
ip route 0.0.0.0 0.0.0.0 <exit-int> <next-hop-IP>
```

Next-hop static IP route	next-hop-ip
Directly connected static IP route	exit intf
Fully specified static IP route	next-hop-ip + exit intf

```
!-----------------------------------------------
! Static IPv6 routes
!-----------------------------------------------
ipv6 route <ipv6-prefix/prefix-length {ipv6-address|interface-type <>}

!default static route
ipv6 route ::/0 {ipv6-address|interface-type <>}
```

EIGRP IPv4

29. marts 2023 12:51

```
!--------------------------------------------------
! EIGRP protocol
! Protocol: 88
! adm distance: Internal=90 - External=170 - Summary=5
! Multicast: 224.0.0.10 or MAC address 01:00:5e:00:00:0a
! Algorithm: DUAL (Diffused Update Algorithm)
! metric: BW+DLY (default)
!--------------------------------------------------
router eigrp <AS>
 eigrp router-id <>
 no auto-summary
 passive-interface {<> | default}
 no passive-interface <int>
 network <net> [<WM>]

!-------------------------------------
! to originate default route
! remember to make ip route 0.0.0.0/0
! often done on gateway router -> ISP
!-------------------------------------
 redistribute static

!-------------------------------------
! Bandwidth
!-------------------------------------
int S x/x
 bandwidth <Kbits>
 ip bandwidth-percent eigrp <AS> <percent>

 !-------------------------------------
 ! Manual summarization
 !-------------------------------------
 ip summary-address eigrp <AS> <NET+SM>

 !-------------------------------------
 ! Manipulate EIGRP timers
 !-------------------------------------
 ip hello-interval eigrp <AS> <SEC> ! default 5 sec
 ip hold-time eigrp <AS> <SEC>      ! default 3x hello

! Show commands
sh ip protocols
sh ip route [<net> <sm>]
sh ip route summary
sh ip route eigrp
sh ip eigrp neighbors
sh ip eigrp interfaces [detail]
sh ip eigrp topology [<prefix>] [all-links]
sh protocols
sh int <>
sh interfaces description

debug eigrp packets
debug ip eigrp <AS>
debug ip egirp summary
```

```
!--------------------------------------------------
! Load balancing
!--------------------------------------------------
!--------------------------------------------------
! disable cef and route caching
! -> per-packet load balancing
! enable cef and route caching
! -> per-destination load balancing
!--------------------------------------------------
no ip cef
int <>
 no ip route-cache

debug ip packet
```

Another way to demonstrate per-packet load balancing, that does not disable CEF, is to use the per-packet load balancing command **ip load-share per-packet** on outgoing interfaces S0/0/0 and S0/0/1

```
!--------------------------------------------------
! enable unequal load balancing
! variance 2 command, which allows unequal-cost load
! balancing bounded by a maximum distance of (2) × (FD)
!--------------------------------------------------
router eigrp <as>
 variance <>
 maximum-path <> !default =4

sh ip route <net>
!look for traffic share count (0=not in use)

sh ip protocols
!look for 'maximum path' and 'maximum metric variance'
```

Advertised Distance/Reported Distance (AD or RD)

- the metric for an EIGRP neighbour router to reach the destination; the metric between the next-hop router and the destination

Feasible Distance (FD=local cost + AD)

- the sum of the AD from the next-hop neighbour, and the cost between the local router and the next-hop router

Successor (lowest FD)

Feasible successor (FS has an AD < current Successor)

Passive Versus **Active** Routes

- Passive is the operational, stable state
- A route is active when it is undergoing recomputation

```
!--------------------------------------------------
! EIGRP Stub
!--------------------------------------------------
Routers configured as stubs do not forward EIGRP learned routes to
other neighbors.
Use the eigrp stub command to configure a router as a stub where
the router directs all IP traffic to a distribution router.

router eigrp <as>
 eigrp stub <> !default connected and summary

  connected      Do advertise connected routes
  leak-map       Allow dynamic prefixes based on the leak-map
  receive-only   Set receive only neighbor
  redistributed  Do advertise redistributed routes
  static         Do advertise static routes
  summary        Do advertise summary routes

receive-only works only alone
usage: ie on router with NAT/PAT

sh run | sec eigrp
sh ip eigrp neighbors detail
```

```
!-----------------------------------------------
!
!-----------------------------------------------
```

EIGRPv6 (Traditional)

29. marts 2023 12:55

```
!-------------------------------------------------
! EIGRP for IPv6 protocol
! adm distance: Internal=90 - External=170 - Summary=5
! Multicast: FF02::A
! Algorithm: DUAL (Diffused Update Algorithm)
! metric: BW+DLY (default)
!-------------------------------------------------
!
!-----------------------------------------------
! enable IPv6 routing
!-----------------------------------------------
ipv6 unicast-routing

!-----------------------------------------------
! enable EIGRP routing on interface
!-----------------------------------------------
int <>
 ipv6 eigrp <as>

!-----------------------------------------------
! Router-ID and other specific EIGRPv6 cmd's
!-----------------------------------------------
ipv6 router eigrp <as>
 eigrp router-id <>
 passive-interface default
 no passive-interface <>
 [no shutdown] !default from IOS 15.2


!-----------------------------------------------
! manual summarisation
!-----------------------------------------------
int <>
 ipv6 summary-address eigrp <as> <net/x>

sh ipv6 route eigrp

!-----------------------------------------------
! default route
!-----------------------------------------------
ipv6 route ::/0 <exit-int> <next-hop-ip>

ipv6 router eigrp <as>
 redistribute static
```

Note: With the use of CEF (Cisco Express Forwarding) it is recommended practice that a next-hop IP address is used instead of an exit-interface. There is a bug in IOS 15.4 that prevents an IPv6 static route with only a next-hop address from being redistributed. A fully specified static route with both an exit-interface and a next-hop address is used in the example.

```
!-----------------------------------------------
! enable IPv6 cef
!-----------------------------------------------
ipv6 cef

sh ipv6 cef [summary]


sh ipv6 eigrp neighbors
sh ipv6 route eigrp
sh ipv6 eigrp topology
sh ipv6 protocols
```

EIGRP Named (Address Family)

23. maj 2016 11:43

```
!-----------------------------------------------
! Named EIGRP using AF (address family)
!-----------------------------------------------
```

What is known as “classic” EIGRP requires separate EIGRP configuration modes and commands for IPv4 and IPv6. Each process is configured separately, **router eigrp** *as-number* for IPv4 and **ipv6 router eigrp** *as-number* for IPv6.

Named EIGRP uses the address family (AF) feature to unify the configuration process when implementing both IPv4 and IPv6

```
ipv6 unicast-routing
!
router eigrp <virtual-instance-name>
 address-family ipv4 unicast autonomous-system <as>
  !{af-interface|eigrp|network|…}
  eigrp router-id <>
  network <net+wm>
  af-interface <>
   passive-interface
   summary-address <net> <sm>
   authentication mode md5 authentication key-chain <>
   exit-af-interface
  topology base
   redistribute static
   exit-af-topology
  exit-address-family

 address-family ipv6 unicast autonomous-system <as>
  eigrp router-id <>
  af-interface <>
   passive-interface
   summary-address <net> <sm>
   authentication mode md5 authentication key-chain <>
   exit-af-interface
  !disable eigrp on interface
  af-interface <>
   shutdown
   exit-af-interface
  topology base
   redistribute static
   exit-af-topology
  exit-address-family
```

By default, all IPv6 interfaces are automatically enabled for EIGRP for IPv6

Redistribution of static routes in named EIGRP is done in topology configuration mode. Topology configuration mode is a subset of an address family. By default, EIGRP has a base topology for each address family. Additional topologies can be configured for Multitopology Routing (MTR) which is used to enable an EIGRP process for a specified topology. MTR is beyond the scope of CCNP.

```
virtual-instance-name does not have to match between neighbors

EIGRP doesn’t start until at least one address family has been
defined (IPv4 or IPv6)
```

OSPFv2

Thursday, 22 December 2022 10.00

```
!--------------------------------------
! OSPF protocol
! Protocol: 89
! adm distance: 110
! Listen to multicast:
!  - (all OSPF routers)
!    224.0.0.5 or
!    MAC address 01:00:5E:00:00:05
!  - (all OSPF DRs)
!    224.0.0.6 or
!    MAC address 01:00:5E:00:00:06
! Algorithm: Dijkstra (SPF)
! Metric: cost (10^8/Bandwidth)
!--------------------------------------
router ospf <process-id>
 router-id <id in DDN>
 passive-interface {<> | default}
 network <net> <wm> area <area no>

 !--------------------------------------
 ! to originate default route
 ! remember to make ip route 0.0.0.0/0
 ! often done on gateway router -> ISP
 !--------------------------------------
 default-information originate [always]

!--------------------------------------
! link cost, hello and dead interval
!--------------------------------------
 ! increase the reference bandwidth
 ! to 10GigE
 ! default reference-bandwidth = 100Mb
 auto-cost reference-bandwidth 10000

int s x/x
 bandwidth x (x in Kbits)
 ! cost = 10^8/BW
or
 ip ospf cost <value>

 ip ospf hello-interval <value>
 ip ospf dead-interval <value>

!--------------------------------------
! show and debug commands
!--------------------------------------
sh ip protocols
sh ip ospf neighbor
sh ip ospf
sh ip ospf database
sh ip ospf interface <>
sh ip ospf interface brief
sh ip route [ospf]
sh ip route static

debug ip ospf ajd
clear ip ospf process
```

Interface Type	10^8/bps = Cost
Fast Ethernet and faster	10^8/100,000,000 bps = 1
Ethernet	10^8/10,000,000 bps = 10
E1	10^8/2,048,000 bps = 48
T1	10^8/1,544,000 bps = 64
128 kbps	10^8/128,000 bps = 781
64 kbps	10^8/64,000 bps = 1562
56 kbps	10^8/56,000 bps = 1785


```
!--------------------------------------------------
! Summarize OSPF areas at the ABR
!--------------------------------------------------
router ospf <>
 area <> range <net> <sm>

!--------------------------------------------------
! Summarize external routes into OSPF at the ASBR
!--------------------------------------------------
router ospf <pid>
 summary-address <net> <sm>

sh ip route ospf | include E2
sh ip ospf summary-address
```

```
!--------------------------------------
! How to reset Router-ID
!--------------------------------------
int lo0
 ip 1.1.1.1 255.255.255.255
reload
! or
router ospf 1
 router-id 1.1.1.1
clear ip ospf process
```

The **show ip ospf** command should be used to verify the OSPF router ID. If the OSPF router ID is using a 32-bit value other than the one specified by the **router-id** command, you can reset the router ID by using the **clear ip ospf** *pid* **process** command and re-verify using the command **show ip ospf**.

If an area is not connected to the backbone, its routes are not advertised outside of its area

```
!--------------------------------------------------
! Virtual link
!--------------------------------------------------
sh ip protocols
sh ip ospf interface

router ospf <>
 area <transit-area-no> virtual-link <remote-router-id>

sh ip ospf virtual-links
```

```
!--------------------------------------------
! Activate interface for routing - new method
!--------------------------------------------
int <>
 ip ospf <> area <>
```

Note: Another option is to use the OSPF **network** command in router configuration mode.

```
!--------------------------------------
! Make loopback int act as p2p in ospf
!--------------------------------------
```

Note: The default behavior of OSPF for loopback interfaces is to advertise a 32-bit host route. To ensure that the full /24 network is advertised, use the **ip ospf network point-to-point** command. Change the network type on the loopback interfaces so that they are advertised with the correct subnet.

```
int lo <>
 ip ospf network point-to-point
```

```
!-----------------------------
! OSPF Hello message
! * has to match!!
!-----------------------------
    -Router ID
    -Hello and Dead Timers*
    -Network mask*
    -Area ID*
    -Neighbors
    -Router priority
    -DR/BDR IP address
    -Authentication password*
    -Stub flag*
```

```
type 1 - Router Link States (router ID)
type 2 - Net Link States (networks in area - intra area)
type 3 - Summary Net Link States (Inter Area)
type 4 - Summary ASB Link States (way to ASBR)
type 5 - Type-5 AS External Link States (external nets)
type 7 - Type-7 AS External Link States (external nets in NSSA)
```

```
!--------------------------------------
! Interface OSPF network types
! - if point-to-  then no DR/BDR election
! - if broadcast  then no neighbor statement
!--------------------------------------
int <>
 ip ospf network point-to-point
 ip ospf network point-to-multipoint [non-
broadcast]
 ip ospf network broadcast
 ip ospf network non-broadcast
```

```
!--------------------------------------
! OSPF DR election on MultiAccess network
!
!--------------------------------------
!--------------------------------------
! link priority - default is 1
! 0 -> do not participate in DR election
! The one with highest value becomes DR
! - remember to clear ip ospf process
!--------------------------------------
int fa 0/0
 ip ospf priority <value>

If all have same link priority then:

DR = Highest Router-ID
priority:
  1) manuel router-ID
     router-id <>
  or
  2) automatic router-ID
     2a) Highest IP on loopback int
     2b) Highest IP on physical int
```

Type	Packet Name	Description
1	Hello	Discovers neighbors and builds adjacencies between them
2	Database Description (DBD)	Checks for database synchronization between routers
3	Link-State Request (LSR)	Requests specific link-state records from router to router
4	Link-State Update (LSU)	Sends specifically requested link-state records
5	Link-State Acknowledgment (LSAck)	Acknowledges the other packet types

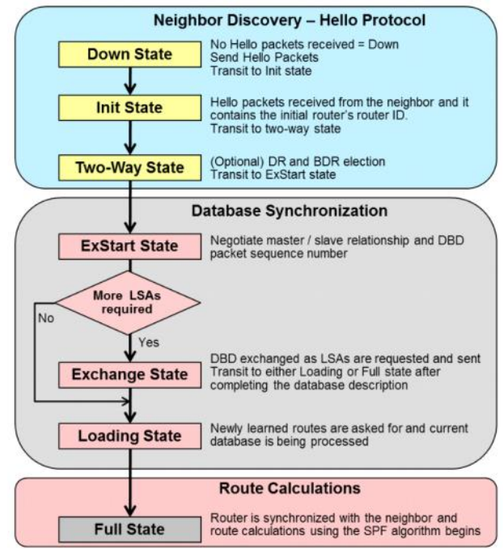
	LSA type	LSA blocking
STUB	1,2,3	4,5
NSSA	1,2,3+7	4,5

(ABR) area X nssa default-information-originate

0.0.0.0/0

		0.0.0.0/0 (OIA)
*Totally STUB	1,2	3,4,5
*Totally NSSA	1,2+7	3,4,5

*cisco propritary




```
!--------------------------------------
! OSPFv2 stub area
!--------------------------------------
router ospf <>
 area <> stub

sh ip route ospf
sh ip ospf

!--------------------------------------
! OSPFv2 totally stubby area
! no-summary only on ABR
!--------------------------------------
router ospf <>
 area <> stub [no-summary]
```

OSPFv3 (Traditional)

!Considered Legacy - use address-family!

Thursday, 22 December 2022 13.17

```
!-------------------------------------
! OSPFv3 protocol
! adm distance: 110
! Multicast:
!  - FF02::5 (all OSPF routers)
!  - FF02::6 (all OSPF DRs)
! Algorithm: Dijkstra (SPF)
! Metric: cost (10^8/Bandwidth)
!-------------------------------------
!
!-----------------------------------------------
! enable IPv6 routing
!-----------------------------------------------
ipv6 unicast-routing

sh run | section ipv6 unicast

!-----------------------------------------------
! enable OSPF routing on interface
!-----------------------------------------------
int <>
 ipv6 ospf 1 area <>

!-----------------------------------------------
! Router-ID and other specific OSPFv3 cmd's
!-----------------------------------------------
ipv6 router ospf <process-id>
 router-id <n.n.n.n>
 passive-interface default
 no passive-interface <>
 default-information originate
 redistribute static ! used in lab on ASBR…
```

Without any IPv4 addresses on the router, the OSPFv3 process will not start unless you manually set the router IDs.

```
! automatically generates a link-
! local address
ipv6 enable

!-------------------------------------
! IPv6 OSPF commands
!-------------------------------------
sh ipv6 protocols
sh ipv6 route …
sh ipv6 route ospf
sh ipv6 route static
sh ipv6 ospf
sh ipv6 ospf interface
sh ipv6 ospf database
sh ipv6 ospf neighbor
sh ipv6 interface
sh ipv6 int brief

clear ipv6 ospf <PID> process




!-----------------------------------------------
! CEF
!-----------------------------------------------
sh ipv6 cef - to enable ipv6 cef -> ipv6 cef
```

```
!-----------------------------------------------
! OSPFv3 stub area
!-----------------------------------------------
ipv6 router ospf <>
 area <> stub

sh ip route ospf
sh ip ospf

!-----------------------------------------------
! OSPFv3 totally stubby area
! no-summary only on ABR
!-----------------------------------------------
ipv6 router ospf <>
 area <> stub [no-summary]
```

```
!-----------------------------------------------
!
!-----------------------------------------------
```

```
Router Link States
Inter Area Prefix Link States
Link LSA (Type-8) Link States
Intra Area Prefix Link States
Type-5 AS External Link States
```

new types

- **Link LSA** (type 8)

Informs neighbors of link local address
Informs neighbors of IPv6 prefixes on link

- **Intra-Area Prefix LSA** (type 9)

Associates IPv6 prefixes with a network or router

	LSA Function Code	LSA type
Router-LSA	1	0x2001
Network-LSA	2	0x2002
Inter-Area-Prefix-LSA	3	0x2003
Inter-Area-Router-LSA	4	0x2004
AS-External-LSA	5	0x4005
Group-membership-LSA	6	0x2006
Type-7-LSA	7	0x2007
Link-LSA	8	0x0008
Intra-Area-Prefix-LSA	9	0x2009

OSPFv3 Address-Family

31. august 2023 10:04

```
!-----------------------------------------------
! OSPF Address families for IPv4 and IPv6
!-----------------------------------------------
ipv6 unicast-routing

router ospfv3 <pid>
 address-family ipv4 unicast
  router-id <>
  passive-interface <>
  default-information originate
  redistribute static
  exit-address-family
 address-family ipv6 unicast
  router-id <>
  passive-interface <>
  default-information originate
  redistribute static
  exit-address-family
 exit

! Enable routing on interface
int <>
 ospfv3 <pid> ipv4 area <>
 ospfv3 <pid> ipv6 area <>

sh ip ospf neighbor
sh ipv6 ospf neighbor

!show both IPv4 and IPv6 neighbors
sh ospfv3 neighbor

sh ip route
sh ipv6 route

sh ip route ospfv3
sh ip route static
sh ipv6 route static


!-----------------------------------------------
! OSPFv3 [totally] stub[by] area
!-----------------------------------------------
router ospfv3 <pid>
 address-family ipv4 unicast
  area <> stub [no-summary]
  exit-address-family
 address-family ipv6 unicast
  area <> stub [no-summary]
  exit-address-family
 exit

sh ospfv3 neighbor

look for passive interfaces
sh ospfv3 interface <>
 -- look for 'No Hellos (Passive interface)'
```

OSPFv3 with the addresses family (AF) unifies OSPF configuration for both IPv4 and IPv6. OSPFv3 with address families also combines neighbor tables and the LSDB under a single OSPF process. OSPFv3 messages are sent over IPv6 and therefore requires that IPv6 routing is enabled and that the interface has a link-local IPv6 address. This is the requirement even if only the IPv4 AF is configured.

Note: After configuring the OSPFv3 address families, the **show ospfv3** command should be used to verify the OSPF router ID for both the IPv4 and IPv6 AF. If the OSPF router ID is using a 32-bit value other than the one specified by the **router-id** command, you can reset the router ID by using the **clear ospfv3** *pid* **process** command and re-verify using the command **show ospfv3**.

Understanding the difference between commands associated with OSPFv2 and OSPFv3 can seem challenging at times. The **show ip route ospfv3** command is used to view OSPFv3 routes in the IPv4 routing table. The **show ipv6 route ospf** command is used to view OSPFv3 routes in the IPv6 routing table. The **show ipv6 route ospf** command is the same command used in with traditional OSPFv3 for IPv6.

```
!-----------------------------------
! IPv6 OSPF commands
!-----------------------------------
sh ipv6 protocols
sh ipv6 route …
```

sh ipv6 route ospf	sh ip route ospfv3

```
sh ipv6 route static
sh ipv6 ospf
sh ipv6 ospf interface
```

sh ipv6 ospf database	sh ospfv3 database
sh ipv6 ospf neighbor	sh ospfv3 neighbor

```
sh ipv6 interface
sh ipv6 int brief
```

BGP

Friday, 23 December 2022 13.46

```
!-------------------------------------------------
! BGP protocol
! adm. distance: External = 20, Internal = 200
! TCP port 179
! Algorithm: Path Vector
!-------------------------------------------------
router bgp <as>
 bgp router-id <>
 neighbor <> remote-as <>
 neighbor <IP> update-source <loopback> ! typically iBGP
 neighbor <IP> next-hop-self
 neighbor <IP> route-map <NAME> {in|out}
 neighbor <IP> default-originate
 neighbor <IP> remove-private-as   !64512-65535
 neighbor <IP> password <PW>       ! md5 authentication

 ! Summary route
 aggregate-address <NET+SM> [summary-only]

 ! network you want bgp to tell about
 network <NET> mask <SM>

sh ip bgp summary
sh ip bgp neighbors

sh ip bgp
```

An asterisk (*) next to a route indicates that the table entry is valid
An angle bracket (>) indicates the best path for a route selected by BGP. This route is offered to the IP routing table

```
sh ip route
sh ip bgp rib-failure
```

Note: The **clear ip bgp *** command is disruptive because it completely resets all BGP adjacencies. This is acceptable in a lab environment but could be problematic in a production network. Instead, if only a change of inbound/outbound routing policies is to be performed, it is sufficient to issue the **clear ip bgp * in** or **clear ip bgp * out** commands**.** These commands perform only a new BGP database synchronization without the disruptive effects of a complete BGP adjacency reset. All current Cisco IOS versions support the route refresh capability that replaces the inbound soft reconfiguration feature that previously had to be configured on a per-neighbor basis.

```
clear ip bgp *           ! resets all BGP adjacencies
clear ip bgp * {in|out}  ! perform only a new BGP database synchronization
clear ip bgp * soft      ! force each router to resend its BGP table (both in and out)

!-------------------------------------------------
! Using BGP to propagate a default route
!-------------------------------------------------
router bgp <as>
 neighbor <> default-originate

!-------------------------------------------------
! Configure a discard static route
!-------------------------------------------------
ip route <net> <sm> null0

!-------------------------------------------------
! synchronisation
!-------------------------------------------------
router bgp <as>
 no synchronisation !default

!-------------------------------------------------
! Next hop self
!-------------------------------------------------
```

It won't install a route if it doesn't know how to get to the next hop

```
router bgp <as>
 neighbor <> next-hop-self

!-------------------------------------------------
! Set local-preference
! Stamp networks/routes recieved from ISP
! - it will affect data traffic towards ISP
! default = 100 (highest value win)
!-------------------------------------------------
route-map <NAME> permit 10
 set local-preference <>
 exit
!
router bgp <as>
 neighbor <> route-map <NAME> in

sh ip bgp

!-------------------------------------------------
! Set Multi Exit Discriminator (MED)
! Stamp networks/routes sent to ISP
! - it will affect data traffic from ISP
! default = 0 (lowest value win)
!-------------------------------------------------
route-map <NAME> permit 10
 set metric <>
 exit
!
router bgp <as>
 neighbor <> route-map <NAME> out

sh ip bgp

!-------------------------------------------------
! Set AS-Path prepend
! send to ISP
!-------------------------------------------------
route-map <NAME> permit 10
 set as-path prepend <as-number> [as-number as-number…]
```

BGP best-path:

1. Prefer the highest weight
2. Prefer the highest local preference
3. Prefer the route originated by the local router
4. Prefer the path with the shorter Accumulated Interior Gateway Protocol (AIGP) metric attribute
5. Prefer the shortest AS_Path
6. Prefer the best origin code (IGP, EGP, Incomplete)
7. Prefer the lowest multi-exit discriminator (MED)
8. Prefer an external path over an internal path
9. Prefer the path through the closest IGP neighbor
10. Prefer the oldest route for eBGP paths
11. Prefer the path with the lowest neighbor BGP RID
12. Prefer the path with the minimum cluster list length
13. Prefer the path with the lowest neighbor IP address

```
!-------------------------------------------------
! filter updates
!-------------------------------------------------
access-list 1 permit <net> <wm>
!
router bgp <as>
 neighbor <> distribute-list 1 out

!-------------------------------------------------
! filter specific routes using prefix-list
!-------------------------------------------------
ip prefix-list <NAME> permit <PREFIX>

router bgp <AS>
 neighbor <IP> prefix-list <NAME> {out|in}

!-------------------------------------------------
! Use the AS_PATH attribute to filter routes
! using filter-list
!-------------------------------------------------
```

AS-path access lists are read like regular access lists. The statements are read sequentially, and there is an implicit deny at the end. Rather than matching an address in each statement like a conventional access list, AS path access lists match on something called a regular expression. Regular expressions are a way of matching text patterns and have many uses. In this case, you will be using them in the AS path access list to match text patterns in AS paths

The first command uses the **^** character to indicate that the AS path must begin with the given number 100. The **$** character indicates that the AS_PATH attribute must also end with 100. Essentially, this statement matches only paths that are sourced from AS 100. Other paths, which might include AS 100 along the way, will not match this list.

In the second statement, the **.** (period) is a wildcard, and the ***** (asterisk) stands for a repetition of the wildcard. Together, **.*** matches any value of the AS_PATH attribute, which in effect permits any update that has not been denied by the previous **access-list** statement.

```
ip as-path access-list 1 deny ^100$
ip as-path access-list 1 .*
!
router bgp <as>
 neighbor <> filter-list 1 out

! show all regular expressions that matches as-path acl
sh ip bgp regex ^100         ! begins with 100
sh ip bgp regex ^100$        ! begins and ends with 100

^ = begin with ..
$ = end with ..
. = wildcard
* = repetition
_ = Matches a space

!-------------------------------------------------
! Configure AS <> non-transit AS
!-------------------------------------------------
```

To avoid being a transit AS, on SanJose1, configure an AS-path filter using an AS-path access list. The access list will only permit locally sourced routes are sent to the provider, ISP1. Routes learned from another AS will be filtered and not included in its updates. This filter is applied to a set of routes announced to the ISP1 neighbor.

The regular expression "**^$**" matches only routes that are locally sourced, do not contain an AS in its AS-path

```
router bgp <as>
 neighbor <> filter-list 1 out
 exit
ip as-path access-list 1 permit ^$
```

BGP - MP (Address Family)

16. marts 2023 11:19

```
!-----------------------------------------------
! Configure eBGP, iBGP
!-----------------------------------------------

router bgp <as>
 bgp router-id <>
 [no bgp default ipv4-unicast]
 bgp log-neighbor-changes

 neighbor <iBGP IPv6> remote-as <as>
 neighbor <iBGP IPv6> update-source Loopback0

 neighbor <iBGP IPv4> remote-as <as>
 neighbor <iBGP IPv4> update-source Loopback0

 neighbor <eBGP IPv4> remote-as <as>
 neighbor <eBGP IPv6> remote-as <as>
 !
 address-family ipv4
  network <> mask <sm>

  neighbor <BGP IPv4> activate

  neighbor <iBGP IPv4> next-hop-self

 exit-address-family
 !
 address-family ipv6
  network <net/x>

  neighbor <BGP IPv6> activate
neighbor <iBGP IPv6> next-hop-self

exit-address-family
```

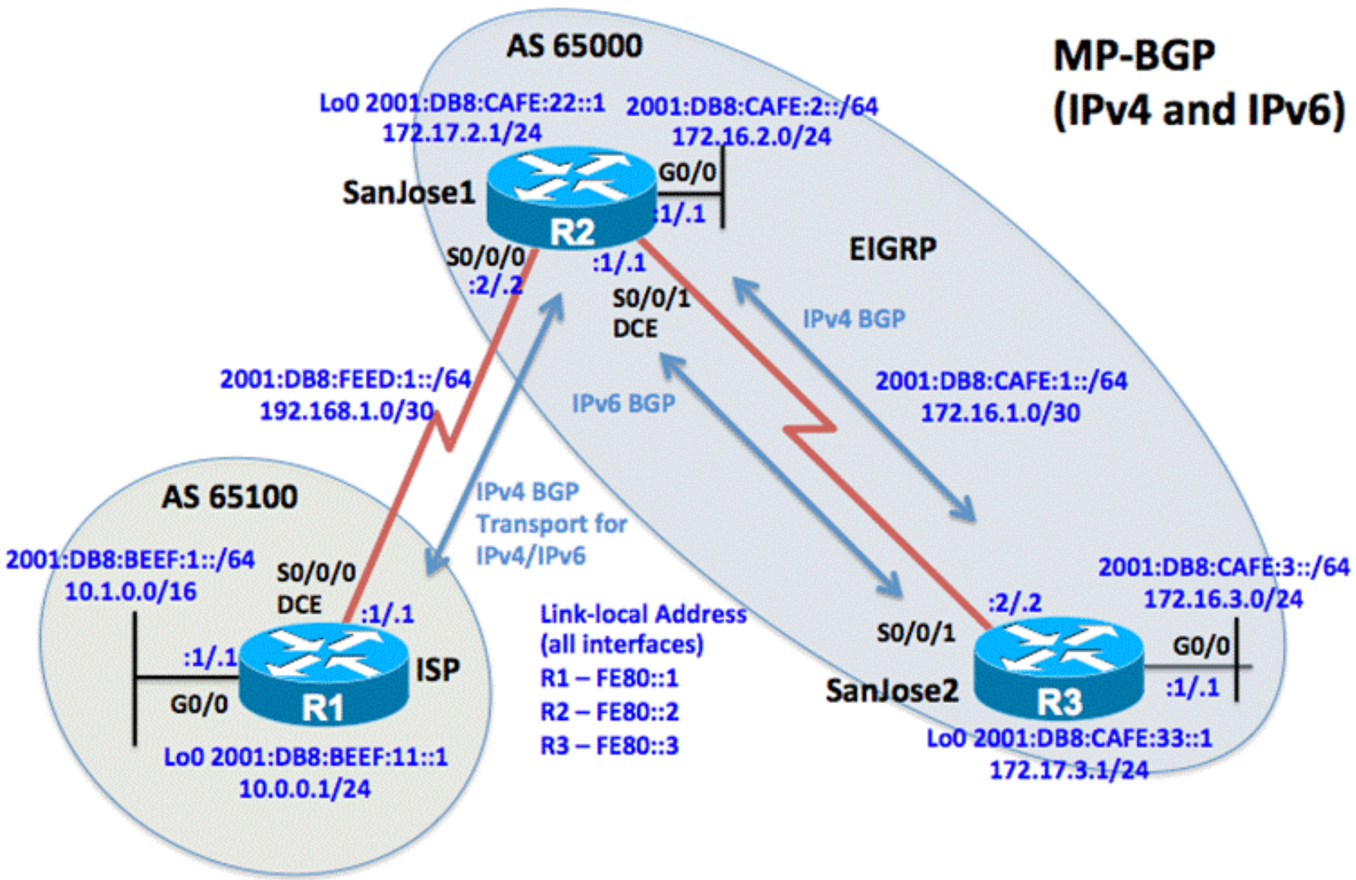
```
!-----------------------------------------------
!
!-----------------------------------------------
```

```
!-----------------------------------------------
! Configure eBGP, iBGP and eigrp using IPv4
! as transport protocol
!-----------------------------------------------
router eigrp <as>
 eigrp router-id <>
 network <net> <wm>
!
router bgp <as>
 bgp router-id <>
 [no bgp default ipv4-unicast]
 bgp log-neighbor-changes
 neighbor <iBGP IPv6> remote-as <as>
 neighbor <iBGP IPv6> update-source Loopback0
 neighbor <iBGP IPv4> remote-as <as>
 neighbor <iBGP IPv4> update-source Loopback0
 neighbor <eBGP IPv4> remote-as <as>
 !
 address-family ipv4
  network <> mask <sm>
  no neighbor <iBGP IPv6> activate
  neighbor <iBGP IPv4> activate
  neighbor <iBGP IPv4> next-hop-self
  neighbor <eBGP IPv4> activate
 exit-address-family
 !
 address-family ipv6
  network <net/x>
  neighbor <iBGP IPv6> activate
  neighbor <iBGP IPv6> next-hop-self
  neighbor <eBGP IPv4> activate
  neighbor <eBGP IPv4> route-map NEXT-HOP-IPV6 out
 exit-address-family
 !
 ipv6 router eigrp <as>
  eigrp router-id <>
 !
 route-map NEXT-HOP-IPV6 permit 10
  set ipv6 next-hop <iBGP IPv6>
 !
 !
 sh bgp all neighbors !show both IPv4 and IPv6
 sh ip bgp neighbors
 sh bgp ipv6 unicast neighbors

 sh bgp ipv4 unicast summary
 sh bgp ipv6 unicast summary

 sh bgp ipv4 unicast !same as "sh ip bgp"
 sh bgp ipv6 unicast

 sh ip route
 sh ipv6 route
```

BGP - MP example

24. maj 2016 12:00

```
!-----------------------------------------------
! Configure eBGP on ISP (R1)
!-----------------------------------------------
router bgp 65100
 bgp router-id 1.1.1.1
 bgp log-neighbor-changes
 neighbor 192.168.1.2 remote-as 65000
 !
 address-family ipv4
  network 10.1.0.0 mask 255.255.0.0
  neighbor 192.168.1.2 activate
 exit-address-family
 !
 address-family ipv6
  network 2001:DB8:BEEF:1::/64
  neighbor 192.168.1.2 activate
  neighbor 192.168.1.2 route-map NEXT-HOP-IPV6 out
 exit-address-family
!
route-map NEXT-HOP-IPV6 permit 10
 set ipv6 next-hop 2001:DB8:FEED:1::1
```




```
!-----------------------------------------------
! Configure eBGP, iBGP and eigrp (R2)
!-----------------------------------------------
router eigrp 1
 network 172.16.0.0
 network 172.17.0.0
 eigrp router-id 2.2.2.2
!
router bgp 65000
 bgp router-id 2.2.2.2
 bgp log-neighbor-changes
 neighbor 2001:DB8:CAFE:33::1 remote-as 65000
 neighbor 2001:DB8:CAFE:33::1 update-source Loopback0
 neighbor 172.17.3.1 remote-as 65000
 neighbor 172.17.3.1 update-source Loopback0
 neighbor 192.168.1.1 remote-as 65100
 !
 address-family ipv4
  network 172.16.2.0 mask 255.255.255.0
  no neighbor 2001:DB8:CAFE:33::1 activate
  neighbor 172.17.3.1 activate
  neighbor 172.17.3.1 next-hop-self
  neighbor 192.168.1.1 activate
 exit-address-family
 !
 address-family ipv6
  network 2001:DB8:CAFE:2::/64
  neighbor 2001:DB8:CAFE:33::1 activate
  neighbor 2001:DB8:CAFE:33::1 next-hop-self
  neighbor 192.168.1.1 activate
  neighbor 192.168.1.1 route-map NEXT-HOP-IPV6 out
 exit-address-family
!
ipv6 router eigrp 2
 eigrp router-id 2.2.2.2
!
route-map NEXT-HOP-IPV6 permit 10
 set ipv6 next-hop 2001:DB8:FEED:1::2
```

```
!-----------------------------------------------
! Configure iBGP and eigrp (R3)
!-----------------------------------------------

router eigrp 1
 network 172.16.0.0
 network 172.17.0.0
 eigrp router-id 3.3.3.3
!
router bgp 65000
 bgp router-id 3.3.3.3
 bgp log-neighbor-changes
 neighbor 2001:DB8:CAFE:22::1 remote-as 65000
 neighbor 2001:DB8:CAFE:22::1 update-source Loopback0
 neighbor 172.17.2.1 remote-as 65000
 neighbor 172.17.2.1 update-source Loopback0
 !
 address-family ipv4
  network 172.16.3.0 mask 255.255.255.0
  no neighbor 2001:DB8:CAFE:22::1 activate
  neighbor 172.17.2.1 activate
 exit-address-family
 !
 address-family ipv6
  network 2001:DB8:CAFE:3::/64
  neighbor 2001:DB8:CAFE:22::1 activate
 exit-address-family
!
ipv6 router eigrp 2
 eigrp router-id 3.3.3.3
```

Redistribution

11. november 2020 14:25

```
!-------------------------------------------------------------------
! redistribution -> RIP default-metric is 0 = infinity
!-------------------------------------------------------------------
router rip
 redistribute protocol[process-id] [metric <hop>]

!-------------------------------------------------------------------
! redistribution -> EIGRP default-metric is infinity
!-------------------------------------------------------------------
router eigrp <AS>
 redistribute protocol[process-id] [metric <BW DLY RELY LOAD MTU>]
! or
! default-metric 10000 100 255 1 1500
! redistribute protocol[process-id]

!-------------------------------------------------------------------
! redistribution -> OSPF
! E1 = external cost + internal cost (use if more paths to dest-net)
! E2 = (default) external cost only
! default-metric is 20
!-------------------------------------------------------------------
router ospf 1
 redistribute protocol[id] [metric <value>] [metric-type {E1|E2}] [subnets]
```

The **subnets** command is necessary because, by default, OSPF only redistributes classful networks and supernets

```
!-------------------------------------------------------------------
! metric seed
! normaly used when redistributing into Distance Vector protocols
!-------------------------------------------------------------------
router protocol [process-id]
 default-metric <value>
```

Administrative Distance

Route Source	Default Distance
Connected Interface	0
Static Route	1
Enhanced IGRP Summary Route	5
External BGP	20
Internal Enhanced IGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
External Enhanced IGRP	170
Internal BGP	200
Unknown	255

```
!-------------------------------------------------
! Configure mutual redistribution between
! OSPFv3 and EIGRP for IPv6
!-------------------------------------------------
router ospfv3 <pid>
 address-family ipv6 unicast
  redistribute eigrp <as> include-connected
  exit
 exit
```

A default seed metric is not required for OSPFv3. Redistributed routes are assigned a metric of 20 by default.

```
ipv6 router eigrp <as>
 redistribute ospf <pid> metric 10000 100 255 1 1500 include-connected

sh ipv6 protocols
```

```
!-------------------------------------------------
! change adm. distance
!-------------------------------------------------
router eigrp <AS>
 distance eigrp <internal> <external>

router ospf 1
 distance ospf intra-area <no> inter-area <no> external <no>

!-------------------------------------------------
! change adm. distance in general
!-------------------------------------------------
router <protocol>
 distance <distance>

!-------------------------------------------------
! Modify distance, based on route source.
!-------------------------------------------------
```

You can also modify administrative distance based on route source using the distance distance address wildcard command, where address and wildcard represent the peer advertising the route. For OSPF, the address is the router ID.

```
router <protocol>
 distance <distance> <NET+WM>

!-------------------------------------------------
! Modify distance, based on an access list.
!-------------------------------------------------
```

You can also modify administrative distance based on which routes match an access list using the distance distance address wildcard acl command. The way you list routes in an access list which will be used to modify distance is similar to how you list them when the access list is used to filter routes. For this lab, create an access list containing all the subnets of 172.16.0.0/16. Then associate the access list with the distance command, setting the address and wildcard to be any IP address (i.e., any route source).

```
router <protocol>
 distance <distance> <NET+WM> <ACL>

i.e.
access-list permit <NET+WM>
router <protocol>
 distance 65 0.0.0.0 255.255.255.255 1

sh ip protocols
```

Filter Routes

24. maj 2016 11:14

```
!-----------------------------------------------
! with distribute-list
!-----------------------------------------------
```

- **Distribute list and ACL**— A distribute list allows an access control lists (ACLs) to be applied to routing updates.
- **Distribute list and prefix list**— A distribute list with a prefix list is an alternative to ACLs designed to filter routes. Prefix lists are not exclusively used with distribute lists but can also be used with route maps and other commands.
- **Route maps**— Route maps are complex access lists that allow conditions to be tested against a packet or route, and then actions taken to modify attributes of the packet or route.

Although a distribute list could be implemented on the receiving router, it is usually best to filter routes from the distributing router

In large enterprise networks, route filtering can be quite complex. The ACLs can be very extensive and therefore taxing on router resources. For this reason, prefix lists should be used instead of ACLs since they are more efficient and less taxing on router resources than ACLs.

```
!-----------------------------------------------
! Distribute list and ACL
!-----------------------------------------------
ip access-list standard <NAME>
 remark Used with Dlist to filter OSPF 20 routes
 deny <net> <wm>
 permit any
 exit
!
router eigrp <as>
 distribute-list <ACL-NAME> out <>

!-----------------------------------------------
! Distribute list and prefix-list
!-----------------------------------------------
ip prefix-list <NAME> description <>
ip prefix-list <NAME> permit/deny <prefix/x> …
ip prefix-list <NAME> permit 0.0.0.0/0 le 32


router ospf <pid>
 distribute-list prefix <NAME> out <>
```

```
!-----------------------------------------------
! Route map
!-----------------------------------------------
```

Route maps can also be used to filter redistributed routes. A route map works like an access list because it has multiple deny and permit statements that are read in a sequential order. However, route maps can match and set specific attributes and therefore provide additional options and more flexibility when redistributing routes.

Route maps are not just for redistribution. They are also commonly used for:

- **Policy-based routing (PBR)**— PBR allows an administrator to define routing policy other than basic destination-based routing using the routing table. The route map is applied to an interface using the **ip policy route-map** interface configuration command.
- **BGP**—Route maps are the primary tools for implementing BGP policy and allows an administrator to do path control and provide sophisticated manipulation of BGP path attributes. The route map is applied using the BGP **neighbor** router configuration command.

```
ip access-list standard <NAME>
 remark <>
 permit <net> <wm>
 permit <net> <wm>
 exit
!
route-map <NAME> deny 10
 description RM filters ... routes
 match ip address <ACL>
 exit
route-map <NAME> permit 20
 description RM permits all other ... routes
 exit
!
router <protocol>
 redistribute <protocol> metric <value> route-map <NAME>


sh ip route eigrp | begin Gateway

!-----------------------------------------------
! Filter redistributed routes and set attributes
! using a route map
!-----------------------------------------------
ip prefix-list <NAME> description <>
ip prifix-list <NAME> permit <prefix/x>
!
route-map <NAME> permit 10
 description RM filters ...
 match ip address prefix-list <NAME>
 set metric 25
 set metric-type type-1
 exit
route-map <NAME> permit 20
 description RM permits all other ... routes
 exit
!
router <protocol>
 redistribute <protocol> metric <value> route-map <NAME>
```

IP SLA/Policy Based Routing (PBR)

11. maj 2016 06:54

```
!-----------------------------------------------
! Configure IP SLA Probes
!-----------------------------------------------
ip sla <no>
 icmp-echo <dest-IP> [<source-IP>]
  frequency <sec>
  exit
!
ip sla schedule <ip-sla-no> life forever start-time now

sh ip sla configuration <>
sh ip sla statistics

!-----------------------------------------------
! Configure tracking options
!-----------------------------------------------
track <track-no> ip sla <ip-sla-no> reachability
 delay down <sec> up <sec>
 exit

!-----------------------------------------------
! Activate tracking
! route goes down if track reachability goes down
! - then route with higher adm-distance takes over
!-----------------------------------------------
ip route 0.0.0.0 0.0.0.0 <next-hop-ip> <adm-distance> track <no>
ip route 0.0.0.0 0.0.0.0 <next-hop-ip> <adm-distance>

debug ip routing
sh ip route | begin Gateway
```

```
!-----------------------------------------------
! Configure Policy Based Routing (PBR)
!-----------------------------------------------
ip access-list standard <NAME>
 remark ACL matches … traffic
 permit <net> <wm>
 exit
!
route-map <NAME> permit
 description RM to …
 match ip address <ACL>
 set ip next-hop <>
 exit

??? what about other traffic??? - Deny any -
--- other traffic is not PBR -> normal forwarding

int <>
 ip policy route-map <NAME>

sh route-map

access-list 1 permit <net> <sm>
debug ip policy <ACLno>
```

Configuring PBR involves configuring a route map with **match** and **set** commands and then applying the route map to the interface.

The steps required to implement path control include the following:

- Choose the path control tool to use. Path control tools manipulate or bypass the IP routing table. For PBR, **route-map** commands are used.
- Implement the traffic-matching configuration, specifying which traffic will be manipulated. The **match** commands are used within route maps.
- Define the action for the matched traffic using **set** commands within route maps.
- Apply the route map to incoming traffic.

Secure Management Plane and Routing

3. maj 2016 13:03

```
!-----------------------------------------------
! Secure Management access
!-----------------------------------------------
security passwords min-length 10
enable secret class12345
service password-encryption
banner motd $Unauthorized access strictly prohibited!$

line con 0
 password <>
 exec-timeout <min> <sec>
 login
 logging synchronous
 exit
!
line vty 0 4
 password <>
 exec-timeout <> <>
 login
 exit
!
line aux 0
 no exec
```

Can you read the enable secret password?
No. The enable secret password is encrypted automatically using the MD5 or SHA hash algorithm. . IOS 15.0(1)S and later default to SHA256 hashing algorithm. SHA256 which is considered to be a very strong hashing algorithm and is extremely difficult to reverse. Earlier IOS versions use the weaker MD5 hashing algorithm.

```
!-----------------------------------------------
! Configure enhanced username password security
!-----------------------------------------------
username <> secret <pw> !create local database entry encrypted to level 4 (SHA256)

line <>
 login local

!-----------------------------------------------
! Enable AAA Authentication
!-----------------------------------------------
aaa new-model
radius server <name-1>
 address ipv4 <ip>
 key <pw>
 exit
radius server <name-2>
 address ipv4 <ip>
 key <pw>
 exit
!
aaa group server radius <group-name>
 server name <name-1>
 server name <name-2>
 exit
!
aaa authentication login default group <radius-group-name> local
aaa authentication login <TELNET-LOGIN> <radius-group-name> local-case

line vty 0 4
 login authentication <TELNET-LOGIN>
```

```
!-------------------------------------
! security SSH
!-------------------------------------
hostname <>
ip domain-name <>
username <user> privilege 15 secret <pw>
!
line vty 0 15
 transport input ssh
 login local
 exit
!
crypto key generate rsa general-keys modulus 1024

ip ssh version 2
ip ssh time-out <sec>
ip ssh authentication-retries <>

! display version and configuration
sh ip ssh

! check the SSH connections to the device
sh ssh

ssh -l <username> [-p <password>] <IP>
ssh -v <IP>
!-v = verbose (give more information)
ssh ?

! delete RSA key pair
crypto key zeroize rsa

!-------------------------------------
! security HTTP server (turn off if not in use)
!-------------------------------------
sh ip http server status
no ip http server

!-------------------------------------
! security Shutdown unused ports
!-------------------------------------
int range fa 0/x-y
 shutdown

!-------------------------------------
! Basic security:
!-------------------------------------
```

Blocks login attempts for 30 seconds if someone fails two attempts within 120 seconds

```
login block-for 30 attempts 2 within 120
sh login
```

```
!-----------------------------------------------
!Secure EIGRP routing protocol using md5 authentication
!-----------------------------------------------
key chain <KEY-NAME>
 key <no>
  key-string <PASSWORD>

interface <int>
 ip authentication mode eigrp <AS> md5
 ip authentication key-chain eigrp <AS> <KEY-NAME>

!-----------------------------------------------
! Secure EIGRP routing protocol using SHA authentication
!-----------------------------------------------
key chain <name>
 key <no>
  key-string <pw-2>
  exit
 exit
!
router eigrp <name>
 adress-family ipv4 autonomous-system <as>
  af-interface <>
   authentication key-chain <name>
   authentication mode hmac-sha-256 <pw-1>
```

```
!-----------------------------------------------
! Secure OSPF routing protocol using md5 authentication
! - On interface level
!-----------------------------------------------
interface <int>
 ip ospf message-digest-key 1 md5 <PASSWORD>
 ip ospf authentication message-digest

!-----------------------------------------------
! Secure OSPF routing protocol using md5 authentication
! - On area level
!-----------------------------------------------
router ospf 1
 area 0 authentication message-digest
!
int <>
 ip ospf message-digest-key 1 md5 <PASSWORD>

debug ip ospf adj
sh ip ospf int <>

!-----------------------------------------------
! Secure OSPF routing protocol using SHA authentication
!-----------------------------------------------
key chain <name>
 key <no>
  key-string <pw-1>
  cryptographic-algorithm hmac-sha-256
  exit
!
int <>
 ip ospf authentication key-chain <name>
```