Notes: The Series (NTS)

Here you will find some of my blog posts (the so-called "**Notes: The Series**", aka **NTS**) that include my personal notes while preparing for the **CCIE SPv3 lab**. I have put them here so it's easier for the reader and the community to find all the information in one place.

These should not be treated as a teaching guide, but just a simple guide with interesting information that might be useful to others besides me. In the majority of the posts, the included information refers to the lab exam software versions, but there are quite a few of notes and configuration examples that refer to newer versions. Hopefully the information is correct at its majority, but **if you encounter anything that seems wrong, please don't hesitate to write a comment** under the relevant NTS page.

I have used various colors, either on the text or on the background, to aid focusing where needed. IOS and IOS-XR have a distinct background color, so it's easier to find the one you're looking for (i tried to put a lot of information about IOS-XR too, since it plays an important role in the exam). Things that i consider **tricky or important** are colored in yellow background. Cli commands and important cli outputs are colored in **blue**, while **green** and **red** are used to differentiate between correct and wrong/error/warning wherever possible.

You can always find the latest version of this file under NTS.

History

Version	Comments
V1.0	Initial version
V1.1	Intro & About me added

About me and the CCIE

While my fist CCIE was completed in 3 months, for this one i had put **2 months** as a target due to various major projects running in the following months at my work. You can read the complete story of my success here.

What is my advice to everyone thinking of trying the current CCIE SP lab? Learn the core technologies (IGPs, MP-BGP, MPLS/TE, Multicast, IPv6) inside-out, combine and test them in all possible scenarios (one above the other, one combined with the other, multiple combinations above or below other technologies, etc.) and then focus on configuration speed. ***Think fast, act faster.*** *When you become a master on that, spend some time on every other topic too. And* ***don't risk any challenge****s, unless you're crazy like me!*

My proposed hints for passing the CCIE SP lab:

- Read the RFCs, participate in IETF WGs
- Use GNS3 for your IOS (and soon IOS-XR) needs
- Use INE's rack rentals or PEC Gold Labs or your work's lab for your IOS-XR needs
- Use INE's full scale labs for testing and improving your readiness
- Keep track of your progress, rate yourself as much as possible but always be honest
- Enable IPv6 on something every day
- Live with MPLS, sleep with MPLS, eat with MPLS, dream of MPLS
- Master the trio of success
- Don't be afraid of changing early your lab tactic if it doesn't work for you
- Take typing lessons

Lab study days	**59**
Lab study hours	**354,5** (from which ~100 hours were spent on this blog)
Average lab study hours per day	**6** (from which ~1,5 hours on avg were spent daily on this blog)
Result	**PASS**

Personally i believe that for someone with a ***good experience in networking****, with* ***deep understanding of the technologies*** *and with the* ***willingness to devote some hours*** *of his/her daily schedule to studying and practicing, while at the same time being* ***honest with his/her readiness****, it's absolutely doable to pass any CCIE lab in just a few months, as long as he/she improves his/her* ***time management skills*** *to compensate for the strict lab timings.*

--
Tassos
CCIE #19858

Table of Contents

UNI-ENI Vlans vs Private Vlans

UNI-ENI Vlans (or just UNI Vlans)

Types

- Isolated Vlans
- Community Vlans

Characteristics

- Configuration happens under the Vlan
- Port configuration doesn't include Vlan type
- Each port can include many UNI-ENI Vlans
- Apply to access, trunk, tunnel ports
- There is only local significance per switch
- L3 config applies to each Vlan separately
- MAC addresses are learned on each vlan separately

Configuration

IOS

```
ME-3400(config)#vlan 150
ME-3400(config-vlan)#uni-vlan ?
  community  UNI/ENI community VLAN
  isolated   UNI/ENI isolated VLAN
```

IOS

```
vlan 150
 uni-vlan community
```

Verification

IOS

```
ME-3400#sh vlan uni-vlan type

Vlan Type
---- -----------------
1    UNI isolated
150  UNI isolated
1301 UNI community
1302 UNI community
```

Private Vlans

Types

- Primary Vlan
- Secondary Vlans
    - Isolated Vlans
    - Community Vlans

Characteristics

- Configuration happens under the Vlan and under the Port
- Port configuration includes a pair of Private Vlans
- Each port can include only one pair of Private Vlans
- Apply only to access ports
- There can be global significance between multiple trunked switches
- VTPv3 required to transfer them automatically across switches
- L3 config applies to Primary Vlan only
- MAC addresses are replicated from Secondary to Primary Vlans

Configuration

IOS

```
vlan 100
  private-vlan primary
  private-vlan association 200,300
!
vlan 200
  private-vlan isolated
!
vlan 300
  private-vlan community
!
interface GigabitEthernet0/7
 port-type nni
 switchport private-vlan mapping 100 200,300
 switchport mode private-vlan promiscuous
!
interface GigabitEthernet0/11
 switchport private-vlan host-association 100 200
 switchport mode private-vlan host
!
interface GigabitEthernet0/12
 switchport private-vlan host-association 100 300
 switchport mode private-vlan host
```

Primary vlan is configured on every type of port.

Verification

IOS

```
ME-3400#sh vlan private-vlan

Primary Secondary Type              Ports
------- --------- ----------------- -------------------------------------
-----
100     200       isolated          Gi0/7, Gi0/11
100     300       community         Gi0/7, Gi0/12
```

Frame-Relay

Multiprotocol Interconnect over Frame-Relay is defined in RFC 2427.
PPP over Frame Relay is defined in RFC 1973.

FECN/BECN

- FECN (Forward Explicit Congestion Notification)
  - If set to 1, it indicates that congestion was experienced in the direction of the frame transmission, so the destination is informed of that congestion.
- BECN (Backwards Explicit Congestion Notification)
  - If set to 1, it indicates that congestion was experienced in the direction opposite of the frame transmission, so the source is informed of that congestion.

DE

If set to 1 by a DTE device, it indicates that the frame has lower importance than other frames, so when the network becomes congested, DCE devices can discard this frame before discarding other frames that do not have the DE bit set.

LMI

LMI VC status messages provide communication and synchronization between Frame Relay DTE and DCE devices, aka reporting on the status of PVCs.

It's enabled by default in all Frame-Relay interfaces and it's type (Cisco, ANSI, Q933a) is automatically detected.

Use keepalives to track PVC status end-to-end if multiple frame-relay providers are in between end-points.

IOS

```
interface Serial2/0
 encapsulation frame-relay
 frame-relay interface-dlci 100
  class FR-MAPCLASS
!
map-class frame-relay FR-MAPCLASS
 frame-relay end-to-end keepalive mode bidirectional
```

CRC

Frame Relay uses cyclic redundancy check (CRC) as an error-checking mechanism. No error correction takes place.

Frame-Relay Switching

Routers can be configured as Frame Relay switches (frames from a PVC arriving on an incoming interface are switched to another PVC on an outgoing interface, so the incoming DLCI in the arriving frames is replaced by an outgoing DLCI). It applies only to physical interfaces.

IOS

```
frame-relay switching
!
interface Serial2/0
 encapsulation frame-relay
 clockrate 64000
 frame-relay intf-type dce
 frame-relay route 200 interface Serial2/1 201
!
interface Serial2/1
 encapsulation frame-relay
 clockrate 64000
 frame-relay intf-type dce
 frame-relay route 201 interface Serial2/0 200
```

Don't forget to always define one interface (the one providing the clock) of each link as DCE and set the clock rate.

"`frame-relay switching`" might also be required in case of AToM. You should get AToM working without frame-relay switching, but after you reload the router you may get a message "`Must enable frame-relay switching to configure DCE/NNI`" while the bootloader is running.

"`no keepalive`" must be configured in real networks, GNS3 can work without it.

Address Resolution

- IPv4
  - Inverse ARP
  - Static mapping
- IPv6
  - Static mapping

IPv6 doesn't need a map for local ping to work (unlike IPv4).

Inverse ARP

- "`no frame-relay inverse-arp`" disables only the request, replies are always sent
- IPv6 does not use inverse ARP
- point-to-point sub-interfaces do not use inverse ARP

- only directly connected devices can be resolved with inverse ARP (hub-n-spoke is an issue)
- if there is a static map for a protocol for a PVC, inverse ARP is disabled for that PVC
- dynamic mappings created by inverse ARP are overwritten by static mappings
- use "`clear frame-relay inarp`" to clear the dynamic mappings (sometimes shut/no-shut or a reload might be needed)

Inverse ARP should be avoided when told to:

- not use dynamic maps
- not use unnecessary PVCs
- not use undefined PVCs
- use only specific PVCs

map vs interface-dlci

- Frame-Relay "`map`" command is used on
    - multipoint subinterfaces
    - physical interfaces without inverse-arp
    - physical interfaces with IPv6
- Frame-Relay "`interface-dlci`" command is used on
    - point-to-point subinterfaces
    - multipoint subinterfaces with inverse-arp
    - ppp over frame-relay
    - interfaces with map-class required

Most IPv6 routing protocols use Link Local addresses for next hop and neighboring, so these need to be mapped too, like the Global Unicast addresses. **Always define manually the IPv6 Link Local Address on each interface if you plan to use it in a map statement.**

In case of hub-n-spoke scenarios, OSPF adjacencies cannot be established between spokes due to TTL=1.

When a multipoint subinterface is created on a physical interface, all the DLCIs are always assigned to the physical interface, until they are specifically assigned to the subinterfaces.

Always prefer to use point-to-point subinterfaces.

Configurations (IOS)

Physical Interface (IPv4)

```
interface Se0/0
 encapsulation frame-relay
 ip address 10.10.10.10 255.255.255.0
 frame-relay map ip 10.10.10.11 100 broadcast
```

"`frame-relay map`" is need if DLCIs aren't provided by a frame-relay switch.

Physical Interface (IPv6)

```
interface Se0/0
 encapsulation frame-relay
 ipv6 address 2001::10:10:10:1/64
 ipv6 address fe80::1 link-local
 frame-relay map ipv6 2001::10:10:10:2/64 100 broadcast
 frame-relay map ipv6 fe80::2 100 broadcast
```

Mapping is required for both LLAs and GUAs in IPv6.

Point-to-Point Subinterface (IPv4)

```
interface Se0/0.1 point-to-point
 ip address 10.10.10.1 255.255.255.0
 frame-relay interface-dlci 101
```

Point-to-Point Subinterface (IPv6)

```
interface Se0/0.1 point-to-point
 ipv6 address 2001::10:10:10:1/64
 frame-relay interface-dlci 101
```

Multi-point Subinterface (IPv4)

```
interface Se0/0.1 multipoint
 ip address 10.10.10.1 255.255.255.0
 frame-relay map ip 10.10.10.2 102 broadcast
 frame-relay map ip 10.10.10.3 103 broadcast
```

Multi-point Subinterface (IPv6)

```
interface Se0/0.1 multipoint
 ipv6 address 2001::10:10:10:1/64
 ipv6 address fe80::1 link-local
 frame-relay map ipv6 2001::10:10:10:2/64 102 broadcast
 frame-relay map ipv6 fe80::2 102 broadcast
 frame-relay map ipv6 2001::10:10:10:3/64 103 broadcast
 frame-relay map ipv6 fe80::3 103 broadcast
```

Configuration Examples

point-to-point on one side

IOS

```
interface POS2/0
 encapsulation frame-relay
!
interface POS2/0.1 point-to-point
 ip address 1.1.1.1 255.255.255.0
 ipv6 address fe80::1 link-local
 ipv6 address 2001:1:1:1::1/64

 frame-relay interface-dlci 22
```

IOS-XR

```
interface POS2/0
 encapsulation frame-relay
!
interface POS2/0.1 point-to-point
 ipv4 address 1.1.1.1/24
 ipv6 address fe80::1 link-local
 ipv6 address 2001:1:1:1::1/64
 pvc 22
```

physical interface on the other side

IOS

```
interface POS2/0
 encapsulation frame-relay
 ip address 1.1.2.2 255.255.255.0
 ipv6 address 2001:1:1:1::2/64
 frame-relay map ipv6 fe80::1 22 broadcast
 frame-relay map ipv6 2001:1:1:1::1 22 broadcast
 frame-relay map ip 1.1.1.1 22 broadcast
 frame-relay intf-type dce
```

IOS-XR

not supported

PPP over Frame-Relay

Use this when you have frame-relay and you require authentication or other PPP specific characteristics.

DCE Router (IOS)

```
interface Serial2/0
 encapsulation frame-relay
 clock rate 64000
 frame-relay interface-dlci 100 ppp Virtual-Template1
 frame-relay intf-type dce
!
interface Virtual-Template1
 ip address 1.1.1.1 255.255.255.0
```

DTE Router (IOS)

```
interface Serial0/0
 encapsulation frame-relay
 frame-relay interface-dlci 100 ppp Virtual-Template2
!
interface Virtual-Template2
 ip address 1.1.1.2 255.255.255.0
```

Under the virtual-template you can configure whatever parameters are applicable to ppp in general.

You might have issues with POS interfaces and PPPoFR in GNS3. Try to use Serial interfaces instead.

Multilink Frame-Relay

R1 (IOS)

```
interface Serial2/0
 encapsulation frame-relay MFR1
 clock rate 64000
!
interface MFR1
 ip address 12.12.12.2 255.255.255.0
 frame-relay map ip 12.12.12.8 100 broadcast
 frame-relay intf-type dce
```

R2 (IOS)

```
interface Serial0/0
 encapsulation frame-relay MFR1
!
interface MFR1
 ip address 12.12.12.8 255.255.255.0
 frame-relay map ip 12.12.12.2 100 broadcast
```

IOS

```
R1#sh frame-relay multilink
Bundle: MFR1, State = up, class = A, fragmentation disabled
 BID = MFR1
 Bundle links:
  Serial2/0, HW state = up, link state = Up, LID = Serial2/0
```

Frame-relay configuration is the usual one. You can also use MFR subinterfaces.

Hints

If you need somehow to differentiate traffic in a Serial/POS interfaces, then using frame-relay encapsulation on it, you can define subinterfaces based on DLCIs. Another (maybe more complex) solution would be to use multiple ppp virtual-templates.

PPP/Serial/POS

PPP (Point-to-Point Protocol) is defined in RFC 1661.
PPPoE (PPP over Ethernet) is described in RFC 2516.

Serial

Don't forget to to set the clock rate (i.e. 64000) on the DCE interface (usually the one on the service provider router).

PPP

You can use "no peer neighbor route" in order to disable creating a /32 for the peer address.

Multilink PPP

R1 (IOS)

```
interface Serial2/0
 encapsulation ppp
 ppp multilink
 ppp multilink group 1
 clock rate 64000
!
interface Multilink1
 ip address 12.12.12.1 255.255.255.0
 ppp multilink
 ppp multilink group 1
```

R2 (IOS)

```
interface Serial0/0
 encapsulation ppp
 ppp multilink
 ppp multilink group 1
!
interface Multilink1
 ip address 12.12.12.2 255.255.255.0
 ppp multilink
 ppp multilink group 1
```

Multichassis Multilink PPP

SGBP is used between routers to coordinate them for multilink ppp termination.

R4 (IOS)

```
sgbp group SGBP-GRP
sgbp member R5 5.5.5.5
sgbp source-ip 4.4.4.4
!
username SGBP-GRP password 0 SGBP-PASS
!
multilink virtual-template 1
!
interface Virtual-Template1
 ip unnumbered Loopback0
 ppp multilink
```

R5 (IOS)

```
sgbp group SGBP-GRP
sgbp member R4 4.4.4.4
sgbp source-ip 5.5.5.5
username SGBP-GRP password 0 SGBP-PASS
!
multilink virtual-template 1
!
interface Virtual-Template1
 ip unnumbered Loopback0
 ppp multilink
```

IOS

```
R4#sh sgbp
Group Name: SGBP-GRP Ref: 0xDE80000
Seed bid: default, 50, default seed bid setting

  Member Name: R5 State: active Id: 1
  Ref: 0xABC0000
  Address: 5.5.5.5
  Other Active Address: 20.4.5.5
```

PPPoE

IOS

PPPoE server

```
bba-group pppoe global
 virtual-template 1
!
interface Virtual-Template1
 mtu 1492
 ip address 10.10.10.1 255.255.255.0
!
interface X
 pppoe enable group global
```

PPPoE client

```
interface X
 pppoe enable
 pppoe-client dial-pool-number 1
!
interface Dialer1
 mtu 1492
 ip address 10.10.10.2 255.255.255.0
 encapsulation ppp
 dialer pool 1
```

Interface X is assumed to be an ethernet interface.

You must define "encapsulation ppp" under the dialer, otherwise the ppp call won't happen.

Not all routers support the PPPoE functionality.

PPPoE server/client is not supported on IOS-XR of C12k.

PPP Authentication

IOS

Server

```
username USERNAME password PASSWORD
!
interface POS2/0
 encapsulation ppp
 ppp authentication chap
```

Client

```
interface POS2/0
 encapsulation ppp
 ppp chap hostname USERNAME
 ppp chap password PASSWORD
```

If you don't define a chap hostname, then the router's name is used as the username.

In the following example the first router authenticates the second using CHAP (encrypted), while the second router authenticates the first using PAP (cleartext).

IOS

Server & Client #1

```
username R2-USER password R2-PASS
!
interface POS2/0
 encapsulation ppp
 ppp authentication chap
 ppp pap sent-username R1-USER password R1-PASS
```

Server & Client #2

```
username R1-USER password R1-PASS
!
interface POS2/0
 encapsulation ppp
 ppp authentication pap
 ppp chap hostname R2-USER
 ppp chap password R2-PASS
```

POS (Packet over SONET/SDH)

POS default MTU is 4470.

MPLS-TE isn't supported on POS frame-relay subinterfaces on C12k running IOS-XR.

POS Configuration

You will find most configurations parameters under the following command:

IOS

```
R1(config-if)#pos ?
  ais-shut      Send LAIS when shutdown
  delay         Delay POS alarm triggers
  flag          Specify byte value
  framing       specify framing
  report        enable reporting of selected alarms
  scramble-atm  Enable POS SPE scrambling
  threshold     Set BER threshold values
```

Verification checks can be performed with:

IOS

```
R1#sh controllers pos2/0
POS2/0
SECTION
  LOF = 0          LOS    = 0                            BIP(B1) = 0
LINE
  AIS = 0          RDI    = 0          FEBE = 0          BIP(B2) = 0
PATH
  AIS = 0          RDI    = 0          FEBE = 0          BIP(B3) = 0
  PLM = 0          UNEQ   = 1          TIM  = 0          TIU     = 0
  LOP = 0          NEWPTR = 0          PSE  = 0          NSE     = 0

Active Defects: PUNEQ
Active Alarms:  None
Alarm reporting enabled for: SF SLOS SLOF B1-TCA B2-TCA PLOP B3-TCA

Framing: SONET
APS

  COAPS = 0          PSBF = 0
  State: PSBF_state = False
  Rx(K1/K2): 00/00  Tx(K1/K2): 00/00
  S1S0 = 00, C2 = 00
  Remote aps status (none); Reflected local aps status (none)
CLOCK RECOVERY
  RDOOL = 0
  State: RDOOL_state = False
PATH TRACE BUFFER: STABLE
  Remote hostname :
  Remote interface:
  Remote IP addr  :
  Remote Rx(K1/K2):   /    Tx(K1/K2):   /
```

```
BER thresholds:  SF = 10e-3  SD = 10e-6
TCA thresholds:  B1 = 10e-6  B2 = 10e-6  B3 = 10e-6

  Clock source:  line
```

Don't expect all things to work in GNS3.

Keepalives

The keepalive command applies to serial interfaces using HDLC or PPP encapsulation. It does not apply to serial interfaces using Frame Relay encapsulation.

Keepalives are independent between the two peers. One peer end can have keepalives enabled; the other end can have them disabled. Even if keepalives are disabled locally, LCP still responds with ECHOREP packets to the ECHOREQ packets it receives.

CRC

The cyclic redundancy check (CRC) on a serial interface defaults to a length of 16 bits. You can change it to 32 bits.

IOS

```
interface POS2/0
 crc 32
```

IOS

```
R1#sh int pos2/0
POS2/0 is up, line protocol is up
  Hardware is Packet over Sonet
  Internet address is 10.10.10.1/24
  MTU 4470 bytes, BW 155000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 32, loopback not set
```

POS Channel

POS channel link bundling provides load-balancing across all active links in a bundle.

IOS

```
interface pos-channel 1
 ip address 30.30.30.1 255.255.255.0
!
interface pos2/0
 channel-group 1
!
interface pos3/0
 channel-group 1
```

POS link bundling is supported on very specific hardware.

APS

The APS feature provides redundancy and allows for a switchover of POS circuits in the event of circuit failure.

You configure a pair of SONET/SDH lines for line redundancy. When the Working (W) interface fails, the Protect (P) interface quickly assumes the traffic load (usual swichover time is 50 ms)

Most configuration options are found under the "aps" command:

IOS

```
R1(config-if)#aps ?
  authentication  Authentication string
  force           Force channel
  group           Group association
  lockout         Lockout protection channel
  manual          Manually switch channel
  protect         Protect specified circuit
  reflector       Configure for reflector mode APS
  revert          Specify revert operation and interval
  signalling      Specify SONET/SDH K1K2 signalling
  timers          APS timers
  unidirectional  Configure for unidirectional mode
  working         Working channel number
```

Configuration

IOS

```
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface POS2/0
 ip address 10.10.10.1 255.255.255.0
 aps group 10
 aps working 1
!
interface POS3/0
 ip address 20.20.20.1 255.255.255.0
 aps group 10
 aps protect 1 1.1.1.1
 aps revert 1
```

You need the configure a similar setup on the peer router too.

You can have the Working and Protect interfaces on different routers and they will communicate each other using PGP (Protect Group Protocol), which runs over UDP.

IOS

```
R1#sh aps
POS3/0 APS Group 10: protect channel 0 (Inactive)
        Working channel 1 at 1.1.1.1 (Enabled)
        bidirectional, revertive (60 seconds)
        PGP timers (default): hello time=1; hold time=3
                hello fail revert time=120
        SONET framing; SONET APS signalling by default
        Received K1K2: 0x00 0x00
                No Request (Null)
        Transmitted K1K2: 0x00 0x05
                No Request (Null)
        Remote APS configuration: (null)

POS2/0 APS Group 10: working channel 1 (Active)
        Protect at 1.1.1.1
        PGP timers (from protect): hello time=1; hold time=3
        SONET framing
        Remote APS configuration: (null)
```

You need an ADM between the routers for the K1/K2 signals to work.

RIP/RIPng

RIPv1 (Routing Information Protocol v1) is defined in RFC 1058.
RIPv2 is defined in RFC 2453.
RIPng (RIP for IPv6) is defined in RFC 2080.

RIP uses UDP port 520.

Metric = hop count (1-16) - use offset-list to modify

Admin distance is 120.

RIP v1

If the advertised prefix is part of a directly connected network, the subnet mask of that connected interface is used as the subnet mask of the prefix. Otherwise, major classes A/B/C are used accordingly.

Use secondary ip addresses on intermediate links to fix the discontinuous class issues in RIP v1.

RIP v1 updates are sent as broadcast to 255.255.255.255

RIP v2

- classless routing
- next-hop included in updates
- authentication
- external route tags
- multicast updates (to 224.0.0.9)

RIP default mode

- send v1 updates
- listen to v1/v2 updates

Passive interface

- listens to RIP messages (use filtering to block if required)
- doesn't send RIP updates (unless a specific neighbor is configured)
- updates the routing table

In IOS, you can use "ip rip triggered" under serial interfaces (on both neighbors) to minimize the number of updates.

Unicast updates

You can specify a specific neighbor for sending unicast updates.

IOS

```
router rip
 neighbor 10.1.1.2
```

IOS-XR

```
router rip
 neighbor 10.1.1.2
```

Use the command "no validate-update-source" under the RIP process if you want to enable exchange of routes between neighbors with different networks.

Configuration

IOS

```
router rip
 version 2
 network 10.0.0.0
 no auto-summary
```

IOS-XR

```
router rip
 interface X
 !
 no auto-summary
```

IOS-XR has v2 enabled by default. You need the change the send/receive version under the interface if v1 is required.

Unless told otherwise, always enable v2 and disable auto-summary.

Route Summarization

IOS

```
interface Serial2/0
 ip summary-address rip x.x.x.x y.y.y.y
```

IOS-XR

not supported

Route Filtering

- prefix-list & gateway
  - `distribute-list prefix PREFIXES gateway SOURCES in`
- extended ACL
  - `distribute-list X in`
  - `access-list x permit ip host SOURCE host PREFIX`

Authentication

Two methods:

- clear text
- MD5

IOS

```
interface X
 ip rip authentication mode md5
 ip rip authentication key-chain KEYCHAIN
!
key chain KEYCHAIN
 key 1
  key-string TESTPASS
```

IOS-XR

```
router rip
 interface TenGigE0/0/0/0
  authentication keychain KEYCHAIN mode md5
!
key chain KEYCHAIN
 key 1
  key-string TESTPASS
```

PE-CE

CE

IOS

```
router rip
 version 2
 network 10.0.0.0
 no auto-summary
```

IOS-XR

```
router rip
 interface X
 !
 no auto-summary
```

PE

IOS

```
router rip
 address-family ipv4 vrf VPN
  network 10.0.0.0
  no auto-summary
  version 2
 exit-address-family
```

IOS-XR

```
router rip
 vrf VPN
  interface X
  !
  no auto-summary
```

In IOS, if RIP v2 is to be used, then it must be defined under the ipv4 vrf address-family on the PE.

RIPng (IPv6)

Same as RIPv2, except:

- uses UDP port 521 (can be changed)
- updates are sent to FF02::9 (can be changed)
- metric can be changed per incoming interface (not per received/advertised prefix)

IOS

```
interface X
 ipv6 rip RIPNG enable
!
ipv6 router rip RIPNG
 port 528 multicast-group FF02::8
```

IOS-XR

not supported

Defining the RIPng process is not required in IOS; it gets automatically created once you enable it under an interface. Removing the RIPng process will also remove all other configuration from interfaces.

Since you can have multiple RIPng processes, you must use a different UPD port for every RIPng process to differentiate the incoming updates.

Process name is only locally significant.

EIGRP

EIGRP (Enhanced Interior Gateway Routing Protocol) is described in draft-savage-eigrp.

EIGRP is protocol number 88.

Packets are sent to multicast 224.0.0.10 (IPv4) or FF02::A (IPv6).

Metrics

- **bandwidth**
  - minimum bandwidth (kbps) => 10^7 / bandwidth
- **delay**
  - total route delay (tens of microseconds) => delay/10
- **reliability**
  - likelihood of successful packet transmission (0-255)
- **load**
  - effective load of the route (0-255)
- **mtu**
  - minimum MTU size (bytes)

All metrics are calculated from the outgoing interface towards the destination.

metric = [K1 * bandwidth + (K2 * bandwidth) / (256 - load) + K3 * delay] * [K5 / (reliability + K4)]

Default

- K1=K3=1
- K2=K4=K5=0

Bandwidth and Delay are the ones used by default.

metric = (10^7/bandwidth + delay/10) **x 256**

Example

- minimum bandwidth = 100 Kbps
- total delay = 20000 + 5000 = 25000 usec
- metric = (10^7/100 + 25000/10) x 256 = 26240000

Mismatched K values (weights for EIGRP metrics) can prevent neighbor relationships.

IOS

```
router eigrp 1
 metric weights 0 1 0 1 0 0
```

IOS-XR

```
router eigrp 1
 address-family ipv4
  metric weights 0 1 0 1 0 0
```

Route Selection

- The lowest calculated metric from a router to a destination is called the **feasible distance (FD)** of that destination
- If a neighbor's advertised distance to a destination is lower than router's FD, then that neighbor becomes a **feasible successor (FS)** to the specific destination
- Every destination for which there is at least one FS, will be installed in the router's EIGRP topology
- For every destination in the router's EIGRP topology, the route with the lowest metric will be installed in the RIB
- The neighbor advertising that route will be successor for that destination

Load Balancing

By default traffic to equal cost paths (up to 4) is load balanced.

Unequal-Cost Load Balancing

You can use a **variance** as a multiplier to determine which routes are feasible for unequal-cost load balancing, according to the following condition:

route metric < lowest cost metric * variance

Routes that follow the above rule are installed into RIB as long as maximum-paths (default=4) is not exceeded.

Traffic Sharing

- balanced (default)
  - traffic is distributed proportionately to the ratios of the route metrics
- minimum
  - traffic is distributed equally across all paths that have a cost equal to the minimum cost path

In general:

- variance
    - affects what non-lowest cost routes are installed into RIB
- traffic-share
    - affects how traffic is distributed across best routes

In order to use only one path for traffic forwarding, but install more different-cost paths into RIB (for faster convergence), you can use a combination of both features.

IOS

```
router eigrp 1
 traffic-share min across-interfaces
 variance x
```

IOS-XR

```
router eigrp 1
 address-family ipv4
  variance x
```

Traffic-share is not supported in IOS-XR.

Stub Routing

A router that is configured as a **stub** with the "`eigrp stub`" command cannot be used as transit and shares **connected and summary** routing information with all neighbor routers by default. Generally, the following can be permitted/denied explicitly :

- connected
- static
- summary
- redistributed
- leak-map
- receive-only

Stub routing also minimizes the exchange of queries.

Route Summarization

IOS

```
interface X
 ip summary-address eigrp 100 x.x.x.x y.y.y.y
```

IOS-XR

```
router eigrp 100
 address-family ipv4
  interface X
   summary-address x.x.x.x/y
```

Default route can be originated the same way

Split-horizon

Split horizon blocks route information from being advertised by a router out of any interface from which that information originated. With non-broadcast networks (such as Frame Relay multipoint), you may want to disable it with "`no ip split-horizon eigrp x`".

An alternative is **poison-reverse**: Once you learn of a route through an interface, advertise it as unreachable back through that same interface.

Configuration

IOS

```
router eigrp 1
 network 1.1.1.0 0.0.0.255
!
ipv6 router eigrp 1
!
interface X
 ipv6 eigrp 1
```

IOS-XR

```
router eigrp 1
 address-family ipv4
  interface X
!
 address-family ipv6
  interface X
```

"`ip hello-interval eigrp x`" and "`ip hold-time eigrp x`" under an interface can be used to tune the convergence time.

Authentication

IOS

```
interface X
 ip authentication mode eigrp 1 md5
 ip authentication key-chain eigrp 1 KEYCHAIN
!
key chain KEYCHAIN
 key 1
  key-string TESTPASS
```

IOS-XR

```
router eigrp 1
 address-family ipv4
  interface X
   authentication keychain KEYCHAIN
!
key chain KEYCHAIN
 key 1
  key-string TESTPASS
  send-lifetime 1:00:00 february 01 2014 infinite
  accept-lifetime 1:00:00 february 01 2014 infinite
```

Only MD5 is supported.

Key-chains in IOS-XR might require the use of lifetimes.

PE-CE

R1 (CE)

IOS

```
router eigrp 1
 network 1.1.1.0 0.0.0.255
```

IOS-XR

```
router eigrp 1
 address-family ipv4
  interface X
```

R2 (PE)

IOS

```
router eigrp 100
 address-family ipv4 vrf VPN autonomous-system 1
  network 1.1.1.0 0.0.0.255
 exit-address-family
```

IOS-XR

```
router eigrp 100
 vrf VPN
  address-family ipv4
   autonomous-system 1
   interface X
```

CE EIGRP process number and PE EIGRP autonomous-system must match.

Some software releases require the manual addition of "`no auto-summary`" under the EIGRP process.

EIGRP adjacency might not get established in IOS devices if you initially forget to add the autonomous-system number and add it later. Try to remove the whole EIGRP config and then reapply it if this is the case.

For **IPv6 VRFs** you have to use the **named configuration** on IOS (see below).

Verification

IOS

```
R1#sh ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(1)
H   Address                 Interface       Hold Uptime    SRTT   RTO  Q
Seq
                                            (sec)          (ms)       Cnt
Num
0   10.1.2.2                Fa0/0             14 00:32:28 1280   5000  0
4

R2#sh ip eigrp vrf VPN_A neighbors
EIGRP-IPv4 Neighbors for AS(1) VRF(VPN)
H   Address                 Interface       Hold Uptime    SRTT   RTO  Q
Seq
                                            (sec)          (ms)       Cnt
Num
0   10.1.2.1                Fa1/0             13 00:32:47   54    324  0
4
```

IOS-XR

```
GSR#sh eigrp vrf VNP neighbors

Sun Jan 12 19:23:12.845 UTC

IPv4-EIGRP neighbors for AS(1) vrf VPN

H   Address                 Interface        Hold Uptime    SRTT   RTO  Q
Seq
                                             (sec)          (ms)        Cnt
Num
0   10.1.0.10               Gi0/1/0/1.1019     13 00:01:10    8    200  0
4
```

IOS

```
R1#sh ip eigrp topology
EIGRP-IPv4 Topology Table for AS(1)/ID(1.1.1.1)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.5.8.0/24, 1 successors, FD is 30720
        via 10.1.2.2 (30720/28160), FastEthernet0/0
P 8.8.8.8/32, 1 successors, FD is 158720
        via 10.1.2.2 (158720/156160), FastEthernet0/0
P 10.1.2.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
P 1.1.1.1/32, 1 successors, FD is 128256
        via Connected, Loopback0

R2#sh ip eigrp vrf VPN topology
EIGRP-IPv4 Topology Table for AS(1)/ID(10.1.2.2) VRF(VPN)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.5.8.0/24, 1 successors, FD is 28160
        via VPNv4 Sourced (28160/0)
P 8.8.8.8/32, 1 successors, FD is 156160
        via VPNv4 Sourced (156160/0)
P 10.1.2.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet1/0
P 1.1.1.1/32, 1 successors, FD is 156160
        via 10.1.2.1 (156160/128256), FastEthernet1/0
```

IOS-XR

```
GSR#sh eigrp vrf VPN topology

Sun Jan 12 19:30:02.425 UTC
```

```
IPv4-EIGRP Topology Table for AS(1)/ID(19.19.19.19) VRF: VPN

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.10/32, 1 successors, FD is 130816
        via 10.1.0.10 (130816/128256), GigabitEthernet0/1/0/1.1019
P 10.9.9.9/32, 1 successors, FD is 130816
        via VPNv4 Sourced (130816/0)
P 10.0.0.0/24, 1 successors, FD is 2816
        via VPNv4 Sourced (2816/0)
P 10.1.0.0/24, 1 successors, FD is 2816
        via Connected, GigabitEthernet0/1/0/1.1019
```

EIGRP route attributes are transferred as extended communities, EIGRP metric is transferred as BGP MED.

IOS

```
R2#sh bgp vpnv4 unicast vrf VPN 8.8.8.8/32
BGP routing table entry for 100:1:8.8.8.8/32, version 10
Paths: (1 available, best #1, table VPN)
  Not advertised to any peer
  Local
    5.5.5.5 (metric 3) from 3.3.3.3 (3.3.3.3)
      Origin incomplete, metric 156160, localpref 100, valid, internal,
best
      Extended Community: RT:100:1 Cost:pre-bestpath:128:156160
        0x8800:32768:0 0x8801:1:130560 0x8802:65281:25600
0x8803:65281:1500
        0x8806:0:0
      Originator: 5.5.5.5, Cluster list: 3.3.3.3
      mpls labels in/out nolabel/23
```

Named vs AS

The following is applicable only to IOS.

- Named (new)
  - supports VRFs under IPv4 and IPv6
  - supports IPv6 VRF-Lite
  - interface configuration goes under the af-interface under the address-family
- AS (old)
  - supports VRFs under IPv4 only
  - interface configuration goes under the physical interface

Named Configuration (new)

IOS

```
interface POS2/0
 ip address 10.10.10.1 255.255.255.0
 ipv6 address 2001:20:20:20::1/64
 ipv6 eigrp 2
!
router eigrp EIGRP1
 !
 address-family ipv4 unicast autonomous-system 1
  !
  af-interface POS2/0
   hello-interval 20
   hold-time 60
  exit-af-interface
  !
  network 10.10.10.0 0.0.0.255
 exit-address-family
 !
 address-family ipv6 unicast autonomous-system 2
  !
  af-interface POS2/0
   authentication mode md5
   authentication key-chain KEYCHAIN
  exit-af-interface
  !
 exit-address-family
```

AS Configuration (old)

IOS

```
interface POS2/0
 ip address 10.10.10.2 255.255.255.0
 ip hello-interval eigrp 1 20
 ip hold-time eigrp 1 60
 ipv6 address 2001:20:20:20::2/64
 ipv6 eigrp 2
 ipv6 authentication mode eigrp 2 md5
 ipv6 authentication key-chain eigrp 2 KEYCHAIN
!
router eigrp 1
 network 10.10.10.0 0.0.0.255
!
ipv6 router eigrp 2
```

IOS

```
R1#sh eigrp address-family ipv4 neighbors
EIGRP-IPv4 VR(EIGRP1) Address-Family Neighbors for AS(1)
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   10.10.10.2              PO2/0             45 00:56:05   46   276  0
30

R2#sh ip eigrp neighbors
EIGRP-IPv4 Neighbors for AS(1)
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q
Seq
                                            (sec)         (ms)       Cnt
Num
0   10.10.10.1              PO2/0             43 00:56:23   42   252  0
29


R1#sh eigrp address-family ipv6 int det
EIGRP-IPv6 VR(EIGRP1) Address-Family Interfaces for AS(2)
                        Xmit Queue   Mean   Pacing Time   Multicast
Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer
Routes
PO2/0              1        0/0       1601       0/1
7969           0
  Hello-interval is 5, Hold-time is 15
  Split-horizon is enabled
  Next xmit serial <none>
  Un/reliable mcasts: 0/0  Un/reliable ucasts: 4/8
  Mcast exceptions: 0  CR packets: 0  ACKs suppressed: 0
  Retransmissions sent: 3  Out-of-sequence rcvd: 2
  Topology-ids on interface - 0
  Authentication mode is md5,  key-chain is "KEYCHAIN"

R2#sh ipv6 eigrp int det
EIGRP-IPv6 Interfaces for AS(2)
                        Xmit Queue   Mean   Pacing Time   Multicast
Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer
Routes
PO2/0              1        0/0        32        0/1
50           0
  Hello-interval is 5, Hold-time is 15
  Split-horizon is enabled
  Next xmit serial <none>
  Un/reliable mcasts: 0/0  Un/reliable ucasts: 3/6
  Mcast exceptions: 0  CR packets: 0  ACKs suppressed: 0
```

```
Retransmissions sent: 2  Out-of-sequence rcvd: 1
Topology-ids on interface - 0
Authentication mode is md5,  key-chain is "KEYCHAIN"
```

OSPFv2/OSPFv3

OSPFv2 (Open Shortest Path First v2) is defined in RFC 2328.
OSPFv3 is defined in RFC 5340.
OSPFv2 as PE/CE protocol is defined in RFC 4577.

OSPF is protocol 89.

Sends updates to multicast 224.0.0.5 (all OSPF routers) and 224.0.0.6 (all DR routers)

Adjacencies

Adjacency can be formed between different networks if "ip unnumbered" is used on both sides.

If multiple "`network`" commands are used, the most specific wins.

The following must match for adjacency to be successful:

- area
- hello/dead timers
- mtu
- network type
- stub
- authentication

DR election per network type

- DR/BDR
  - broadcast (default on ethernet, multicast hellos)
  - non-broadcast (default on frame-relay, unicast hellos)
- no DR/BDR
  - point-to-point (default on serial/pos, multicast hellos)
  - point-to-multipoint (multicast hellos)
  - point-to-multipoint non-broadcast (unicast hellos)

Use the "`neighbor`" command to send unicast hellos.

You can use the commands "`ip ospf hello-interval`" and "`ip ospf dead-interval`" in order to tune the convergence time. Fast hellos (1 sec) are also possible using the "`minimal`" keyword.

By default the loopback interface is advertised as a /32 (stub). Use network-type point-to-point under the loopback interface to advertise it with the original subnet mask.

Path Selection

- "`bandwidth`" under the interface (not applicable to IOS-XR)
- "`ip ospf cost`" under the interface
- "`auto-cost reference-bandwidth`" under the ospf process
- "`neighbor x.x.x.x cost`" under the ospf process

OSPF Route Preference

O - OSPF (intra-area)
IA - OSPF inter area
E1 - OSPF external type 1
E2 - OSPF external type 2
N1 - OSPF NSSA external type 1
N2 - OSPF NSSA external type 2

Distance and metric are evaluated as a second step, between routes of same type.

LSAs

LSA types carried in IPv6 Stub areas:

- Router-LSAs
- Network-LSAs
- Inter-Area-Prefix-LSAs
- Link-LSAs
- Intra-Area-Prefix-LSAs

LSA types carried in IPv6 NSSA areas:

- Router-LSAs
- Network-LSAs
- Inter-Area-Prefix-LSAs
- Link-LSAs
- Intra-Area-Prefix-LSAs
- NSSA-LSAs

LSA types carried in IPv4 Stub areas:

- Router-LSAs
- Network-LSAs
- Summary-LSAs

LSA types carried in IPv4 NSSA areas:

- Router-LSAs
- Network-LSAs
- Summary-LSAs
- NSSA-LSAs

NSSA

The NSSA ASBR redistributes routes into OSPF and originates the Type-7 LSAs.

Type-7 LSAs are only flooded within the originating NSSA area.

Type-7 LSAs have a propagate (P) bit that, when set, tells an NSSA ABR to translate a Type-7 LSA into a Type-5 LSA.

The NSSA ABR translates Type-7 LSAs into Type-5 LSAs and floods them into area 0.

If there are multiple NSSA ABRs, the router with the highest Router ID is elected as the translator.

The NSSA ASBR and the NSSA ABR can be the same router.

Preference between two Type-7 LSAs is determined by the following tie breaker rules:

- An LSA with the P-bit set is preferred over one with the P-bit clear
- If the P-bit settings are the same, the LSA with the higher router ID is preferred

Links

- IETF - RFC 3101

LSDB optimization

You can decrease the LSA DB size by doing one or more of the following:

- configure interfaces as unnumbered
- remove network LSAs (caused by DRs) by using point-to-point as network type on interfaces
- remove transit prefixes by activating prefix-suppression

Prefixes that will be removed by prefix-suppression can be found under "Link connected to: a Stub Network" in Router LSAs (loopbacks, secondary IPs and passive interfaces are excluded).

LSA flood-reduction

OSPF requires every LSA to be refreshed by default every 1800 seconds (30 mins) or else the LSA will expire when it reaches 3600 seconds (1 hour).

When flood-reduction is enabled on a router (towards a neighbor), then this router will flood its self-originated LSAs with the DoNotAge (DNA) bit set, so they do not have to be re-flooded every 30 mins. Of course any change in the contents of the LSA will cause the new LSA to be re-flooded (again with the DoNotAge bit set).

IOS

```
interface X
 ip ospf flood-reduction
 ipv6 ospf flood-reduction
```

IOS-XR

```
router ospf X
 flood-reduction enable
 area 0
  flood-reduction enable
  interface X
   flood-reduction enable
```

In IOS-XR, flood-reduction can be configured under the ospf process, under a specific area and under a specific interface.

Prefer to enable it on stable topologies.

The **demand-circuit** offers the same functionality plus the suppression of periodic hello packets.

Links

- IETF - RFC 1793

Route Filtering

- distribute-list
  - in: filter the routes from entering the RIB
  - out: filter the redistributed routes (E1/E2) entering OSPF on an ASBR
- stub area
  - stub (filter LSA-5)
  - totally stub (filter LSA-3/4/5)
  - nssa (filter LSA-5, allow LSA-7)
  - totally nssa (filter LSA-3/4/5, allow LSA-7)
- LSA-3 prefix filter
  - "`area x filter-list prefix x`"

LSA Searching

Depending on what type of LSAs you're searching for, you can use the following commands to do so:

IOS

```
sh ip ospf database router | i Link State ID
  Link State ID: x.x.x.x

sh ip ospf database network | i Link State ID
  Link State ID: x.x.x.x (address of Designated Router)

sh ip ospf database summary | i Link State ID
  Link State ID: x.x.x.x (Summary Network Number)

sh ip ospf database asbr-summary | i Link State ID
  Link State ID: x.x.x.x (AS Boundary Router address)

sh ip ospf database external | i Link State ID
  Link State ID: x.x.x.x (External Network Number)
```

IOS-XR

```
sh ospf database router | i Link State ID
  Link State ID: x.x.x.x

sh ospf database network | i Link State ID
  Link State ID: x.x.x.x (address of Designated Router)

sh ospf database summary | i Link State ID
  Link State ID: x.x.x.x (Summary Network Number)

sh ospf database asbr-summary | i Link State ID
  Link State ID: x.x.x.x (AS Boundary Router address)

sh ospf database external | i Link State ID
  Link State ID: x.x.x.x (External Network Number)
```

Searching for IPv6 is a little bit different, because the IPv6 prefix information is stored in another attribute.

Summarization

- **Type-3 summary**
    - at ABR
    - `area x range 10.10.10.0 255.255.255.0` (IOS)
    - `area 1 range 10.10.10.0/24` (IOS-XR)
- **Type-5 summary**
    - at ASBR
    - `summary-address 20.20.20.0 255.255.255.0` (IOS)
    - `summary-prefix 20.20.20.0/24` (IOS-XR)

Both types of summarization can also accept "`not-advertise`" as parameter.

Sub-routes must pre-exist in order for the summaries to be advertised.

OSPFv3

Multicast addresses have become **FF02::x** from 224.0.0.x (where x=5 for all OSPF routers, or x=6 for all DR routers).

If there is no IPv4 address assigned to any interface, then you must manually configure an **IPv4-formatted router-id** under the OSPFv3 process.

OSPFv3 runs per-link instead of per-subnet.

You cannot automatically detect OSPFv3 neighbors when using NBMA interfaces. You must manually configure your router to detect neighbors when using an NBMA interface.

All manually configured neighbors in OSPFv3 must be identified by their link-local IPv6 address.

On all OSPFv3 interfaces except virtual links, OSPFv3 packets are sent using the interface's associated link-local unicast address as the source address.

On virtual links, a global scope IPv6 address must be used as the source address for OSPFv3 packets.

Router-LSAs (Type-1) and Network-LSAs (Type-2) no longer contain network addresses, but simply express topology information.

Link-LSAs (Type-8) include the prefixes which are configured on links and are flooded only on local-link scope. **Link-local addresses appear only in Link-LSAs**.

For Stub areas, the Inter-area Prefix LSA can only be a default route.
For NSSA areas, the AS-External LSA can also be a default route.

IOS

```
ipv6 router ospf 1
 router-id 2.2.0.2
!
interface Ethernet1/0
 ipv6 ospf 1 area 0
```

IOS-XR

```
router ospfv3 1
 router-id 2.2.0.8
 address-family ipv6 unicast
 area 0
  interface Loopback0
   passive
  !
  interface GigabitEthernet0/2/1/1
```

A use for running multiple OSPFv3 instances is to have a single link belong to two or more OSPFv3 areas. Also on a LAN you can have multiple adjacencies between different routers, each one on a separate OSPFv3 process/instance.

OSPFv2 Authentication

- Null (Type 0) - default
- Plain-text (Type 1)
- MD5 (Type 2)

In IOS, you can configure the authentication type under the ospf process or under the interface.

IOS

```
router ospf 1
 area 0 authentication
!
interface X
 ip ospf authentication
 ip ospf authentication-key xxx
```

IOS

```
router ospf 1
 area 0 authentication message-digest
!
interface X
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 xxx
```

In IOS-XR, you can configure the authentication type/key under the ospf process, under the area or under the interface.

IOS-XR

```
router ospf 1
 authentication-key xxx
 authentication
 area 0
  authentication-key xxx
  authentication
  interface X
   authentication-key xxx
   authentication
```

IOS-XR

```
router ospf 1
 authentication message-digest
 message-digest-key 1 md5 xxx
  area 0
  authentication message-digest
  message-digest-key 2 md5 xxx
  interface X   authentication message-digest
   message-digest-key 3 md5 xxx
```

You can always use various combination of enabling authentication for a specific area (under the ospf process) or for a specific adjacency (under the interface).

More specific configurations override the less specific ones.

OSPFv3 Authentication

You can configure an authentication (AH) or encryption (ESP) policy, either on an interface or for an OSPFv3 area/process.

IPSec AH

- Authentication
  - MD5
  - SHA1

IPSec ESP

- Encryption
  - 3DES
  - AES (128,192,256 bits)

    - DES
    - NULL
- Authentication
    - MD5
    - SHA1

To use the IPsec AH (for authentication), you must use commands with the "`authentication`" keyword.

To use the IPsec ESP (for authentication & confidentiality), you must use commands with the "`encryption`" keyword.

ESP may use encryption and authentication or only authentication (when encryption=null), but is not recommended.

IOS

```
ipv6 router ospf 1
 area 0 authentication ipsec spi 256 md5 yyy
 area 0 encryption ipsec spi 256 esp 3des md5 yyy
!
interface X
 ipv6 ospf authentication ipsec spi 256 md5 zzz
 ipv6 ospf encryption ipsec spi 256 esp 3des md5 zzz
```

IOS > 12.3(4)T is required for OSPFv3 IPsec authentication.

IOS-XR

```
router ospfv3 1
 authentication ipsec spi 256 md5 password xxx
 encryption ipsec spi 256 esp 3des password xxx
!
 area 0
  authentication ipsec spi 256 md5 password yyy
  encryption ipsec spi 256 esp 3des password yyy
!
  interface GigabitEthernet0/2/1/1.78
   authentication ipsec spi 256 md5 password zzz
   encryption ipsec spi 256 esp 3des password zzz
```

More specific configurations override the less specific ones.

Links

- IETF - RFC 4552

Options Bits

Hello/DBD/LSA Options Bits

- V6 bit: It should be set, unless the router will not participate in IPv6 topology calculation and IPv6 transit routing. If this bit is clear, the router/link should be excluded from any IPv6 routing calculations.
- R bit: It should be set, unless the router will not participate in any transit routing. It allows the router to participate in the unicast topology, but does not allow transit traffic.
- E bit: It should be set if the interface attaches to a regular area (i.e., not a stub or NSSA area).
- N bit: It should be set if the interface attaches to an NSSA area.
- DC bit: This bit describes the router's handling of demand circuits. It should be set in Hellos/DBDs if the router wishes to suppress the sending of future Hellos over the interface. It should be set in LSAs, if the router can correctly process the DoNotAge bit when it appears in the LS age field of LSAs.

IPv6 Prefix Options Bits

- NU bit: The "No Unicast" capability bit. If set, the prefix should be excluded from IPv6 unicast calculations. If not set, it should be included.
- LA bit: The "Local Address" capability bit. If set, the prefix is actually an IPv6 interface address of the Advertising Router.
- P bit: The "Propagate" bit. Set on NSSA area prefixes that should be readvertised by the translating NSSA area border.
- DN bit: The "Down" bit. This bit controls an inter-area-prefix-LSAs or AS-external-LSAs re-advertisement in a VPN environment. It is used for loop prevention in PE=>CE=>PE advertisements and should not be checked in CE multi-vrf (vrf-lite) scenarios.

Special multi-hop adjacencies

- virtual-link
  - connects two areas 0 or extends area 0 across a transit area
  - uses a transit area in order to connect areas 0 or extend area 0
  - used in normal environments with multiple areas 0 or area 0 extension
  - configured betweens two ABRs under the OSPF process
- sham-link
  - connects two areas X (including 0)
  - uses the MPLS core in order to connect the areas
  - used in MPLS VPN environments with backdoor links
  - configured betweens two PEs/ABRs under the OSPF vrf process

Virtual-Link

All areas in an OSPF autonomous system must be connected to area 0. When this is not possible in terms of direct connectivity, then **a virtual-link can be used in order to connect the non-backbone areas to area 0, as long as there is a common area between them.**

- connect two areas 0
    - R1 <=(area 0)=> R2 <=(area 1)=> R3 <=(area 0)=> R4
    - a virtual link can be configured between ABRs R2,R3 that connect to area 0 from different sides and have a common area between them
- extend area 0
    - R1 <=(area 0)=> R2 <=(area 1)=>R3 <=(area 2)=> R4
    - a virtual link can be configured between ABRs R2,R3 that connect to a common area, with only one ABR directly connected to area 0.
    - area 0 is extended to R3, in order to serve area 2

If a common area does not exist between the ABRs, then an additional area can be created to become the transit area.

The transit area through which the virtual link is configured, must have full routing information, so it cannot be any type of stub area. If this is the case, a GRE tunnel can be used to connect the two areas 0.

For virtual-links in OSPFv3 you have to use the remote neighbor's router-id (IPv4 format).

ABR #1

IOS

```
router ospf 1
 area 1 virtual-link 2.2.2.2
```

IOS-XR

```
router ospf 1
 area 1
  virtual-link 2.2.2.2
```

ABR #2

IOS

```
router ospf 1
 area 1 virtual-link 3.3.3.3
```

IOS-XR

```
router ospf 1
 area 1
  virtual-link 3.3.3.3
```

Sham-Link

To make a route through an MPLS backbone appear to be an intra-area route, it is necessary to make it appear as if there is **an intra-area link connecting the two PE routers**. A sham link can be thought of as a indirect relation between two VRFs. If two VRFs are to be connected by a sham link, **each VRF must be associated with a "Sham Link Endpoint Address", a 32-bit IPv4 address** that is treated as an address of the PE router containing that VRF.

The Sham Link Endpoint Address is an address in the VPN's address space, not the SP's address space. **It must not be advertised inside customer's OSPF**, because when there is no BGP VPNv4 route to the Sham Link Endpoint Address, that address must become unreachable, so that the sham link comes down.

The sham link is an unnumbered point-to-point intra-area link and is advertised as a **type 1 LSA**.

Sham links are treated as OSPF Demand Circuits. This means that LSAs will be flooded over them, but periodic refresh traffic will be avoided. Normal flooding is done over the backdoor link, but if that fails, flooding will occur over the sham-link (because LSA synchronization between sites must continue).

Configuration Steps

- Create a /32 loopback that belongs to the relevant VRF on both PEs
- Advertise the above /32 into BGP VPNv4 on both PEs
- Don't advertise the above /32 into the OSPF vrf process on both PEs
- Create a sham-link between the above /32 of the PEs under the OSPF vrf process

PE1

IOS

```
interface Loopback1
 vrf forwarding VPN
 ip address 1.1.1.1 255.255.255.255
!
router ospf 100 vrf VPN
 area 0 sham-link 1.1.1.1 2.2.2.2
```

IOS-XR

```
interface Loopback1
 vrf VPN
 ipv4 address 1.1.1.1/32
!
router ospf 100
 vrf VPN
  area 0
   sham-link 1.1.1.1 2.2.2.2
```

PE2

IOS

```
interface Loopback1
 vrf forwarding VPN
 ip address 2.2.2.2 255.255.255.255
!
router ospf 100 vrf VPN
 area 0 sham-link 2.2.2.2 2.1.1.1
```

IOS-XR

```
interface Loopback1
 vrf VPN
 ipv4 address 2.2.2.2/32
!
router ospf 100
 vrf VPN
  area 0
   sham-link 2.2.2.2 1.1.1.1
```

Verification

IOS

```
R1#sh ip ospf sham-links
Sham Link OSPF_SL0 to address 2.2.2.2 is up
Area 0 source address 1.1.1.1
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 1 State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
    Hello due in 00:00:04
    Adjacency State FULL (Hello suppressed)
    Index 2/2, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

```
R1#sh ip ospf interface | b _SL
OSPF_SL0 is up, line protocol is up
  Internet Address 0.0.0.0/0, Area 0
  Process ID 100, Router ID 10.10.10.1, Network Type SHAM_LINK, Cost: 1
  Topology-MTID    Cost    Disabled    Shutdown      Topology Name
        0           1         no          no            Base
  Configured as demand circuit.
  Run as demand circuit.
  DoNotAge LSA allowed.
  Transmit Delay is 1 sec, State POINT_TO_POINT
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:06
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
```

```
    Adjacent with neighbor 10.10.10.2  (Hello suppressed)
  Suppress hello for 1 neighbor(s)


R1#sh ip ospf neighbor
Neighbor ID     Pri   State           Dead Time   Address
Interface
...
10.10.10.2        0   FULL/  -           -        2.2.2.2
OSPF_SL0
...

R1#sh ip ospf neighbor detail | b 2.2.2.2
 Neighbor 10.10.10.2, interface address 2.2.2.2
    In the area 0 via interface OSPF_SL0
    Neighbor priority is 0, State is FULL, 6 state changes
    DR is 0.0.0.0 BDR is 0.0.0.0
    Options is 0x32 in Hello (E-bit, L-bit, DC-bit)
    Options is 0x72 in DBD (E-bit, L-bit, DC-bit, O-bit)
    LLS Options is 0x1 (LR)
    Neighbor is up for 00:12:16
    Index 2/2, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

IOS-XR

```
GSR#sh ospf vrf VPN sham-links

Sham Links for OSPF 2, VRF VPN

Sham Link OSPF_SL0 to address 2.2.2.2 is up
Area 0, source address 1.1.1.1
IfIndex = 2
  Run as demand circuit
  DoNotAge LSA allowed., Cost of using 1
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:08
    Adjacency State FULL (Hello suppressed)
    Number of DBD retrans during last exchange 0
    Index 2/2, retransmission queue length 0, number of retransmission 0
    First 0(0)/0(0) Next 0(0)/0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Both sham-links and virtual-links can have most of their "interface" attributes (hellos, cost, authentication, etc.) configured (IOS-XR gives more options).

The OSPF Sham Link endpoint address must not be used as the endpoint address of an OSPF Virtual Link.

OSPF Multi-Area

Applicable to OSPFv2 and OSPFv3.

It allows **a link to be configured in more than one area**, so that the link could be considered as an intra-area link in all those areas and **get preference over inter-area links**.

It exists as a logical construct over an existing primary interface for OSPF; however, the neighbor state on the primary interface is independent of the multi-area interface.

Only point-to-point adjacencies are supported.

IOS

```
interface Ethernet 0/0
 ip ospf 1 area 0
 ip ospf multi-area 1
```

IOS-XR

```
router ospf 1
 area 0
  interface GigabitEthernet0/2/1/2
 area 1
  multi-area-interface GigabitEthernet0/2/1/2
 area 2
  multi-area-interface GigabitEthernet0/2/1/2
```

The multi-area interface inherits the interface characteristics from its primary interface, but some interface characteristics can be configured under the multi-area interface configuration.

It also inherits the BFD characteristics from its primary interface.

OSPF Multiple-instance

Both IOS and IOS-XR allows you to run multiple OSPFv3 instances. Peer routers need to use the same instance-id for ospfv3 communication to happen.

Also, OSPFv3 can support multiple address-families using a different instance per address-family.

Unlike OSPFv3 where the Instance ID can be used for multiple purposes, such as putting the same interface in multiple areas, the OSPFv2 Instance ID is reserved for identifying protocol instances.

Although the relevant RFC defines the mechanism to differentiate packets for different instances sent and received on the same interface, Cisco's current IOS implementation allows you to have multiple OSPFv2 processes (not instances) using only different interfaces. Some of these processes can be VRF ones.

Links

- IETF - RFC 5838
- IETF - RFC 6549

IS-IS

IS-IS (Intermediate System to Intermediate System) is defined in ISO 10589 and in RFC 1142 and RFC 1195.
IS-IS Multi-Instance is defined in RFC 6822.

IS-IS PDU types

- LAN Hello
- Serial (Point-to-Point) Hello
- Link State PDU (LSP)
- Complete Sequence Number PDU (CSNP)
- Partial Sequence Number PDU (PSNP)

IS-IS LSPs are like OSPF LSAs.

CSNPs are generated:

- by the DIS in order for all routers connected to the LAN to synchronize their databases
- by routers on a point-to-point network while setting up their adjacency

PSNPs are generated:

- by routers that are not synchronized with the DIS and need additional LSPs in their database
- by routers on a point-to-point network to acknowledge received LSPs

ISIS Hellos

- Point-to-Point
  - Serial IIH are exchanged
- Multiacces/Broadcast
  - L2 LAN IIH are exchanged

If you get error messages like "%CLNS-3-BADPACKET: ISIS: P2P hello, bad circuit type 0" on point-to-point (Serial/POS) interfaces on GNS3, then just ignore them.

The default hello interval is 10 seconds for non-DIS interfaces, and 3.333 seconds for DIS interfaces.

The hello timers do not need to agree on the neighbors.

On point-to-point links where a single Hello is used, a single hello timer must be used for both L1 and L2 adjacencies.

The fastest neighbor down detection with hello timers tuning is 1 sec, if "`isis hello-interval minimal`" is used. For faster detection use BFD.

All hellos are padded to the full interface MTU by default. You can disable this behavior with "`no isis hello padding`" (although Cisco routers always send the first five hellos padded), if you are having time-sensitive application traffic that travels across low-bandwidth interfaces or you want to minimize interface buffer resources when frequent hellos are configured.

ISIS vs CLNS

- Use "`sh clns`" commands to view
  - neighbors
  - adjacencies
  - hellos
  - PDUs
  - interfaces
  - metrics
- Use "`sh isis`" commands to view
  - neighbors
  - LSPs
  - topologies
  - SPF logs
  - routes

In IOS-XR, all related show commands are found under the "`sh isis`" hierarchy.

If you need to change the MTU for ISIS to work, change the CLNS MTU.

ISIS over Frame-Relay requires CLNS broadcast mapping for the particular DLCI.

NET

NET = AREA-ID + SYSTEM-ID + SEL

```
net = 49.0001.0000.0000.2222.00
```

`AREA-ID = 49.0001` (used for inter-area routing)
`SYSTEM-ID = 0000.0000.2222` (used for intra-area routing)
`SEL = 00`

NET must begin and end with a single octet.

Cisco IS-IS implementation requires a 6-octets System-ID.

In order to ease with area migration, multiple Area-IDs can be configured on each router (**System-ID must be kept the same**).

IOS

```
router isis
 max-area-addresses 5
 net 49.0001.0000.0000.0002.00
 net 49.0002.0000.0000.0002.00
 net 49.0003.0000.0000.0002.00
 net 49.0004.0000.0000.0002.00
 net 49.0005.0000.0000.0002.00
```

In L1 adjacencies, max-area-addresses must match between neighbors.

Dynamic hostname exchange is by default enabled, but you can disable it if required.

IOS

```
router isis
 no hostname dynamic
```

IOS-XR

```
router isis 1
 hostname dynamic disable
```

before...

IOS

```
R2#sh isis neighbors

System Id      Type Interface   IP Address      State Holdtime Circuit Id
R1             L2   Fa1/0       10.1.2.1        UP    25
R2.03
```

after...

IOS

```
R2#sh isis neighbors

System Id      Type Interface   IP Address      State Holdtime Circuit Id
0000.0000.0001 L2   Fa1/0       10.1.2.1        UP    23
0000.0000.0002.03
```

```
R2#sh isis hostname
Level  System ID      Dynamic Hostname  (notag)
 2     0000.0000.0001 R1
     * 0000.0000.0002 R2
```

Configuration

IOS

```
router isis X
 net 49.0001.0000.0000.2222.00
 is-type level-2-only
 passive-interface Loopback0
!
interface X
 ip router isis
```

IOS-XR

```
router isis X
 is-type level-2-only
 net 49.0001.0000.0000.1111.00
 interface Loopback0
  passive
  address-family ipv4 unicast
  !
 interface X
  circuit-type level-2-only
  address-family ipv4 unicast
```

Although it's not required to use an IS-IS process/instance number ("router isis x") in IOS, it's better to use one as a reference in next tasks. IOS-XR requires a process ID.

"isis circuit-type level-2-only" under all interfaces creates an empty local L1 database.
"is-type level-2-only" under the IS-IS process doesn't create a L1 database at all.

"passive-interface X" under the IS-IS process doesn't require "ip router isis" under the interface in IOS, because the interface is automatically advertised. IOS-XR requires the whole config (address-family, circuit-type, passive) under each interface.

"log-adjacency-changes" in IOS or "log adjacency changes" in IOS-XR is not enabled by default under the IS-IS process.

For L1 adjacencies a common area (Area-ID) must be used between neighbors.

It is recommended that routers operating at a single level be configured specifically at that level in order to minimize the number of adjacencies, LSPs, and related SPF/PRC calculations.

Multi-area IS-IS

You can have multiple L1 area processes per router, but only one L2 area process. Interfaces can belong to only one process.

That way you can also have connectivity between different L1 areas that are connected to the same L1/L2 router.

IOS

```
router isis 1
 net 49.0001.0000.0000.0003.00
!
router isis 11
 net 49.0011.0000.0000.0003.00
 is-type level-1
!
interface FastEthernet0/0.1
 ip router isis 1
!
interface FastEthernet0/0.11
 ip router isis 11
```

You might hit a bug in some releases with the following message appearing when trying to activate the L1 process under an interface.

```
%CLNS: Duplicate system ID configured in ip vrf <default> with router
isis null
```

Routing table per router type & route-leaking

- L1 routers have
  - L1 routes for prefixes originating from L1 routers in the same area (even crossing many L2 routers)
  - L1 route for the default route originating from L1/L2 routers
- L1/L2 routers have
  - L1 routes for prefixes originating from "attached" L1 routers
  - L2 routes for all other prefixes
- L2 routers have
  - L2 routes for all prefixes

When using multiple Area-IDs under the same IS-IS process of a L1/L2 router, then L1 routes from one L1 area can pass over to another L1 area.

Use **route-leaking** in L1/L2 routers in order to change the above. Route-leaking can be accomplished either with distribute-list or with a route-map in IOS.

IOS

```
router isis
 redistribute isis ip level-2 into level-1 distribute-list 100
!
 address-family ipv6
  redistribute isis level-2 into level-1 distribute-list PREFIXLIST
 exit-address-family
!
access-list 100 permit ip host 1.1.1.1 host 255.255.255.255
!
ipv6 prefix-list PREFIXLIST permit 2001:1::1/128
```

IPv4 distribute-list used in route-leaking should have the above "awkward" format in order to allow only 1.1.1.1/32 to be leaked.

When doing route-leaking in IOS, you must define the ISIS process/instance right after the "`redistribute isis`" command, although you might not see it in the actual configuration.

IOS-XR

```
router isis X
 address-family ipv4
  propagate level 2 into level 1 route-policy IPv4-RPL
 address-family ipv6
  propagate level 2 into level 1 route-policy IPv6-RPL
```

Leaked L2=>L1 routes appear as "ia" (inter-area) in L1 routers.

L1 routes are always preferred over L2.

Level-2 subdomain must not be partitioned in order for routing to work properly.

Always prefer to have a flat L2 network if possible.

Default admin distance of IS-IS is 115.

When a L1/L2 router advertises a route from L2 to L1, it sets the **U/D bit**, so any other L1/L2 router that receives this L1 LSP with the U/D bit set can ignore it and not advertise it any further.

It's good practice to also **enable wide metrics when doing route-leaking** in order to get "correct" metrics for the leaked routes.

Links

- IETF - RFC 5302

Default Route & ATT bit

An LSP with the **ATT bit** set, creates a default route in the L1 router. More specifically, a L1/L2 router sets the ATT bit in a L1 LSP when it has connectivity to another L1 area too.

The ATT bit can be managed in various ways:

IOS

```
router isis X
 set-attached-bit route-map NODEF-ROUTEMAP
!
route-map NODEF-ROUTEMAP permit 10
 match clns address CLNS-FILTER-SET
!
clns filter-set CLNS-FILTER-SET permit 99.9999
```

Use a non-existent CLNS area if you want to avoid setting the ATT-bit.

IOS-XR

```
CRS(config-isis-af)#attached-bit receive ?
  ignore  Ignore the attached bit in received LSPs
CRS(config-isis-af)#attached-bit send ?
  always-set  Always set the attached bit in our LSP
  never-set   Never set the attached bit our LSP
```

Beware of cases where L1/L2 router connects to multiple L1 areas and somehow loses L2 connectivity. Continuing to advertise a default route to L1 routers might lead to a blackhole.

You can also advertise a default route from a L2 router into a L1 router, by using the following configuration:

IOS

```
router isis X
 default-information originate route-map DEF-ROUTE-ROUTEMAP
!
route-map DEF-ROUTE-ROUTEMAP permit 10
 set level level-1
```

The default route will be advertised to the L1 router, so you end up with 2 default routes (the second one is created automatically from the LSP that has the ATT bit set, but has lower preference).

IOS

```
R2#sh isis rib 0.0.0.0 0.0.0.0

IPv4 local RIB for IS-IS process 1000

IPV4 unicast topology base (TID 0, TOPOID 0x0) =================

0.0.0.0/0
  [115/L1/10] via 10.0.220.20(FastEthernet0/0.220), from 10.0.220.20, tag
0, LSP[12/12]
  [115/L1/10] via 10.0.220.20(FastEthernet0/0.220), from 10.0.220.20, tag
0, LSP[0/28]
```

Remember to filter it from other L1/L2 adjacencies (if such a need arises).

IOS-XR

```
router isis X
 address-family ipv4 unicast
  default-information originate route-policy DEF-ROUTE-RPL
```

DR/DIS

LSPID "router.XX-00" (with next-to-last octet being non-zero) is being sent by a DIS, while LSPID "router.00-00" is being sent by everyone.

IOS

```
R2#sh isis database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime
ATT/P/OL
R2.00-00            * 0x00000015   0xE041        1184              0/0/0
R2.01-00            * 0x00000006   0xFA3B        380               0/0/0
R8.00-00              0x0000000B   0x4F55        1177              0/0/0
R8.09-00              0x00000001   0x5E58        1177              0/0/0
R9.00-00              0x0000000E   0x7C35        806               0/0/0
```

If the Circuit-Id in the "`sh isis neighbors`" output contains a router name (Circuit-ID+System-ID=LAN-ID), then this router is the DIS in the broadcast connection between the local router and the router shown under the System-Id.

IOS

```
R2#sh isis neighbors

Tag null:
System Id      Type Interface   IP Address      State Holdtime Circuit Id
R7             L1   Et0/2       2.2.27.7        UP    8
R7.04
R8             L1   Et0/1       2.2.28.8        UP    7
R8.01
R9             L1   Et0/0       2.2.29.9        UP    9
R9.03
```

Each router generates an LSP for all its interfaces.

Each DIS generates a Pseudo-Node LSP for its attached broadcast interfaces.

DR/DIS election

- highest priority (0-127)
- highest mac address

Setting priority to 0 doesn't disable DIS election; use point-to-point to disable it.

There can be separate DRs for L1 and L2 adjacencies.

There is no backup DR. If the primary DR fails, a new DR is elected.

DR preemption is enabled by default.

"`sh isis rib`" in IOS will accept only a network prefix as input, but it will return all routes for the classful network. Always use a network mask to get the output for a specific route.

The "*" in front of a prefix in the "`sh isis ipv6 rib`" output means that the specific route will be installed in the router RIB also (not including directly connected networks).

Overload-bit

The overload bit is included in an LSP of the router and if it is set, it notifies routers in the area that **the router is not available for transit traffic**. It may be configured and cleared independently for IPv4 and IPv6 topologies.

IOS

```
router isis
 set-overload-bit
```

IOS-XR

```
router isis 100
 set-overload-bit
```

When used in combination with "wait-for-bgp", then if BGP sessions come up and BGP keepalives are not received from **all the BGP neighbors**, IS-IS will disable the overload bit after 10 minutes by default.

The **IS-IS overload bit avoidance** when activated, allows TE LSPs to continue working, although the router in that path has its overload bit set.

IOS

```
R1(config)#mpls traffic-eng path-selection overload allow ?
  head    Allow overloaded head node in TE CSPF
  middle  Allow overloaded middle node in TE CSPF
  tail    Allow overloaded tail node in TE CSPF
```

IOS-XR

```
mpls traffic-eng path-selection ignore overload

CRS(config)#mpls traffic-eng path-selection ignore overload ?
  head  Ignore overload node during CSPF for role head
  mid   Ignore overload node during CSPF for role mid
  tail  Ignore overload node during CSPF for role tail
  <cr>
```

Metrics

Default metric is 10 for each interface, 0 for passive interfaces.

IOS

```
R2(config-if)#isis metric ?
  <1-16777214>  Default metric
  maximum       Maximum metric. All routers will exclude this link from
their
                SPF
```

IOS-XR

```
GSR(config-isis-if-af)#metric ?
  <1-16777214>  Default metric: <1-63> for narrow, <1-16777214> for wide
  maximum       Maximum wide metric. All routers will exclude this link
from the
ir SPF
```

Wide metrics are used in:

- MPLS-TE
- prefix tags
- multi-topology
- need for metric > 63

IOS

```
router isis
 metric-style wide
```

IOS-XR

```
router isis 100
 address-family ipv4 unicast
  metric-style wide
 !
 address-family ipv6 unicast
  metric-style wide
```

If two connected routers have different metric styles, an adjacency will be formed between them and LSPs will be exchanged, but routes will not be installed. You can use "`debug isis rib local`" to verify if routes are being generated for use by the RIB.

The maximum metric that can be assigned to an IS-IS route is 1023 (without wide metrics enabled).

IS-IS Authentication

Authentication can be enabled:

- per domain (old style)
    - "`domain-password`" under IS-IS process
    - applies to LSPs by default
    - for CSNPs, PSNPs extra command is required
    - for Level-2 only
    - clear text (type-1)
    - adjacency formed but no L2 LSPs exchanged if wrong authentication
- per area (old style)
    - "`area-password`" under IS-IS process
    - applies to LSPs by default
    - for CSNPs, PSNPs extra command is required
    - for Level-1 only
    - clear text (type-1)
    - adjacency formed but no L1 LSPs exchanged if wrong authentication
- per interface (old style)
    - "`isis-password`" under interface
    - applies to Hellos by default

  - for Level-1/Level-2
  - clear text (type-1)
  - no adjacency formed if wrong authentication
- **per interface (new style)**
  - "isis authentication" under interface (IOS)
  - "hello-password" under interface under IS-IS process (IOS-XR)
  - applies to Hellos by default
  - for Level-1/Level-2
  - enhanced clear text (type-1) or MD5 (type-54)
  - no adjacency formed if wrong authentication
- **per instance (new style)**
  - "authentication" under IS-IS process (IOS)
  - "lsp-password" under IS-IS process (IOS-XR)
  - applies to LSPs, CSNPs, PSNPs by default
  - for Level-1/Level-2
  - enhanced clear text (type-1) or MD5 (type-54)
  - adjacency formed but no LSPs exchanged if wrong authentication

Prefer to use the new way of authentication, when not told otherwise.

You can use "text" in new-style authentication in order to be compatible with old-style authentication. For old-style area authentication you have to use "level-1" in new-style, while for old-style domain authentication you have to use "level-2" in new-style.

On point-to-point links where a single Hello is used, a common password must be used for both L1 and L2 adjacencies.

IOS

```
interface FastEthernet0/0
 isis authentication mode md5
 isis authentication key-chain KEYCHAIN
!
key chain KEYCHAIN
 key 1
  key-string TESTPASS
```

IOS-XR

```
router isis 26
 interface X
  hello-password hmac-md5 TESTPASS
```

In IOS-XR, a key-chain can also be used instead of hmac-md5.

The only way to verify authentication in current releases is by using debug commands ("debug isis update-packets" and "debug isis authentication information").

IS-IS Topologies

- IOS
  - default is single-topology
  - configure "`multi-topology`" under ipv6 address-family to change
- IOS-XR
  - default is multi-topology
  - configure "`single-topology`" under ipv6 address-family to change
  -

In order to allow adjacency to be formed in mismatched address-families in single-topology, the "`no adjacency-check`" command must be configured under the IPv6 address family. The same command is enabled by default in multi-topology.

Only one IPv6 process is allowed in IOS.

IS-IS Single-Topology requirements

- Both IPv4 IS-IS and IPv6 IS-IS routing protocols must share a common network topology
- Any interface configured for IPv4 IS-IS must also be configured for IPv6 IS-IS, and vice versa
- All routers in the IS-IS area (for Level 1 routing) or the domain (for Level 2 routing) must support an identical set of address families (IPv4 only, IPv6 only, or both IPv4 and IPv6) on all interfaces
- Wide metrics are not necessary in single-topology

Links

- IETF - RFC 5120

ISIS as PE-CE

PE

IOS

```
router isis X
 vrf VPN
 net 49.0001.0000.0000.0001.00
!
interface FastEthernet0/0
 vrf forwarding VPN
 ip router isis X
```

IOS-XR

not supported

CE

IOS

```
router isis X
 net 49.0001.0000.0000.0002.00
```

IOS-XR

```
router isis X
 net 49.0001.0000.0000.0002.00
```

IS-IS for IPv6 is not supported as a PE-CE protocol in IOS.

IOS-XR doesn't support IS-IS as a PE-CE protocol at the role of PE.

Multi-Instance

IOS-XR

```
router isis 1
 is-type level-2
 net 47.0002.0000.0000.0008.00
 address-family ipv4 unicast
  metric-style wide
 !
 address-family ipv6 unicast
  metric-style wide
 !
 interface GigabitEthernet0/2/2/0
  address-family ipv4 unicast
  address-family ipv6 unicast
!
router isis 2
 is-type level-2
 net 49.0002.0000.0000.0008.00
 address-family ipv4 unicast
  metric-style wide
 !
 address-family ipv6 unicast
  metric-style wide
 !
 interface GigabitEthernet0/2/1/0
  address-family ipv4 unicast
  address-family ipv6 unicast
```

You can configure up to five IS-IS instances (processes).

MPLS can run on multiple IS-IS processes as long as the processes run on different sets of interfaces. Each interface may be associated with only a single IS-IS instance.

Because RIB treats each of the IS-IS instances as equal routing clients, you must be careful when redistributing routes between IS-IS instances.

The RIB does not know to prefer Level 1 routes over Level 2 routes from different instances, so if you are running Level 1 and Level 2 instances, you must enforce the preference by configuring different administrative distances for these two instances.

multi-instance vs multi-area

- multi-instance
  - supported in IOS-XR
  - multiple L2 areas
  - multiple L1 areas
  - redistribution allowed between different processes
  - multiple IPv6 processes allowed
  - use when: run multiple IS-IS processes
- multi-area
  - supported in IOS
  - only one L2 area
  - multiple L1 areas
  - redistribution not allowed between different processes
  - only one IPv6 process allowed
  - use when: connect multiple L1 areas on the same router

Fast Convergence

- BFD
- tunning of hellos
- ip event dampening
- point-to-point adjacencies
- tuning of SPF/PRC/LSP timers
- tag specific prefixes and give them high priority

Timers

- per process/instance
  - **LSP refresh interval**
    - seconds the router will wait before refreshing (re-creating and re-flooding) its own LSPs
    - recommended: 65535 sec

    - **Max LSP lifetime**
        - the lifetime in the LSP header (in order to age out old LSPs)
        - recommended: 65535 sec
    - **PRC interval** (exponential backoff)
        - seconds between two consecutive PRCs (triggered when changes that do not affect the topology, such as advertised external prefixes or metric changes, are detected)
        - recommended: 5, 1, 20
    - **LSP generation interval** (exponential backoff)
        - seconds between creating new versions of a given LSP on a per-node basis
        - recommended: 5, 1, 20
    - **SPF interval** (exponential backoff)
        - seconds between two consecutive SPF calculations
        - recommended: 5, 1, 20
- per interface
    - **LSP interval**
        - milliseconds between the transmission of LSPs
    - **LSP retransmit interval**
        - seconds between the retransmission of an LSP on point-to-point links
    - **LSP retransmit throttle interval**
        - milliseconds between all the retransmitted LSPs on point-to-point links

The value set for the lsp-refresh-interval should be less than the value of the max-lsp-lifetime command. Usually the software will automatically reduce the LSP refresh interval to prevent the LSPs from timing out.

iSFP can be used to limit the SPF recalculations to specific portions of the topology.

Advertise minimum prefixes

IOS

```
router isis
 advertise passive-only
 set-overload-bit suppress interlevel external
!
interface X
 no isis advertise prefix
```

IOS-XR

```
router isis 100
 no set-overload-bit advertise external interlevel
 address-family ipv4 unicast
  advertise passive-only
 !
 interface X
  suppressed
```

Summarization

IOS

```
router isis
 summary-address 11.11.11.0 255.255.255.0
 address-family ipv6
  summary-prefix 11:11:11::/64
```

IOS-XR

```
router isis 1
 address-family ipv4 unicast
  summary-prefix 11.11.11.0/24
 !
 address-family ipv6 unicast
  summary-prefix 11:11:11::/64
```

You can also define the level into which you want to advertise the summary.

BGP

BGP (Border Gateway Protocol) is defined in RFC 4271.

Uses TCP port 179.

The router with the highest router-id is used as the TCP client.

Best Path Selection

#	Attribute	Rule	Default	Notes	Affects Traffic	Applies to route-map
1	Prefix Length	Longest match		Always checked	inbound & outbound	
2	Cost Community	Lowest Cost Community Number Lowest Cost Community ID	2147483647	Checked if "set extcommunity cost pre-bestpath" is configured. Skipped if "bgp bestpath cost-community ignore" is configured.		
3	WEIGHT	Highest Weight	32768	Local to the router. Local originated prefixes have weight 32768 by default. Only for Cisco; not recommended for general use.		
4	LOCAL PREFERENCE	Highest Local Preference	100	Used for separating customer/peering/transit traffic.	outbound	inbound
5		Prefer local-sourced routes		Everything announced through network/aggregate commands or redistribution is considered as local-sourced.		

6	AS-PATH	Shortest as-path		Ignored if "`bgp bestpath as-path ignore`" is configured.	inbound	outbound
7	ORIGIN	Lowest Origin Type (IGP<EGP<Incomplete)				
8	MED	Lowest Multi-Exit Discriminator (MED)		1st AS must be the same, unless "`bgp always-compare-med`" is configured.	inbound	outbound
9		Prefer eBGP over iBGP				
10		Lowest IGP metric to the BGP next hop				
11	Cost Community	Lowest Cost Community Number Lowest Cost Community ID	2147483647	Checked if "`set extcommunity cost igp`" is configured. Skipped if "`bgp bestpath cost-community ignore`" is configured.		
12		Check for BGP Multipath				
13		If both paths are external, prefer the path received first				
14		Lowest BGP router-id				
15		If the originator-id or the router-id is the same for multiple paths, prefer path with minimum cluster list length		Only for route reflectors		
16		Prefer the path with the lowest neighbor address				

<u>MED</u>

- `bgp deterministic-med`
    - compare MED when choosing routes advertised by different neighbors in the same autonomous system
    - routes from the same autonomous system are grouped together and the best entries of each group are compared
- `bgp always-compare-med`

  - compare MED when choosing routes advertised by different neighbors in different autonomous systems

"`bgp deterministic-med`" and "`bgp always-compare-med`" are recommended in order to alway have a standard best path selection algorithm.

In order to avoid mis-interpreting a missing MED (zero vs infinity), when setting MED on a peering, it's best to set it also on all other peerings. Otherwise the command "`bgp bestpath med missing-as-worst`" can be used.

Common Comparisons

- Prefix Length
- Highest Local Preference
- Shortest as-path
- Lowest Multi-Exit Discriminator (MED)
- Prefer eBGP over iBGP
- Lowest IGP metric to the BGP next hop
- Lowest BGP router-id

origin

The origin attribute indicates how BGP learned about a particular route.

- i (IGP)
  - interior to the originating AS (i.e. when the network configuration command is used to inject the route into BGP)
- e (EGP)
  - learned via EGP (rarely seen)
- ? (incomplete)
  - unknown or learned via some other way (i.e. redistributed into BGP, or from eBGP)

Address Families

AFIs
- 1 (IPv4)
- 2 (IPv6)
- 25 (L2VPN)

SAFIs
- 1 (Unicast)
- 2 (Multicast)
- 4 (NLRI with MPLS labels)
- 65 (VPLS)
- 128 (VPN with MPLS labels (VRFs))

Configuration

IOS

```
router bgp 2
 no synchronization
 bgp log-neighbor-changes
 network 5.5.5.5 mask 255.255.255.255
 neighbor 20.4.5.4 remote-as 1
 no auto-summary
```

IOS-XR

```
router bgp 1
 bgp log neighbor changes detail
 address-family ipv4 unicast
  network 4.4.4.4/32
 !
 neighbor 20.4.5.5
  remote-as 2
  address-family ipv4 unicast
```

In IOS-XR, if there is no loopback configured with an ipv4 address, the BGP session won't come up, until you explicitly configure the **bgp router-id**.

BGP is designed to refuse a session with itself because of the router-id check. You can use a per-vrf assignment of BGP router-id in order to have a VRF-to-VRF peering on the same router.

In IOS-XR, every eBGP session requires an explicit route-policy in order to allow incoming/outgoing updates. It's good practice to create one named PASS-RPL with default action "pass" and use it when first activating each eBGP session. Afterwards you can create the required route-policy and use that instead.

When told to advertise a prefix into BGP, prefer to use the "`network`" statement, unless told to do otherwise. Also prefer to do the "network" advertisements to another AS on routers running eBGP to that AS.

You can use the "`network x.x.x.x backdoor`" command in order to change the admin distance of an eBGP route (default 20) to that of iBGP (200), so that the equivalent IGP route can be preferred.

Route Aggregation

- redistribution of static or IGP
- aggregate-address
  - summary-only
  - suppress-map
  - unsuppress-map (per neighbor)
  - advertise-map
- inject-map

A more specific prefix must exist in the BGP table before doing aggregation.

Communities

- Standard Communities (32-bit)
  - Used for well-known communities and for specific communities of type $ASN:$TAG in BGP
  - `send-community` (IOS)
  - `send-community-ebgp` (IOS-XR)
- Extended Communities (64-bit) are defined in
  - Used in MPLS VPNs for RT and SOO
  - `send-community extended` (IOS)
  - `send-extended-community-ebgp` (IOS-XR)

Communities are configured through community lists.

When **regular expressions** are required, **expanded** (standard or extended) **community lists** must be used.

Configuration

- Standard
  - `ip community-list 1 permit 100:10`
  - `ip community-list standard X-COMMLIST permit 100:10`
- Expanded Standard
  - `ip community-list 100 permit 100:*`
  - `ip community-list expanded X-COMMLIST permit 100:*`
- Extended
  - `ip extcommunity-list 1 permit rt 200:20`
  - `ip extcommunity-list standard X-COMMLIST permit rt 200:20`
- Expanded Extended
  - `ip extcommunity-list 100 permit rt 200:*`
  - `ip extcommunity-list expanded X-COMMLIST permit rt 200:*`

In IOS, all communities are not sent by default to iBGP or eBGP sessions.

In IOS-XR, all communities are sent by default on iBGP sessions, but not on eBGP sessions.

Well-known communities

- internet
- no-export (don't advertise to eBGP neighbor)
- local-as (don't advertise to other confederation sub-AS)
- no-advertise(don't advertise to any neighbor)

Delete communities

IOS

```
route-map DELCOMM1-ROUTEMAP permit 10
 set comm-list 1 delete
!
route-map DELCOMM2-ROUTEMAP permit 10
 set community none
!
route-map DELCOM3-ROUTEMAP permit 10
 set extcomm-list 1 delete
```

IOS-XR

```
route-policy DELCOM1-RPL
  delete community in (*:*)
end-policy
!
route-policy DELCOM3-RPL
  delete extcommunity rt all
end-policy
```

Use the "`additive`" keyword to add communities to existing ones.

Links

- IETF - RFC 1997
- IETF - RFC 4360
- IANA Extended Communities

Synchronization

A BGP router with synchronization enabled does not install iBGP learned routes into its routing table and propagate them to an eBGP peer, if it is not able to **validate those routes in its IGP first**. It's used to ensure that there are **no black holes inside the AS** caused by intermediate routers that do not run BGP.

It's disabled by default ("`no synchronization`"), because nowadays most networks run iBGP or MPLS.

Route Reflectors

Route Reflectors modify iBGP split-horizon rules.

Routes learned on a RR from a RR-Client are propagated to other RR-Clients and Non-Clients
Routes learned on a RR from a Non-Client are propagated only to RR-Clients

RRs can be assigned per address-family.

RRs do not modify the next-hop of advertised routes by default.

RRs can be in the forwarding path or not.

Use "`no bgp client-to-client reflection`" on RRs, when their clients are also fully meshed.

An RR reflecting a route received from an RR-Client adds the following attributes:

- **Originator ID**
  - the Router ID of the originator of the route
  - if the update comes back to the originator (so the local Router-ID is the same as the Originator-ID), the update is ignored
- **Cluster List**
  - a list of Cluster IDs that an update has passed through
  - when an RR reflects a route from a client to a non-client, the local Cluster ID is appended to the Cluster List
  - if the update comes back to the RR (so the local Cluster-ID is contained in the prefix Cluster List) the update is ignored

Originator and Cluster List are used to prevent loops in RR environments.

By default Cluster-ID = RR Router-ID. In case of two RRs, two different Cluster-IDs will be used. This increases memory utilization, because the same route is stored multiple times, each one with a different Cluster-ID.

You can use a common Cluster-ID in redundant RRs (in order to decrease memory utilisation, although rarely needed), only when you're sure that connectivity for RR clients won't break if the RR client looses one of its RR connections.

IOS

```
router bgp 100
 neighbor 2.2.2.2 remote-as 100
 neighbor 2.2.2.2 update-source Loopback 0
 neighbor 2.2.2.2 route-reflector-client
```

IOS-XR

```
router bgp 100
 neighbor 2.2.2.2
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
   route-reflector-client
```

Links

- IETF - RFC 4456

Confederations

The AS is split into smaller autonomous systems in order to reduce the number of iBGP sessions.

It's common practice to use the private AS range (64512 – 65535) to denote a sub-autonomous system.

These internal ASNs are hidden and only a single external ASN is announced to eBGP neighbors.

BGP confederations modify iBGP as-path processing

When sending:

- updates to iBGP neighbors
  - as-path is not changed
- updates to intra-confederation eBGP neighbors
  - the intra-confederation ASN is prepended to the as-path
- updates to eBGP neighbors
  - the intra-confederation ASNs are removed and the external ASN is prepended to the as-path

Intra-confederation eBGP session is:

- like eBGP session when establishing the session (ebgp-multihop)
- like iBGP session when sending routing updates (local pref, next-hop, etc.)

IOS

```
router bgp INTERNAL-ASN-100
 bgp confederation identifier EXTERNAL-ASN-1
 bgp confederation peers INTERNAL-ASN-200 INTERNAL-ASN-300
 neighbor 2.2.2.2 remote-as INTERNAL-ASN-200
 neighbor 3.3.3.3 remote-as INTERNAL-ASN-300
 neighbor 9.9.9.9 remote-as EXTERNAL-ASN-9
```

IOS-XR

```
router bgp INTERNAL-ASN-100
 bgp confederation peers
  INTERNAL-ASN-200
  INTERNAL-ASN-300
 !
 bgp confederation identifier EXTERNAL-ASN-1
 !
 neighbor 2.2.2.2
  remote-as INTERNAL-ASN-200
 !
 neighbor 3.3.3.3
  remote-as INTERNAL-ASN-300
 !
 neighbor 9.9.9.9
  remote-as EXTERNAL-ASN-9
```

EXTERNAL-ASNs define the ASNs used for eBGP sessions between different ASNs.

INTERNAL-ASNs define the ASNs used for eBGP sessions between different sub-ASNs of the same ASN.

Example

IOS

```
router bgp 65100
 bgp confederation identifier 1
 bgp confederation peers 65200 65300
 neighbor 2.2.2.2 remote-as 65200
 neighbor 3.3.3.3 remote-as 65300
 neighbor 9.9.9.9 remote-as 9
```

IOS-XR

```
router bgp 65100
 bgp confederation peers
  65200
  65300
 !
 bgp confederation identifier 1
 !
 neighbor 2.2.2.2
  remote-as 65200
 !
 neighbor 3.3.3.3
  remote-as 65300
 !
 neighbor 9.9.9.9
  remote-as 9
```

Links

- IETF - RFC 5065

Next-Hop

- advertisement to eBGP peer
  - next-hop changes to self
  - use "next-hop-unchanged" to not change
- advertisement to iBGP peer

  - next-hop doesn't change
  - use "next-hop-self" to change

You can't use the next-hop-self for setting the next-hop in reflected iBGP routes. Instead use an outbound route map.

In IOS-XR, you can use "ibgp policy out enforce-modifications" in combination with an outbound route-map in order to force modification of the routes attributes (including next-hop) when sent to an iBGP neighbor.

---

keepalive & holdtime

Neighbor holdtime timers are negotiated while initially setting the BGP session and the smaller one gets used by both neighbors. Keepalive timers are then based on that holdtime value. It's not recommended to have less than 3 secs as a holdtime.

The fastest convergence on a BGP session that can be achieved by changing the keepalive/holdtime timers is 3 sec.

In order to protect the control-plane, you can put a limit on the lowest holdtime number accepted by using the "min-holdtime" command. If the neighbor doesn't comply, then the BGP session is rejected.

Mass Neighbor Configuration

In order to minimize neighbor configuration regarding the BGP session parameters you can use the following:

peer groups (IOS)

```
router bgp 100
 neighbor PEER-GROUP peer-group
 neighbor PEER-GROUP remote-as 100
 neighbor PEER-GROUP update-source Loopback0
!
 neighbor 1.1.1.1 peer-group PEER-GROUP
!
 address-family vpnv4
  neighbor PEER-GROUP send-community extended
  neighbor 1.1.1.1 activate
```

neighbor groups (IOS-XR)

```
router bgp 100
 neighbor-group NEI-GROUP
  remote-as 100
  update-source Loopback0
```

```
!
 neighbor 1.1.1.1
  use neighbor-group NEI-GROUP
```

peer session templates (IOS)

```
router bgp 100
 template peer-session PEER-TEMPLATE
  remote-as 100
  update-source Loopback0
!
 neighbor 1.1.1.1 inherit peer-session PEER-TEMPLATE
```

---

eBGP Peerings & TTL

IOS

```
router bgp 100
 neighbor 2.2.2.2 ttl-security hops X
 neighbor 3.3.3.3 ebgp-multihop Y
```

IOS-XR

```
router bgp 100
 neighbor 2.2.2.2
  ttl-security
 neighbor 3.3.3.3
  ebgp-multihop Y
```

IOS

```
R1#sh bgp nei | i TTL|Session
  Session: 2.2.2.2
Mininum incoming TTL 255-X, Outgoing TTL 255
  Session: 3.3.3.3
Mininum incoming TTL 0, Outgoing TTL Y
```

eBGP Multihop

It allows a neighbor connection between two external peers that do not have direct connection. You should also configure an IGP or static routing to allow the neighbors without direct connection to reach each other.

TTL Security Check

It's a lightweight security mechanism to protect eBGP neighbor sessions from CPU utilization-based attacks (DoS attacks that flood the network with IP packets that contain forged source IP addresses).

IOS

When configured for an eBGP neighbor, the router accepts only IP packets with a TTL count that is greater or equal to maximum TTL value (255) minus the hop count that is configured locally for the relevant eBGP session. **If the TTL value in the IP packet is less than the maximum TTL value (255) minus the hops configured value, the incoming packet is silently discarded.**

Supports both directly connected neighbor sessions and multihop eBGP neighbor sessions

IOS-XR

When configured for a **directly adjacent eBGP neighbor**, the router accepts only IP packets with a TTL count that is equal to the maximum TTL value (255). **If the TTL value in the IP pakcet is less than the maximum TTL value (255), the incoming packet is silently discarded.**

TTL values according to BGP setup:

- R1 config: `neighbor R2`
  - R1 sends packets to R2 with TTL=1
- R1 config: `neighbor R2 ttl-security hops X`
  - R1 sends packets to R2 with TTL=255
- R1 config: `neighbor R2 ebgp-multihop X`
  - R1 sends packets to R2 with TTL=X

ebgp-mutlihop combined with ttl-security on two eBGP routers

R1: `ebgp-multihop X (<255)`
R2: `ttl-security hops Y (<254)`

- R1 sends packets to R2 with TTL=X
  - R2 doesn't reply back
  - R2 accepts packets with TTL < 255-Y
- R2 sends packets to R1 with TTL=255
  - R1 replies back
  - R1 accepts packets with any TTL

General Rule

If (X - ActualHops >= 255 - Y) then the eBGP session can be established.

Interesting Cases

R1: `ebgp-multihop X`
R2: `ttl-security hops 254`

- R1 sends packets to R2 with TTL=X
  - R2 replies back
- R2 sends packets to R1 with TTL=255

  - R1 replies back

R1: `ebgp-multihop 255`
R2: `ttl-security hops Y`

- R1 sends packets to R2 with TTL=255
  - R2 replies back
  - R2 accepts packets with TTL < 255-Y
- R2 sends packets to R1 with TTL=255
  - R1 replies back
  - R2 accepts packets with any TTL

If ebgp-multihop is set to 255 or ttl-security is set to 254 (aka when at least one of these parameters is set to its max), then the eBGP session can be established, as long as their packets can reach each other.

R1: `ttl-security hops X`
R2: `ttl-security hops Y`

- R1 sends packets to R2 with TTL=255
  - R2 replies back
  - R2 accepts packets with TTL < 255-Y
- R2 sends packets to R1 with TTL=255
  - R1 replies back
  - R1 accepts packets with TTL < 255-X

If both routers use ttl-security, then the eBGP session can be established regardless of the hop values used, as long as their packets can reach each other.

R1: `ebgp-multihop X`
R2: `ebgp-multihop Y`

- R1 sends packets to R2 with TTL=X
  - R2 replies back
  - R2 accepts packets with any TTL
- R2 sends packets to R1 with TTL=Y
  - R1 replies back
  - R1 accepts packets with any TTL

If both routers use ebgp-multihop, then the eBGP session can be established regardless of the hop values used, as long as their packets can reach each other.

If loopback interfaces are used to connect single-hop eBGP peers, you can configure the "`neighbor disable-connected-check`" command before you can establish the eBGP peering session.

PMTUD

IOS

```
R2#sh bgp vpnv4 unicast all nei 19.19.19.19 | i tcp|segment
  Transport(tcp) path-mtu-discovery is enabled
Datagrams (max data segment is 1432 bytes):
```

If you have BGP PMTUD enabled (by default in most releases), BGP packets will be sent with DF bit set.

You can disable BGP PMTUD (either for all neighbors or for a specific neighbor) with the following commands.

IOS

```
router bgp 100
 no bgp transport path-mtu-discovery
 neighbor 19.19.19.19 transport path-mtu-discovery disable
```

If global command "`ip tcp path-mtu-discovery`" is disabled (default) and BGP PMTUD is disabled too, then the default MSS (536) is used for BGP neighbors.

If "`ip tcp path-mtu-discovery`" is enabled but BGP PMTUD is disabled, then the maximum MSS is used for BGP neighbors.

You can use "`ip tcp mss X`" to change the global TCP MSS.

IOS-XR

```
tcp path-mtu-discovery
```

Links

- IETF - RFC 1191

Advanced BGP

BGP (Border Gateway Protocol) is defined in RFC 4271.
MP-BGP (Multi-Protocol BGP) is defined in RFC 4760.
Labeled BGP (BGP+Label) is defined in RFC 3107.

enforce-first-as

When enabled, updates received from an eBGP peer that does not list its ASN at the beginning of the as-path in the incoming update are denied (in order to prevent spoofing).

It's enabled by default.

IOS

```
router bgp 100
 no bgp enforce-first-as
```

IOS-XR

```
router bgp 65000
 bgp enforce-first-as disable
```

local-as & dual-as

When local-as is enabled for a neighbor, it allows a router to appear to be a member of a second ASN, in addition to its real ASN.

This feature can only be used for true eBGP peers (i.e. members of different confederation sub-ASs are not supported).

R4 (IOS)

```
router bgp 1
 network 4.4.4.4 mask 255.255.255.255
 neighbor 20.4.5.5 remote-as 2
 neighbor 20.4.5.5 local-as 11
```

R5 (IOS)

```
router bgp 2
 network 5.5.5.5 mask 255.255.255.255
 neighbor 20.4.5.4 remote-as 11
```

By default, the new local-as is prepended in incoming and outgoing updates.

IOS

```
R4#sh bgp ipv4 unicast
BGP table version is 3, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       0.0.0.0                  0         32768 i
*> 5.5.5.5/32       20.4.5.5                 0             0 11 2 i

R5#sh bgp ipv4 unicast
BGP table version is 3, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       20.4.5.4                 0             0 11 1 i
*> 5.5.5.5/32       0.0.0.0                  0         32768 i
```

Use the "no-prepend" option to avoid prepending the new local-as in the incoming updates.

R4 (IOS)

```
router bgp 1
 network 4.4.4.4 mask 255.255.255.255
 neighbor 20.4.5.5 remote-as 2
 neighbor 20.4.5.5 local-as 11 no-prepend
```

IOS

```
R4#sh bgp ipv4 unicast
BGP table version is 5, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       0.0.0.0                  0         32768 i
*> 5.5.5.5/32       20.4.5.5                 0             0 2 i

R5#sh bgp ipv4 unicast
BGP table version is 5, local router ID is 5.5.5.5
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       20.4.5.4                 0             0 11 1 i
*> 5.5.5.5/32       0.0.0.0                  0         32768 i
```

Use the "no-prepend replace-as" option to avoid prepending the real ASN in the outgoing updates.

R4 (IOS)

```
router bgp 1
 network 4.4.4.4 mask 255.255.255.255
 neighbor 20.4.5.5 remote-as 2
 neighbor 20.4.5.5 local-as 11 no-prepend replace-as
```

IOS

```
R4#sh bgp ipv4 unicast
BGP table version is 7, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       0.0.0.0                  0         32768 i
*> 5.5.5.5/32       20.4.5.5                 0             0 2 i

R5#sh bgp ipv4 unicast
BGP table version is 7, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       20.4.5.4                 0             0 11 i
*> 5.5.5.5/32       0.0.0.0                  0         32768 i
```

Use the "no-prepend replace-as dual-as" option to avoid prepending the new local-as in the incoming updates and the real ASN in the outgoing updates and at the same time **allow eBGP connections with both the real ASN and the new local-as**.

R4 (IOS)

```
router bgp 1
 network 4.4.4.4 mask 255.255.255.255
 neighbor 20.4.5.5 remote-as 2
 neighbor 20.4.5.5 local-as 11 no-prepend replace-as dual-as
```

R5 (IOS)

```
router bgp 2
 network 5.5.5.5 mask 255.255.255.255
 neighbor 20.4.5.4 remote-as 11
```

IOS

```
R4#sh bgp ipv4 unicast
BGP table version is 9, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       0.0.0.0                  0         32768 i
*> 5.5.5.5/32       20.4.5.5                 0             0 2 i
```

```
R5#sh bgp ipv4 unicast
BGP table version is 9, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       20.4.5.4                 0             0 11 i
*> 5.5.5.5/32       0.0.0.0                  0         32768 i
```

or

R5 (IOS)

```
router bgp 2
 network 5.5.5.5 mask 255.255.255.255
 neighbor 20.4.5.4 remote-as 1
```

```
R4#sh bgp ipv4 unicast
BGP table version is 11, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
```

```
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       0.0.0.0                  0         32768 i
*> 5.5.5.5/32       20.4.5.5                 0             0 2 i

R5#sh bgp ipv4 unicast
BGP table version is 11, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 4.4.4.4/32       20.4.5.4                 0             0 1 i
*> 5.5.5.5/32       0.0.0.0                  0         32768 i
```

PE-CE Routing

In order to allow VPN sites with the same ASN talk to each other, you can use one of the following:

- "`neighbor PE allowas-in`" in the CE
  - CE accepts its own ASN
- "`neighbor CE as-override`" in the PE
  - PE replaces the common CE ASN with its own

eBGP sessions in IOS-XR require an in/out PASS routing policy under the appropriate address-family. Alternatively in some cases you can use "`bgp unsafe-ebgp-policy`" in order to bypass this.

IOS-XR

```
vrf VPN
 address-family ipv4 unicast
  import route-target
   100:1
  export route-target
   100:1
!
router bgp 100
 address-family ipv4 unicast
 vrf VPN
  rd 100:1
  bgp unsafe-ebgp-policy
  address-family ipv4 unicast
  neighbor 2.2.2.2
   remote-as 200
   address-family ipv4 unicast
    as-override
```

Labeled BGP

It's a BGP capability (negotiated between neighbors during session setup) that allows you to **exchange labels together with IPv4/IPv6 unicast prefixes**. It's used in Inter-AS, CsC, 6PE scenarios, and when LDP+IGP or RSVP-TE are not available for label distribution.

Configuration

IOS

```
router bgp 100
 address-family ipv4
  neighbor 1.1.1.1 send-label
```

IOS-XR

```
router bgp 100
 address-family ipv4 unicast
  allocate-label all
 neighbor 1.1.1.1
  address-family ipv4 labeled-unicast
```

You can also filter the prefixes for which to allocate labels.

Verification

```
R2#sh bgp ipv4 unicast neighbors 1.1.1.1 | b capabilities
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Four-octets ASN Capability: advertised and received
    Address family IPv4 Unicast: advertised and received
    ipv4 MPLS Label capability: advertised and received
    Multisession Capability: advertised and received
```

In IOS-XR, when you activate a new ipv4/ipv6-labeled session for an existing ipv4/ipv6 neighbor, you need to re-apply all settings (i.e. route-policy, send-community) from the ipv4 session to the ipv4-labeled session.

L3VPN

"`send-community extended`" is usually automatically enabled when activating a neighbor under the BGP VPNv4 address-family. Since RT is an extended community, without this command VPNv4 routes won't be advertised in BGP.

In order to see the **VPN label** to be used by the PEs, you just need to check the relevant BGP route.

```
R2#sh bgp vpnv4 unicast all 6.6.6.6/32
...
    5.5.5.5 (metric 4) from 5.5.5.5 (5.5.5.5)
...
      mpls labels in/out nolabel/28
```

In order to see the **IGP/Transport label** to be used by the PEs and Ps, you just need to find the label for the route's next-hop. **Remember to add the "`detail`" keyword in order to see the whole label stack (due to possible route recursion).**

```
R2#sh mpls forwarding-table 5.5.5.5
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label       or Tunnel Id     Switched      interface
27         26          5.5.5.5/32       0             Fa0/0.23   20.2.3.3
```

In order to see the **whole label stack** (which includes both the VPN and the IGP label), you can check the relevant CEF entry (inside the VRF) on the PEs.

```
R2#sh ip cef vrf VPN 6.6.6.6 det
6.6.6.6/32, epoch 0, flags rib defined all labels
  recursive via 5.5.5.5 label 28
    nexthop 20.2.3.3 FastEthernet0/0.23 label 26
```

If you want to follow a Intra-AS L3VPN path (assuming control-plane has been setup correctly), then you can execute the following algorithm:

- first router (start PE)
    - Find the VPN label for the prefix
    - Find the Transport label(s) for the prefix's next-hop
- n router
    - Follow the Transport top label swaps until there is a "Pop Label" for next router
- n+1 router
    - Find the local VPN label for the prefix
        - If VPN label is "no label", then
            - router is the end PE
            - VPN is locally attached
        - If VPN label is other, then
            - ?
        - If VPN label doesn't exist, then
            - ?

If the route is learned from IGP, the Transport label must be allocated through LDP/RSVP-TE.
If the route is learned from BGP, the Transport label must be allocated through BGP.

Dynamic L3VPN with mGRE Tunnels

If MPLS is not available in a network, you can use GRE (or other types of encapsulation) to "automatically" build dynamic tunnels in order to provide L3VPN services.

The BGP nexthop is used for tunnel endpoint discovery, but instead of adding a transport label, VPN traffic is encapsulated into GRE (having as source a local interface and as destination the neighbor PE).

The L3VPN BGP configuration (regarding VRFs and VPNv4) remains the same as in MPLS L3VPN.

Configuration Steps

- create a new VRF for the mGRE tunnels
- create a mGRE tunnel (with no destination) and assign the above VRF to it
- create a default static route that forwards the above VRF traffic into the mGRE tunnel
- activate the above VRF under BGP
- apply an inbound route-map that changes the next-hop to the above VRF to all the PE sessions
-

The same tunnels can be used for all L3VPNs between the same PEs.

IOS

```
vrf definition L3VPN-VRF
 rd 1:99
!
interface Tunnel 1
 tunnel mode gre multipoint l3vpn
 tunnel source loopback0
 ip vrf forwarding L3VPN-VRF
 ip address 99.99.99.1 255.255.255.255
 tunnel key 99
!
ip route vrf L3VPN-VRF 0.0.0.0 0.0.0.0 Tunnel1
!
router bgp 1
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
!
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 2.2.2.2 route-map L3VPN-ROUTEMAP in
 exit-address-family
!
 address-family ipv4 vrf L3VPN-VRF
 exit-address-family
!
route-map L3VPN-ROUTEMAP permit 10
 set ip next-hop in-vrf L3VPN-VRF
```

In latest releases you can also use multipoint L2TPv3 tunnels instead of the default mGRE ones.

You can also define l3vpn encapsulation profiles for fully automatic tunnel provisioning.

IOS

```
l3vpn encapsulation ip L3VPN-PROFILE
 transport source loopback 0
 protocol gre key 99
!
router bgp 1
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
!
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 2.2.2.2 route-map L3VPN-ROUTEMAP in
 exit-address-family
!
route-map L3VPN-ROUTEMAP permit 10
 set ip next-hop encapsulate L3VPN-PROFILE
```

Link Bandwidth

It is used with BGP multipath to configure **load balancing over links with unequal bandwidth**.

When enabled, routes learned from directly connected external neighbors are propagated through the iBGP network with the bandwidth of the source external link stored in an extended community.

The **link bandwidth extended community** attribute is used as a traffic sharing value relative to other paths while forwarding traffic.

Two or more paths are designated as equal for load balancing if weight, local-preference, as-path length, MED and IGP costs are the same.

BGP can originate the link bandwidth community only for directly connected links to eBGP neighbors.

Configuration Steps

- "dmzlink-bw" must be enabled on all BGP routers that need to process the link bandwidth community
- "dmzlink-bw" must be enabled on all eBGP neighborships from where the bandwidth will be acquired
- "send-community extended" must be enabled on all iBGP peerings where the link bandwidth community must be propagated to
- multipath must be enabled where more than one path is expected

R2 (IOS)

```
router bgp 1
 bgp dmzlink-bw
 neighbor 3.3.3.3 remote-as 1
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 4.4.4.4 remote-as 1
 neighbor 4.4.4.4 update-source Loopback0
 maximum-paths ibgp 4
```

R3 (IOS)

```
router bgp 1
 bgp dmzlink-bw
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 2.2.2.2 next-hop-self
 neighbor 2.2.2.2 send-community extended
 neighbor 4.4.4.4 remote-as 1
 neighbor 4.4.4.4 update-source Loopback0
 neighbor 4.4.4.4 next-hop-self
 neighbor 4.4.4.4 send-community extended
 neighbor 20.3.6.6 remote-as 2
 neighbor 20.3.6.6 dmzlink-bw
 maximum-paths 4
 maximum-paths ibgp 4
!
interface FastEthernet0/0.36
 bandwidth 36000
```

R4 (IOS)

```
router bgp 1
 bgp dmzlink-bw
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 2.2.2.2 next-hop-self
 neighbor 2.2.2.2 send-community extended
 neighbor 3.3.3.3 remote-as 1
 neighbor 3.3.3.3 update-source Loopback0
 neighbor 3.3.3.3 next-hop-self
 neighbor 3.3.3.3 send-community extended
 neighbor 20.4.5.5 remote-as 2
 neighbor 20.4.5.5 dmzlink-bw
 neighbor 20.4.6.6 remote-as 2
 neighbor 20.4.6.6 dmzlink-bw
 maximum-paths 4
 maximum-paths ibgp 4
!
interface FastEthernet0/0.45
 bandwidth 45000
!
interface FastEthernet0/0.46
 bandwidth 46000
```

IOS

```
R2#sh bgp
BGP table version is 5, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*mi19.19.19.19/32   4.4.4.4                  2    100      0 2 i
*>i                 3.3.3.3                  2    100      0 2 i

R2#sh bgp ipv4 unicast 19.19.19.19/32
BGP routing table entry for 19.19.19.19/32, version 5
Paths: (2 available, best #2, table default)
Multipath: iBGP
  Not advertised to any peer
  2
    4.4.4.4 (metric 5) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 2, localpref 100, valid, internal, multipath
      DMZ-Link Bw 11375 kbytes
  2
    3.3.3.3 (metric 5) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 2, localpref 100, valid, internal, multipath,
best
      DMZ-Link Bw 4500 kbytes
```

Although BGP multipath is enabled, the BGP selection algorithm still chooses one path as the best (based on the standard BGP selection criteria), but both paths are tagged with the "multipath" keyword and appear in the routing table for forwarding.

```
R2#sh ip route 19.19.19.19
Routing entry for 19.19.19.19/32
  Known via "bgp 1", distance 200, metric 2
  Tag 2, type internal
  Last update from 3.3.3.3 00:04:36 ago
  Routing Descriptor Blocks:
  * 4.4.4.4, from 4.4.4.4, 00:04:36 ago
      Route metric is 2, traffic share count is 5
      AS Hops 1
      Route tag 2
      MPLS label: none
    3.3.3.3, from 3.3.3.3, 00:04:36 ago
      Route metric is 2, traffic share count is 2
      AS Hops 1
      Route tag 2
      MPLS label: none
```

Divide the bandwidth entry (Kbps) by 8 to find out the DMZ-Link Bw (KBps) in the"`sh bgp`" output.

IOS-XR

```
router bgp 2
 address-family ipv4 unicast
  maximum-paths ibgp 4
  maximum-paths ebgp 4
 !
 neighbor 6.6.6.6
  dmz-link-bandwidth
```

The above (old-style) configuration is not recommended. In later IOS-XR releases (>4.3.2) you can set the bandwidth extcommunity in a route-policy towards the iBGP neighbor in order to achieve the same thing.

Links

- IETF - draft-ietf-idr-link-bandwidth

RT Constrain (RTC)

The default behavior is for the PEs to filter out the unwanted RTs, after they receive the prefixes from the RR. After enabling this feature on the PE and the RR, the PE informs the RR what RTs it actually needs and the RR sends only those.

This feature causes two exchanges to happen:

- The PE sends an RT Constraint (RTC) NLRI to the RR
- The RR installs an outbound route filter

The rtfilter address-family must be activated on both the RR and the PE.

IOS

```
router bgp 100
 neighbor 1.1.1.1 remote-as 100
 neighbor 1.1.1.1 update-source Loopback0
 !
 address-family vpnv4
  neighbor 1.1.1.1 activate
  neighbor 1.1.1.1 send-community extended
 exit-address-family
 !
 address-family rtfilter unicast
  neighbor 1.1.1.1 activate
  neighbor 1.1.1.1 send-community extended
 exit-address-family
```

IOS-XR

```
router bgp 100
 address-family vpnv4 unicast
 !
 address-family ipv4 rt-filter
 !
 neighbor 1.1.1.1
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
  !
  address-family ipv4 rt-filter
```

It requires IOS-XR > 4.3 or IOS > 15.1.

Links

- IETF - RFC 4684

Fast Convergence

- Different RD per PE
- BGP Multipath
- BGP Best-external
- BGP PIC
- Two RRs (one for primary, one for secondary)

Multipath

It allows installation of multiple BGP paths to the same destination into the IP routing table. These paths are installed in the table together with the best path for load sharing. **BGP Multipath does not affect best-path selection.** For example, a router still designates one of the paths as the best path, according to the algorithm, and advertises this best path to its neighbors.

- eBGP multipath
  - `maximum-paths x` (IOS)
  - `maximum-paths ebgp x` (IOS-XR)
- iBGP multipath
  - `maximum-paths ibgp x` (IOS, IOS-XR)
- eiBGP multipath (under ipv4 vrf address-family)
  - `maximum-paths eibgp x` (IOS, IOS-XR)

In IOS-XR, you can also use the "`selective`" keyword in order to restrict multipath to specific neighbors (the ones with "`multipath`" configured).

CEF load-sharing might need to be tuned also.

"`bgp bestpath as-path multipath-relax`" can be used to skip checking the as-path contents and check only its length.

Best-External Path

When configured, enables the advertisement of the best-external path to iBGP/RR peers, if the locally selected best-path is from an internal peer. That way routers internal to the AS have knowledge of more exit paths from the AS.

Usually it's configured on the backup router.

IOS

```
router bgp 100
 address-family vpnv4
  bgp advertise-best-external
```

IOS-XR

```
router bgp 100
 address-family ipv4 unicast
  advertise best-external
```

PIC (Prefix Independent Convergence)
When configured, provides a capability to install a backup path into the forwarding table to provide prefix independent convergence in case of PE-CE link failure

Core/Edge

IOS
```
router bgp 100
 address-family vpnv4
  bgp additional-paths install
  bgp recursion host
```

IOS-XR (3.9)
```
router bgp 100
 address-family vpnv4 unicast
  additional-paths install backup
```

For faster convergence you might need to remove the command "bgp recursion host".

Links
- IETF - draft-ietf-idr-best-external

QPPB (QoS Policy Propagation via BGP)

It allows you to match BGP routes based on attributes (i.e. community, as-path), mark these with ip prec or qos-group (or other attributes depending on software version) and then mark appropriately the relevant source/destination packets matching the above routes. Further actions (i.e. policing, queuing) can be performed on the marked packets afterwards.

IOS
```
ip community-list 1 permit 100:1
!
ip as-path access-list 1 permit _200$
!
route-map QPPB-ROUTEMAP permit 10
 match community 1
 set ip precedence 2
!
route-map QPPB-ROUTEMAP permit 20
 match as-path 1
 set ip precedence 5
!
router bgp 100
```

```
 table-map QPPB-ROUTEMAP
!
interface FastEthernet0/0
 bgp-policy source ip-prec-map
```

IOS-XR

```
route-policy QPPB-ROUTEPOLICY
  if community matches-any (100:1) then
    set qos-group 2
  endif
  if as-path originates-from '200'  then
    set qos-group 5
  endif
end-policy
!
router bgp 100
 address-family ipv4 unicast
  table-policy QPPB-ROUTEPOLICY
!
interface GigabitEthernet0/0/0/0
 ipv4 bgp policy propagation input qos-group source
```

IOS-XR has various limitations depending on hw used.

RTBH (Remotely Triggered Black Hole) routing/filtering

It allows you to quickly "block" various attacks on your edge routers, by advertising a null route from a single router to all edge routers.

Configuration Steps

- configure null static route with dummy next-hop on your edge routers
- configure route-map that matches a tag and sets a dummy next-hop (plus whatever else) on your rtbh router
- configure redistribution of static routes into BGP using the above route-map on your rtbh router
- in case of attack, configure a null static route with the appropriate tag for the destination on the rtbh router

i.e. for destination-based RTBH:

edge (IOS)

```
ip route 192.168.1.1 255.255.255.255 Null0
```

rtbh router (IOS)

```
router bgp 100
 redistribute static route-map RTBH-ROUTEMAP
```

```
!
route-map RTBH-ROUTEMAP
 match tag 99
 set ip next-hop 192.168.1.1
 set community no-export no-advertise additive
```

When attack to 10.10.10.10 happens:

rtbh router (IOS)

```
ip route 10.10.10.10.10 255.255.255.255 Null0 tag 99
```

It is assumed that the rtbh router has BGP connectivity with all edge routers (either directly, or through RRs).

If you combine loose uRPF + RTBH, you can use it for blocking source ips too.

VRF

VRF Basic Configuration

IOS

```
ip vrf VPN-A
 rd 100:1
 route-target export 100:1
 route-target import 100:1
!
vrf definition VPN-B
 rd 100:2
 address-family ipv4
  route-target export 100:2
  route-target import 100:2
 exit-address-family
 address-family ipv6
  route-target export 100:2
  route-target import 100:2
 exit-address-family
```

IOS-XR

```
vrf VPN-C
 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
!
router bgp 100
 vrf VPN-C
  rd 100:3
```

Prefer to use the "`vrf definition`" command to configure VRFs. Always include an address-family, most probably the ipv4 one.

You can have different import/export RTs per address family. If you have common ones, then you can define them directly under the vrf definition.

IOS-XR requires the VRF RD config under the BGP process.

You can use the "rd auto" command in IOS -XR in order to automatically create unique RDs per VRF.

IOS-XR

```
router bgp 100
 vrf VPN
  rd auto
```

If you don't define any export RTs for a VRF on the local PE, then the prefixes will by default get dropped when they are transferred to the remote PE.

You can view all routing tables (global and VRF ones) by using the command "sh ip route vrf *".

On a router that acts as a default gateway, the following can be configured if the next-hop is in the global routing table and you want to have the static route inside the VRF but pointing to the global routing table.

IOS

```
ip route vrf VPN 0.0.0.0 0.0.0.0 1.1.1.1 global
```

You can change the default label allocation (if you want to decrease label usage) for all VRFs or a specific VRF using the following command:

IOS

```
R1(config)#mpls label mode vrf VPN protocol bgp-vpnv4 ?
  per-prefix     Per prefix label (default)
  per-vrf        Per VRF label for entire VRF
  vrf-conn-aggr  Per VRF label for connected and BGP aggregates in VRF
```

Export RT per Prefix

You can use an export map in order to set different export RTs per prefix.

IOS

```
vrf definition VPN-A
 rd 100:1
 !
 address-family ipv4
  export map R1-MAP
 exit-address-family
!
route-map R1-MAP permit 10
 match ip address R1-ACL
 set extcommunity rt 2.2.2.2:11 2.2.2.2:111
!
```

```
ip access-list standard R1-ACL
 permit 1.1.1.1
!
```

IOS-XR

```
vrf VPN-A
 address-family ipv4 unicast
  export route-policy R1-RPOLICY
!
route-policy R1-RPOLICY
  if destination in (1.1.1.1/32) then
    set extcommunity rt (2.2.2.2:11, 2.2.2.2:111)
  endif
end-policy
```

In the IOS-XR route-policy you must use destination in order to match the required prefix. Also you can use parenthesis whenever you need to group parameters, like the multiple RTs.

If you initially have no export RTs and later decide to add some through an export map, then you must reset the VPNv4 BGP sessions in order to have the BGP routes get the new RTs immediately.

Import global routes into VRF

You have the option of importing various global routes into a specific VRF, while at the same time limiting the number of them.

IOS

```
vrf definition VPN_A
 rd 100:1
 route-target export 100:1
 route-target import 100:1
 !
 address-family ipv4
  import ipv4 unicast 5 map GLOBAL-TABLE-ROUTEMAP
 exit-address-family
!
route-map GLOBAL-TABLE-ROUTEMAP permit 10
 match ip address prefix-list PREFIX-LIST
!
ip prefix-list PREFIX-LIST seq 5 permit 4.4.4.4/32
 match ip address prefix-list PREFIX-LIST
```

IOS

```
R2#sh bgp vpnv4 unicast all
BGP table version is 8, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
```

```
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
Import Map: GLOBAL-TABLE-ROUTEMAP, Address-Family: IPv4 Unicast, Pfx
Count/Limit: 1/5
*> 1.1.1.1/32       10.1.2.1                 0         32768 i
*> 4.4.4.4/32       20.2.4.4                10         32768 i
*> 10.1.2.0/24      0.0.0.0                  0         32768 i
*>i10.19.20.0/24    19.19.19.19              0    100      0 i
*>i20.20.20.20/32   19.19.19.19              0    100      0 i

R2#sh bgp vpnv4 unicast all 4.4.4.4/32
BGP routing table entry for 100:1:4.4.4.4/32, version 8
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  Local, imported path from 4.4.4.4/32
    20.2.4.4 from 0.0.0.0 (2.2.2.2)
      Origin IGP, metric 10, localpref 100, weight 32768, valid,
external, no-import, best
```

The global route to be imported must exist in the global BGP table (existence in RIB doesn't matter).

No RTs or other attributes will be assigned to the imported prefixes. Use the route-map set commands to configure those.

In latest IOS releases (>15.x) you have to option of doing the opposite too, export prefixes from a VRF into the global BGP table.

The commands "`import/export map`" (which are different from the above) are used to filter the VRF <=> MP-BGP prefixes, while the commands "`import/export ipv4 unicast`" are used to leak routes between the VRF and the global BGP table.

Inbound VPN Prefix Filtering due to RTs

In VPNv4/v6 BGP setups, **all BGP VPN prefixes are checked against the local RT import policies and if there is no match found, then the prefix is discarded** (in order to keep the BGP table small).

Route-Reflectors have this filter disabled by default, because they need all the prefixes to accommodate all possible PEs. In order to disable it manually on other routers, you can use the following commands.

IOS

```
router bgp 100
 no bgp default route-target filter
```

IOS-XR

```
router bgp 100
 address-family vpnv4 unicast
  retain route-target all
```

IOS-XR supports also the selective filtering of RTs by using a policy that matches specific RTs.

IOS-XR

```
router bgp 100
 address-family vpnv4 unicast
  retain route-target route-policy RT-POLICY
!
route-policy RT-POLICY
  if extcommunity rt matches-any (100:1, 100:2) then
    pass
  else
    drop
  endif
```

Multi-VRF (VRF-Lite)

It allows a logical separation of a CE router into multiple VRFs, without the need for MPLS/MP-BGP. Using MPLS Multi-VRF you can extend the LSPs to the CE and all routing domains that the CE supports.

Usually, a router with Multi-VRF is shared by several customers and each customer has their own routing table.

Characteristics

- one or more VRFs configured and assigned to interfaces
- no MPLS configured
- no BGP VPNv4 configured
- no RTs under "`vrf definition`" required on IOS
- no RDs under "`bgp vrf`" required on IOS-XR

Configuration Steps

- Configure VRFs on the PE and the CE
- Configure the routing protocol for each VRF on the CE towards each customer
- Configure BGP or IGP as PE-CE protocol for each VRF
- Configure labeling either with BGP+Label or IGP+LDP (if required)

Special care must be taken in case of using OSPF as PE-CE protocol, because due to setting the DN bit by default on specific routes advertised from PE to CE, these routes wouldn't be installed in the Multi-VRF CE's routing table.

Always prefer to use BGP as PE-CE, due to its simpler configuration and better filtering.

Traffic classification into a VRF

- One VRF per interface
  - Classic method (IOS, IOS-XR)
- Multiple VRFs per interface
  - Multi-VRF Selection Using Policy-Based Routing (IOS)
    - match an ip access-list
  - VRF Selection Based on Source IP Address (IOS)
    - match the source ip address
  - VRF-Autoclassify (IOS)
    - match the directly connected prefix
  - ACL Based Forwarding with VRF Next-Hop (IOS-XR)
    - match an ip access-list

Classic Method

IOS

```
interface X
 ip vrf forwarding VPN-A
!
interface X
 vrf forwarding VPN-B
```

IOS-XR

```
interface X
 vrf VPN-C
```

Multi-VRF Selection Using Policy-Based Routing

It allows a specific interface on a PE router to route packets to different VPNs based on **various match criteria defined in an ip access list or on packet length**.

IOS

```
interface X
 ip vrf receive VPN-A
 ip vrf receive VPN-B
 ip vrf receive VPN-C
 ip address 1.1.1.1 255.255.255.0
 ip policy route-map VRF-SELECTION-PBR

ip access-list standard VPN-A-ACL
 permit 1.1.2.2
ip access-list standard VPN-B-ACL
 permit 1.1.3.3
ip access-list standard VPN-C-ACL
 permit 1.1.4.4
!
route-map VRF-SELECTION-PBR permit 10
 match ip address VPN-A-ACL
 set vrf VPN-A
!
route-map VRF-SELECTION-PBR permit 20
 match ip address VPN-B-ACL
 set vrf VPN-B
!
route-map VRF-SELECTION-PBR permit 30
 match ip address VPN-C-ACL
 set vrf VPN-C
```

Limitations

- multicast is not usually supported by PBR

VRF Selection Based on Source IP Address

It allows a specific interface on a PE router to route packets to different VPNs based upon the **source IP address of the packets**. It's supported only on very specific hardware.

IOS

```
vrf selection source 1.1.1.0 255.255.255.0 vrf VPN-A
vrf selection source 2.2.2.0 255.255.255.0 vrf VPN-B
!
int X
 ip vrf select source
 ip vrf receive VPN-A
 ip vrf receive VPN-B
```

Limitations

- supported by limited hardware
- performance can degrade with longer subnet masks

If it's not supported by the router, you'll get the following output:

```
R1(config)#vrf selection source 1.1.1.0 255.255.255.0 vrf VPN-A
% VRF Select: failed to add config
```

VRF-Autoclassify

It allows a specific interface on a PE router to route packets to different VPNs based on the **ip addresses assigned to directly connected hosts**.

At the same time it also enables certain types of Policy Based Routing (PBR) to be created dynamically without configuring all the related route maps and access lists. More specifically it enables the dynamic configuration of policies that are required for the mapping of packets to the VRFs, **depending on whether the source address of the packet belongs to those connected routes.**

IOS

```
interface X
 ip address 1.1.1.1 255.255.255.0 secondary vrf VPN-A
 ip address 1.1.2.1 255.255.255.0 secondary vrf VPN-B
 ip address 2.2.2.2 255.255.255.0
 ip vrf autoclassify source
!
interface Y
 ip vrf forwarding VPN-A
 ip address 10.10.10.1 255.255.255.0
!
interface Z
 ip vrf forwarding VPN-B
 ip address 20.20.20.1 255.255.255.0
```

Limitations

- any directly connected hosts must not run routing protocols
- the router that is enabled with the VRF-Autoclassify feature must not run routing protocols
- applies only to unicast packets

ACL Based Forwarding (ABF) with VRF Next-Hop

It allows packet arriving on ingress VRF to get routed on an egress interface which is in a different VRF based on **various match criteria defined in an ip access list**.

So you actually specify a nexthop VRF instead of a specific IP address for the nexthop. When this config is enabled, packet destination lookup will be performed in the ABF Nexthop VRF.

IOS-XR

```
interface X
 ipv4 address 10.10.10.10 255.255.255.0
 vrf VPN-A
 ipv4 access-group ABF-ACL ingress
!
ipv4 access-list ABF-ACL
 permit ipv4 1.1.2.2 0.0.0.0 any nexthop1 vrf VPN-A
 permit ipv4 1.1.3.3 0.0.0.0 any nexthop1 vrf VPN-B
 permit ipv4 1.1.4.4 0.0.0.0 any nexthop1 vrf VPN-C
```

Limitations

- applies only to ASR9k and CRS running IOS-XR

MPLS/LDP

LDP (Label Distribution Protocol) is defined in RFC 5036.
MPLS (Multi-Protocol Label Switching) architecture is defined in RFC 3031.
MPLS Label Stack Encoding is defined in RFC 3032.

LDP messages

- LDP Discovery (to directly connected neighbors)
  - Multicast UDP to 244.0.0.2:646
- targeted LDP Discovery (to non-directly connected neighbors)
  - Unicast UDP to x.x.x.x:646
- LDP Session/Advertisement/Notification (to all)
  - Unicast TCP to x.x.x.x:646
  -

A working IGP between neighbors is a requirement for all the above, besides LDP Discovery.

Label retention/distribution

- Label retention
  - **liberal**
  - conservative
- Label distribution
  - **downstream**
    - **unsolicited**
    - on-demand

Liberal, downstream, unsolicited is the most common case.

General

Although in latest software releases LDP is the default label protocol, it's a good practice to **always enable it with "`mpls label protocol ldp`"**. The same applies with the "**`mpls ldp router-id`**", which should in most cases be loopback0.

Use "`sh mpls ldp bindings`" to check the **LIB** (labels for all IGP database prefixes)
Use "`sh mpls forwarding`" to check the **LFIB** (labels for RIB installed prefixes)

Outgoing Label under "show mpls forwarding-table":

- X Label
  - Local device is an LSR
- Pop Label
  - Local device is an LSR and also the PHP
- No Label

  - Local device is a LER

You need to configure "`mpls ldp explicit-null`" if you want to keep the EXP QoS till the end PE. Default is implicit-null due to PHP.

"`debug mpls packet`" includes the label stack {Label EXP TTL} information

```
Fa0/0.1: rx: Len 122 Stack {29 0 253} - ipv4 data
Fa0/0.2: tx: Len 122 Stack {30 0 252} - ipv4 data
```

Debugging should be the last thing you should do in case of a problem in production networks. So learn not to depend on it.

Label Allocation Methods

- an IGP/Transport Label is allocated through
  - LDP (+IGP)
  - RSVP (MPLS TE)
  - Labeled BGP
- a VPN Label (L3VPN) is allocated through
  - MP-BGP (VPNv4/v6)
- a PW Label (L2VPN) is allocated through
  - Targeted LDP
- an IPv6 Label (6PE) is allocated through
  - Labeled BGP

Targeted LDP Sessions

You can create targeted LDP sessions (assuming ip connectivity exists) using the following methods:

IOS

Static LDP neighbors on both routers

R1

```
mpls ldp neighbor R2 targeted
```

R2

```
mpls ldp neighbor R1 targeted
```

Static LDP neighbor on one router and accept targeted hellos on the other

R1

```
mpls ldp neighbor R2 targeted
```

R2

```
mpls ldp discovery targeted-hello accept
```

MPLS/LDP under a TE tunnel interface on one router and static LDP neighbor on the other

R1

```
interface Tunnel0
 tunnel destination R2
 mpls ip
```

R2

```
mpls ldp neighbor R1 targeted
```

MPLS/LDP under a TE tunnel interface on one router and accept targeted hellos on the other

R1

```
interface Tunnel0
 tunnel destination R2
 mpls ip
```

R2

```
mpls ldp discovery targeted-hello accept
```

Something similar applies to IOS-XR too.

IOS-XR

R1

```
mpls ldp
 neighbor R2 targeted
```

R2

```
mpls ldp
 discovery targeted-hello accept
```

In case of RSVP in the core and LDP in the access, you can have tLDP sessions over RSVP, where end-to-end LSPs will have 1 label (LDP) in the access and 2 labels (RSVP/LDP) in the core.

You can also use RSVP solely for (one-hop) link protection, having tLDP on top of it.

Simple VPN with iBGP & LDP

The rule for LSP usage in BGP is that **when an LSP is available for the BGP next-hop of a route, that LSP can be used to forward traffic to that route destination.**

Assuming a network of R1-R2-R3-R4-R5-R6, where R2,R3,R4,R5 run LDP+IGP in the core and R2,R5 run iBGP between them, then R1 and R6 can communicate between each other as long as their networks as advertised to R2,R5 (i.e with eBGP) and R2,R5 are using their loopbacks as next-hops in their iBGP. The intermediate routers R3,R4 just do mpls switching based on the R2,R5 loopback labels.

R2,R5 have BGP-generated labels for the R1,R6 prefixes which according to BGP have R2,R5 as next-hops. **These labels (which are the same as the ones used for the BGP next-hop) are shown only if you exclusively define the network in "`sh mpls forwarding-table`" or use "`sh ip cef`".**

IOS

```
R2#sh mpls forwarding-table | i 6.6.6.6
R2# -no entry shown-

R2#sh mpls forwarding-table 6.6.6.6
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
None       22         6.6.6.6/32       0             Fa0/0.23   20.2.3.3

R2#sh mpls forwarding-table | i 5.5.5.5
21         22         5.5.5.5/32       0             Fa0/0.23   20.2.3.3

R2#sh ip route 6.6.6.6
Routing entry for 6.6.6.6/32
  Known via "bgp 100", distance 200, metric 0
  Tag 20, type internal
  Last update from 5.5.5.5 00:35:32 ago
  Routing Descriptor Blocks:
  * 5.5.5.5, from 5.5.5.5, 00:35:32 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
      Route tag 20
      MPLS label: none

R2#sh ip cef 6.6.6.6 det
6.6.6.6/32, epoch 0, flags rib only nolabel, rib defined all labels
  recursive via 5.5.5.5
    nexthop 20.2.3.3 FastEthernet0/0.23 label 22
```

R3,R4 have LDP-generated labels for the R2,R5 next-hops

Static Labels

After you configure the label range, you need to remove the mpls ldp config from the IGP process or from the interfaces in order to use the label range. If you just clear the LDP neighbors, then the old labels remain.

Check "Inter-AS MPLS L3VPN" for more examples.

LDP Auto-configuration

- Supported in OSPF and IS-IS
- Not supported on MPLS-TE Tunnels

IOS

```
router ospf/isis X
 mpls ldp autoconfig
!
interface X
  no mpls ldp igp autoconfig
```

IOS-XR

```
mpls ldp
!
router ospf/isis X
 mpls ldp auto-config
 !
 interface X
  igp auto-config disable
```

Use "`sh mpls interfaces detail`" or "`sh mpls ldp discovery detail`" to find out how LDP was activated on an interface.

IOS-XR requires the explicit activation of MPLS LDP prior to LDP autoconfiguration.

LDP Authentication

Authentication is applicable only to the LDP TCP session.

After setting the LDP password, the LDP session might need to be cleared manually to have the password enabled.

When setting "`mpls ldp password required`", all LDP sessions are cleared automatically.

Password can be configured:

- per neighbor
  - "`mpls ldp neighbor x.x.x.x password`" (IOS, IOS-XR)
- per group of neighbors
  - "`mpls ldp password option X for Y-ACL`" (IOS)
- as a default password for all neighbors
  - "`mpls ldp password fallback`" (IOS)
  - "`mpls ldp neighbor password`" (IOS-XR)

Label Filtering

The LDP default behavior is to allocate local labels for all non-BGP prefixes, which includes IGP learned prefixes and connected interfaces with LDP on.

- **Local Label Allocation Filtering**
  - controls the allocation of local labels
  - uses prefix-lists for filtering
  - use "`allocate global prefix-list`" under "mpls ldp label" config (IOS)
  - use "`mpls ldp label allocate`" under global config (IOS-XR)
  - use "sh mpls ldp bindings local" to verify
- **Inbound Label Binding Filtering**
  - controls label bindings that a router accepts from a specific neighbor
  - uses access-lists for filtering
  - use "`mpls ldp neighbor x.x.x.x labels accept`" under global config (IOS)
  - use "`mpls ldp label accept`" under global config (IOS-XR)
  - use "sh mpls ldp bindings neighbor" to verify
- **Outbound Label Binding Filtering**
  - controls label bindings that a router sends to a specific neighbor
  - uses access-lists for filtering
  - use "`mpls ldp advertise-labels`" under global config (IOS) - "`no mpls advertise`" is needed first
  - use "`mpls ldp label advertise`" under global config (IOS-XR)
  - use "sh mpls ldp bindings neighbor" to verify

LDP does not apply the configured local label filter to redistributed BGP routes in the global table for which BGP allocates the local label, but LDP does the advertisements (i.e. Inter-AS Option C). LDP neither forwards these entries, nor releases the local labels allocated by BGP.

Common use of label filtering is to allocate labels only for PE loopback addresses.

LDP Session Protection

When enabled, a **new targeted LDP session is created between the neighbors, in order to keep their LDP session active over any backup path, after the direct/primary link fails**. When the primary/direct link is restored, label bindings do not need to be re-exchanged.

2 implementation choices:

- both neighbors must be configured for session protection
- one router must be configured for session protection and the other router must simply respond to targeted hellos

IOS

```
mpls ldp session protection
mpls ldp session protection for LDP-NEI-ACL duration X
```

IOS-XR

```
mpls ldp
 session protection
 session protection for LDP-NEI-ACL duration X
```

You can enable it for all LDP neighbors or for specific ones using an ACL.

IOS

```
R2#sh mpls ldp neighbor detail | i Protection|duration
        LDP Session Protection enabled, state: Ready
            duration: 86400 seconds
        LDP Session Protection enabled, state: Incomplete
            duration: 86400 seconds
        LDP Session Protection enabled, state: Protecting
            duration: 86400 seconds

R2#sh mpls ldp neighbor 3.3.3.3 detail
...
        LDP discovery sources:
          FastEthernet0/0.23; Src IP addr: 20.2.3.3
            holdtime: 15000 ms, hello interval: 5000 ms
          Targeted Hello 2.2.2.2 -> 3.3.3.3, active, passive;
            holdtime: infinite, hello interval: 10000 ms
```

Successful recovery

```
%LDP-5-SP: 3.3.3.3:0: session hold up initiated
%LDP-5-SP: 3.3.3.3:0: session recovery succeeded
```

Failed recovery

```
%LDP-5-SP: 4.4.4.4:0: session hold up initiated
%LDP-5-SP: 4.4.4.4:0: session recovery failed
```

```
%LDP-5-NBRCHG: LDP Neighbor 4.4.4.4:0 (1) is DOWN (Session Protection
disabled targeted session)
```

LDP IGP Synchronization

When enabled, links where LDP adjacencies are not established, will have their IGP metric increased to the max by the local IGP process.

Generally **when an IGP adjacency is established on a link with LDP-IGP Sync on, but LDP-IGP Sync is not yet achieved (or is lost), the IGP advertises the max-metric on that link**. That way the link won't be preferred for passing traffic and black-holing will be prevented.

IOS

```
router ospf/isis X
 mpls ldp sync
!
mpls ldp igp sync holddown x
!
interface X
 no mpls ldp igp sync
 mpls ldp igp sync delay x
```

IOS-XR

```
router ospf/isis X
 mpls ldp sync
!
mpls ldp
 igp sync delay x
 interface X
  igp sync delay x
```

IOS

```
R2#sh mpls ldp igp sync
    FastEthernet0/0.23:
        LDP configured; LDP-IGP Synchronization enabled.
        Sync status: sync achieved; peer reachable.
        Sync delay time: 0 seconds (0 seconds left)
        IGP holddown time: infinite.
        Peer LDP Ident: 3.3.3.3:0
        IGP enabled: OSPF 1

R2#sh mpls ldp igp sync
    FastEthernet0/0.23:
        LDP configured; LDP-IGP Synchronization enabled.
        Sync status: sync not achieved; peer reachable.
        Sync delay time: 0 seconds (0 seconds left)
        IGP holddown time: infinite.
        Peer LDP Ident: 3.3.3.3:0
        IGP enabled: OSPF 1
```

IOS-XR

```
GSR#sh mpls ldp igp sync

GigabitEthernet0/1/0/1.619:
  Sync status: Ready
  Peers:
    6.6.6.6:0
```

Targeted LDP sessions (i.e. AToM) are not supported, which is expected because these are already tLDP sessions that can use IGP for rerouting.

In IS-IS the maximum wide metric -1 (0XFFFFFE) is used with MPLS LDP IGP synchronization.

Links

- IETF - RFC 5443
- IETF - RFC 6138

TTL Propagation

Default behavior is to copy the TTL from the IP header to the MPLS header (topmost label).

2 extra options are available:

- do not copy the TTL for forwarded packets
  - "`no mpls ip propagate-ttl forwarded`" (IOS)
  - "`mpls ip-ttl-propagate disable forwarded`" (IOS-XR)
- do not copy the TTL for locally generated packets
  - "`no mpls ip propagate-ttl local`" (IOS)
  - "`mpls ip-ttl-propagate disable local`" (IOS-XR)

If the TTL is not copied for forwarded packets, then a traceroute from a local CE to a remote CE, will include the local PE, the remote PE and the remote CE (all the intermediate P routers won't be shown). You can use this in order to hide the MPLS hops from the customer.

You only need to disable the TTL propagation on the PEs, since the P (LSR) routers do not see the original IP packet, so no TTL propagation takes place there.

Traceroute in MPLS L3VPNs works a little bit differently than in normal IP networks, because **when the traceroute packet reaches the MPLS core (P routers), the local generated ttl-exceeded response packet must first reach the PE at the other side of the VPN before it's returned back to the traceroute source**.

i.e. in a traceroute from CE1 to CE2 (CE1-PE1-P-PE2-CE2), the following happens

- CE1 sends a ICMP packet with TTL=1
    - Source=CE1
    - Destination=CE2
    - TTL=1
- PE1 receives the ICMP packet
- PE1 sends an ICMP ttl-exceeded response back to CE1
    - Source=PE1
    - Destination=CE1
- CE1 receives the ICMP response
- CE2 sends an ICMP packet with TTL=2
    - Source = CE1
    - Destination=CE2
    - TTL=2
- PE1 receives the ICMP packet
- PE1 forwards the ICMP packet to the next P
    - Source=CE1
    - Destination=CE2
    - TTL=1
- P receives the ICMP packet
- P creates an ICMP ttl-exceeded response and sends it to PE2 using the original label stack
    - Source=P
    - Destination=CE1
    - TTL=default
- PE2 receives the ICMP response and forwards it to CE1 which is the actual destination
    - Source=P
    - Destination=CE1
- CE1 receives the ICMP response
- CE1 sends a ICMP packet with TTL=3
    - and so on...

MPLS MTU

Every label adds 4 bytes to the frame size.

Common label stacks

- L3VPN
    - LDP label + VC label
- L2VPN/VPLS
    - LDP label + VC label
- MPLS-TE

    - TE label + VC label
    - TE label + LDP label + VC label
- MPLS-TE/FRR
    - FRR label + TE label + VC label
    - FRR label + TE label + LDP label + VC label
- AToM & TE/FRR & CsC
    - FRR label + TE label + LDP label + VPN label + VC label

Common transport header sizes (in bytes):

- Ethernet port:14
- Ethernet VLAN: 14 + 4 per vlan tag
- Frame-Relay DLCI: 2 (Cisco), 8 (IETF)
- HDLC/PPP: 4
- AToM Control Word: 4

All L3 protocols (i.e. IPv4, IPv6, MPLS, CLNS) inherit their MTU settings from L2 MTU.

The default MPLS MTU value of a link equals the interface MTU value. You need to first change the interface MTU in order to be able to increase the MPLS MTU too.

IOS
```
R4#sh mpls int detail
Interface FastEthernet0/0:
        IP labeling not enabled
        LSP Tunnel labeling enabled
        BGP labeling not enabled
        MPLS operational
        MTU = 1500
```

IOS
```
interface FastEthernet0/0
 mtu 1530

R4#sh mpls int detail
Interface FastEthernet0/0:
        IP labeling not enabled
        LSP Tunnel labeling enabled
        BGP labeling not enabled
        MPLS operational
        MTU = 1530
```

IOS
```
interface FastEthernet0/0
 mtu 1530
 mpls mtu 1508
```

```
R4#sh mpls int detail
Interface FastEthernet0/0:
        IP labeling not enabled
        LSP Tunnel labeling not enabled
        BGP labeling not enabled
        MPLS operational
        MTU = 1508
```

In IOS-XR, interface MTU includes the L2 header (i.e. +14 bytes in case of untagged ethernet).

IOS-XR

```
interface TenGigE0/0/0/6
 mtu 9214
```

IOS-XR

```
ASR9k#sh imds int Te0/0/0/6

View: OWN - Owner, L3P - Local 3rd Party, G3P - Global 3rd Party,
      LDP - Local Data Plane, GDP - Global Data Plane, RED - Redundancy

Node 0/RSP0/CPU0 (0x41)

Interface TenGigE0/0/0/6, ifh 0x000002c0 (up, 9214)
  Interface flags:          0x000000000010059f (IFCONNECTOR|IFINDEX
                            |SUP_NAMED_SUB|BROADCAST|CONFIG|HW|VIS|DATA
                            |CONTROL)
  Encapsulation:            ether
  Interface type:           IFT_TENGETHERNET
  Control parent:           None
  Data parent:              None
  Views:                    GDP|G3P

  Protocol        Caps (state, mtu)
  --------        -----------------
  None            spio (up, 9214)
  None            ether (up, 9214)
  arp             arp (up, 9200)
  ipv4            ipv4 (up, 9200)
  mpls            mpls (up, 9200)
  ether_sock      ether_sock (up, 9200)
  ether_link_oam  ether_link_oam (up, 9200)
```

The "`sh imds`" command is hidden in most IOS-XR releases.

IOS-XR

```
ASR9k#sh ip int Te0/0/0/6
TenGigE0/0/0/6 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 10.201.10.221/30
  MTU is 9214 (9200 is available to IP)
```

In IOS-XR, if you change the interface MTU, then you need to take into account the L2 header. If you change the L3 protocol MTU, then it's the same as in IOS.

Fragmentation

If a labeled packet is received and the LSR notices that the outgoing MTU is not big enough for this packet, the LSR strips off the label stack, fragments the IP packet, puts the label stack (after the pop, swap, or push operation) onto all fragments, and forwards the fragments.

If the IP header has the DF bit set, the LSR doesn't fragment the IP packet, but it drops the packet and returns an ICMP error message "Fragmentation needed and do not fragment bit set" to the originator of the IP packet (following the same procedure as with traceroute).

Fragmentation should be avoided if possible.

IOS uses **MRU** in order to "inform" the LSR how big a received labeled packet of a certain FEC can be in order for that to be forwarded out of this LSR without fragmenting it.

IOS

```
R6#sh mpls forwarding-table 19.19.19.19 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
None       No Label   19.19.19.19/32   0             Fa0/0.619
20.6.19.19
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA020BB00008CA010630000008100026B0800
        No output feature configured
```

L3VPN Redistribution

Configuration Steps

- Configure the VRFs
- Configure the RDs
- Configure the import/export RTs
- Assign the PE=>CE interfaces to VRFs
- Configure IGP/BGP between PE-CE
- Configure MP-BGP between PEs
- **Mutually redistribute between MP-BGP and the PE-CE IGP**

BGP<=>RIP

RIP=>BGP
RIP metric => BGP MED (auto)

RIP=>BGP=>RIP
RIP metric => BGP MED => RIP metric (auto)

OTHER=>BGP=>RIP
BGP X => RIP metric (manual)

If "auto" doesn't work (for whatever reason), you can trying clearing the vrf routing table on the PE or you can use the following to set manually the RIP metric:

- redistribute bgp 100 metric transparent
- redistribute bgp 100 metric X
- redistribute bgp 100 route-map X

Clearing of vrf routing table might be needed every time a new prefix is redistributed.

If version 2 is to be used, then it must be defined under the ipv4 vrf address-family on the PE.

RIP metric = hops (0-16)

Configuration

IOS

```
router rip
 address-family ipv4 vrf VPN
  redistribute bgp 200
!
router bgp 100
 address-family ipv4 vrf VPN
```

```
  redistribute rip
```

IOS-XR

```
router rip
 vrf VPN
  redistribute bgp 200
!
router bgp 200
 vrf VPN
  address-family ipv4 unicast
   redistribute rip
```

BGP<=>EIGRP

EIGRP=>BGP
EIGRP composite metric => BGP MED (auto)
EIGRP vector metrics => BGP Extended Cost Community (auto)

EIGRP=>BGP=>EIGRP
EIGRP composite metric => BGP MED => EIGRP composite metric (auto)
EIGRP vector metrics => BGP Extended Cost Community => EIGRP vector metrics (auto)
original internal EIGRP routes appear as internal EIGRP routes when redistributed
original external EIGRP routes appear as external EIGRP routes when redistributed

OTHER=>BGP=>EIGRP
BGP X => EIGRP metrics (manual)
original routes appear as external EIGRP routes when redistributed

If "auto" doesn't work (for whatever reason), you can trying clearing the vrf routing table on the PE or you can use the following to set manually the EIGRP metrics:

- redistribute bgp 100 metric K1 K2 K3 K4 K5
- redistribute bgp 100 route-map X
- redistribute bgp 100 route-policy X
- redistribute bgp 100 & default-metric K1 K2 K3 K4 K5

Clearing of vrf routing table might be needed every time a new prefix is redistributed.

EIGRP vector metrics = K1 K2 K3 K4 K5 (i.e. 1000 10 255 1 1500)

Configuration

IOS

```
router eigrp 100
 address-family ipv4 vrf VPN autonomous-system 1
```

```
  redistribute bgp 200
 exit-address-family
!
router bgp 200
 address-family ipv4 vrf VPN
  redistribute eigrp 1
```

IOS-XR

```
router eigrp 100
 vrf VPN
  address-family ipv4
   autonomous-system 1
   redistribute bgp 200
!
router bgp 200
 vrf VPN
  address-family ipv4 unicast
   redistribute eigrp 1
```

Redistribution of EIGRP into the BGP vrf requires the EIGRP autonomous-system number to be redistributed. Some software releases may accept the global EIGRP process too.

You can use the SoO extended community to prevent any possible loops.

BGP<=>ISIS

ISIS=>BGP
ISIS metric => BGP MED (auto)

ISIS=>BGP=>ISIS
ISIS metric => BGP MED => ISIS metric (auto)

OTHER=>BGP=>ISIS
BGP X => ISIS metric (manual)

You can use the following to set manually the ISIS metric:

- redistribute bgp 100 metric X
- redistribute bgp 100 route-map X

Clearing of vrf routing table might be needed every time a new prefix is redistributed.

ISIS metric = hops (10)

Configuration

IOS

```
router isis 100
 address-family ipv4 vrf VPN
  redistribute bgp 200
!
router bgp 200
 address-family ipv4 vrf VPN
  redistribute isis 100
```

IOS-XR

```
router isis 100
 vrf VPN
  redistribute bgp 200
!
router bgp 200
 vrf VPN
  address-family ipv4 unicast
   redistribute isis 100
```

Redistribution doesn't take into account the IS-IS connected routes. You have to explicitly define them.

In order to void a possible loop while doing redistribution (when L1 is involved), you can change the distance of the ISIS advertised routes (excluding connected) on the PE to be higher than BGP's.

IOS

```
router isis 100
 vrf VPN
 distance 201 0.0.0.0 255.255.255.255 ISIS-NOT-CONNECTED-ACL
```

BGP<=>OSPF

OSPF=>BGP
OSPF metric => BGP MED + 1 (auto)
OSPF Area/LSA => BGP extended community "OSPF RT" (auto)

OSPF=>BGP=>OSPF
OSPF metric => BGP MED + 1 => OSPF metric (auto)

- original intra-area routes appear as inter-area routes when redistributed (if same OSPF Domain-ID)
- original intra-area routes appear as external-2 routes when redistributed (if different OSPF Domain-ID)
- type-4 LSAs are not redistributed into BGP
- original external routes appear as external-2 routes when redistributed (requires "match external" in redistribution from OSPF to BGP)

OTHER=>BGP=>OSPF
BGP X => OSPF metric (manual)

You can always use the following to manually set the OSPF metric:

- redistribute bgp 200 metric X
- redistribute bgp 200 route-map X

Clearing of vrf routing table might be needed every time a new prefix is redistributed.

OSPF metric = interface cost (0-65535)

"OSPF RT" Extended Community

"OSPF RT" format is "Area:LSA-Type:External-Type"

LSA Type to OSPF RT conversion

- Type-1/2 => RT 2
- Type-3 => RT 3
- Type-5 => RT 5
- Type-7 => RT 7
- Sham-links => RT 129

Examples

- OSPF RT:0.0.0.0:2:0
  - area 0.0.0.0
  - LSA-Type 1/2
- OSPF RT:0.0.0.0:5:0
  - LSA-Type 5
  - External 1
- OSPF RT:0.0.0.0:5:1
  - LSA-Type 5
  - External 2

Configuration

IOS

```
router ospf 100 vrf VPN
 redistribute bgp 200 subnets
!
router bgp 200
 address-family ipv4 vrf VPN
  redistribute ospf 100 vrf VPN
```

IOS-XR

```
router ospf 100
 vrf VPN
```

```
  redistribute bgp 200
!
router bgp 200
 vrf VPN
  address-family ipv4 unicast
   redistribute ospf 100
```

In IOS, if you don't include the vrf name in the redistribution of OSPF into BGP, it gets automatically added to the configuration.

The DN Bit and the VPN Route Tag

For a PE it is necessary to know if a particular prefix has been learned from another PE router, in order to avoid re-advertisement of it into BGP and cause a loop.

Two mechanisms are mainly used for loop prevention when OSPF is used as PE-CE protocol.

- the DN bit
- the VPN Route (or OSPF Domain) tag

By default, when a type 3, 5, 7 LSA is sent from a PE to a CE, the DN bit is set by the PE.

When another PE receives from a CE router, a type 3, 5, 7 LSA with the DN bit set, the prefix information from that LSA is not used during the OSPF route calculation, which means that the prefix doesn't get installed into the PE's BGP table.

Almost all Cisco software releases support the setting of DN bit only for Type-3 LSAs and they use a 32-bit VPN Route tag for Type-5/7 LSAs. The configuration and inclusion of the VPN Route Tag is required by all implementations for backward compatibility with older implementations that do not set the DN bit in type 5/7 LSAs.

If a PE router receives an LSA that contains the same VPN Route Tag as the locally configured tag, then the local PE router knows that another PE router (from the same domain) generated this route and the LSA is ignored.

- 16bit ASNs
  - VPN Route tag Format: 1101 000000000000 ASN_of_VPN_Backbone
- 32bit ASNs
  - VPN Route tag must be defined manually

You can change this default value by using the "`domain-tag`" command within the OSPF VRF process configuration.

IOS

```
router ospf 100 vrf VPN
```

```
 domain-tag 12345
```

IOS-XR

```
router ospf 100
 vrf TEST
  domain-tag 12345
```

In case of **Multi-VRF (VRF-Lite)**, the router that is accepting the LSA with the DN bit is actually a CE router with no BGP VPNv4 functionality, so there is no danger of redistributing this prefix into BGP. In order to **bypass this DN bit check**, the following configuration can be enabled.

IOS

```
router ospf 100 vrf VPN
 capability vrf-lite
```

IOS-XR

```
router ospf 100
 vrf VPN
  disable-dn-bit-check
```

Verification

IOS

```
R1#sh ip ospf 100 database summary 10.7.7.7

            OSPF Router with ID (10.1.3.1) (Process ID 100)

                Summary Net Link States (Area 0)

  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 1196
  Options: (No TOS-capability, DC, Downward)
  LS Type: Summary Links(Network)
  Link State ID: 10.7.7.7 (summary Network Number)
  Advertising Router: 10.1.2.2
  LS Seq Number: 80000005
  Checksum: 0x2761
  Length: 28
  Network Mask: /32
        MTID: 0         Metric: 2


R1#sh ip ospf 100 database external 7.7.7.7

            OSPF Router with ID (10.1.3.1) (Process ID 100)

                Type-5 AS External Link States
  Routing Bit Set on this LSA in topology Base with MTID 0
  LS age: 1302
  Options: (No TOS-capability, DC)
  LS Type: AS External Link
  Link State ID: 7.7.7.7 (External Network Number )
  Advertising Router: 10.1.2.2
  LS Seq Number: 80000004
  Checksum: 0x6DCF
  Length: 36
  Network Mask: /32
        Metric Type: 2 (Larger than any link state path)
        MTID: 0
        Metric: 20
        Forward Address: 0.0.0.0
        External Route Tag: 3489661028
```

Links

- IETF - RFC 4576 (the DN-bit)

OSPF Domain-ID

OSPF Domain-ID is an attribute that defines how (internal, external) the OSPF routes will be transferred from one CE to another CE over their PEs BGP VPNv4 session.

On a PE, if the OSPF Domain-ID of the received BGP prefixes (encoded as extended community) is the same as the OSPF Domain-ID of the local OSPF process, then:

- the MPLS core is treated like a SuperBackbone area (which is considered higher than area 0)
- the PE is treated like an ABR (instead of an ASBR)
- internal routes are being redistributed as Type-3 LSAs (instead of Type-5)

IOS-XR uses a null Domain-ID by default, so this needs to be changed if the other PE is running IOS (which is encoding the OSPF process-id as domain-id). OSPF Domain-ID needs to be changed on the PEs (where redistribution between BGP and OSPF takes place), not on the CEs.

The "type" value can be different is some cases for backwards compatibility (like in 0005 vs 8005).

Detailed Steps

OSPF=>BGP redistribution on PE1

- if the OSPF Domain tag of the local OSPF process is the same as the VPN Route tag of the prefix, then that route isn't installed into BGP
- if the OSPF DN bit check is enabled in the local OSPF process and the OSPF route has this bit set, then that route isn't installed into BGP
- if the route is installed into BGP

    - the Domain-ID of the local OSPF process is encoded into OSPF DOMAIN ID community on the prefix
    - the area and the LSA type of the OSPF prefix is encoded into OSPF RT community on the prefix
    - the Router-ID of the local OSPF process is encoded into OSPF ROUTER ID community on the prefix

BGP=>OSPF redistribution on PE2

- if the Domain-ID of the local OSPF process is the same as the OSPF DOMAIN ID community of the prefix, then that route is passed to the CE as internal else as external

Configuration

IOS

```
router ospf 100 vrf VPN
 domain-id type 0005 value 000000440101
```

IOS-XR

```
router ospf 100
 vrf VPN
  domain-id type 0005 value 000000440101
```

Verification

You can use "sh ip ospf" to see the Domain-ID of the local OSPF process.

You can use "sh bgp vpn4 unicast" to see the Domain-ID encoded as extended community in the BGP prefixes (OSPF RT is included too).

```
R2#sh ip ospf 100
 Routing Process "ospf 100" with ID 10.1.2.2
   Domain ID type 0x0005, value 0x000000440101
 Start time: 00:13:37.092, Time elapsed: 00:36:17.144
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Supports Link-local Signaling (LLS)
 Supports area transit capability
 Connected to MPLS VPN Superbackbone, VRF VPN
 Event-log disabled
 It is an area border and autonomous system boundary router
 Redistributing External Routes from,
    bgp 100, includes subnets in redistribution


R2#sh bgp vpnv4 unicast vrf VPN 1.1.1.1/32
BGP routing table entry for 100:1:1.1.1.1/32, version 2
Paths: (1 available, best #1, table VPN)
  Advertised to update-groups:
     1
  Local
    10.1.2.1 from 0.0.0.0 (2.2.2.2)
      Origin incomplete, metric 2, localpref 100, weight 32768, valid,
sourced, best
      Extended Community: RT:100:1 OSPF DOMAIN ID:0x0005:0x000000440101
        OSPF RT:0.0.0.0:3:0 OSPF ROUTER ID:10.1.2.2:0
      mpls labels in/out 28/nolabel


R2#sh ip ospf 100 database

            OSPF Router with ID (10.1.2.2) (Process ID 100)
...
                Summary Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
1.1.1.1         10.1.1.1        980         0x80000002 0x00F336
...
```

LSA Type-3 (Summary) in local OSPF table is encoded as "OSPF RT:0.0.0.0:3:0" in local BGP table.

Propagation of OSPF routes between CE1 and CE2

- same domain-id
  - CE1 O => CE2 IA
- different domain-id
  - CE1 O => CE2 E2
- sham-link (regardless of domain-id)
  - CE1 O => CE2 O

Extra care needs to be taken **if route tags are changed manually on OSPF=>BGP redistribution**, because external OSPF routes are tagged by the BGP ASN when BGP=>OSPF redistribution takes place, which means that the original tag is lost (which **could lead to a loop**)

IOS

```
R6#sh ip route 1.1.1.1
Routing entry for 1.1.1.1/32
  Known via "ospf 100", distance 110, metric 2
  Tag Complete, Path Length == 1, AS 100, , type extern 2, forward metric
1
  Last update from 10.10.10.5 on POS4/0, 00:00:03 ago
  Routing Descriptor Blocks:
  * 10.10.10.5, from 10.10.10.5, 00:00:03 ago, via POS4/0
      Route metric is 2, traffic share count is 1
      Route tag 3489661028
```

If a BGP VPNv4 route is redistributed into OSPF, then redistributed into another IGP like RIP (where all the information (DN bit, VPN Route-Tag) needed to prevent looping is lost), and then redistributed back into OSPF, then it is possible that it could be redistributed back into BGP as a VPNv4 route, thereby causing a loop.

You can use route tags at every step of redistribution in order to avoid possible routing loops, either caused by the above scenario or by mutual redistribution in two places.

Inter-AS MPLS L3VPN

Inter-AS MPLS L3VPN Options are defined in RFC 4364.

Inter-AS Options

- **Inter-AS Option A** (Back-to-Back VRF)
    - one logical/physical interface per VRF in the interconnection
    - one PE-CE eBGP/IGP session per VRF between ASBRs
    - IP traffic between ASBRs
    - no need for common RDs/RTs between ASNs
    - 2 LSPs and 1 IP path from one PE to the other PE
- **Inter-AS Option B** (MP-eBGP between ASBRs)
    - one physical/logical interface for all VRFs in the interconnection
    - eBGP VPNv4 between ASBRs
    - MPLS traffic between ASBRs
    - common RDs/RTs between ASNs (unless RT rewrite is used)
    - next-hop-self on each ASBR for iBGP
        - 3 LSPs from one PE to the other PE
    - redistribute connected/static on each ASBR for the interconnection
        - 2 LSPs from one PE to the other PE
        - filter to redistribute only the peer's address
    - multihop (loopback) peering between ASBRs
        - 2 LSPs from one PE to the other PE
        - static routes for peer's loopback on each ASBR
        - LDP between ASBRs
        - MPLS static label binding for peer's loopback pointing to interconnection on each ASBR
- **Inter-AS Option C** (Multihop MP-eBGP between RRs/PEs)
    - one physical/logical interface for all VRFs in the interconnection
    - labeled eBGP session between ASBRs for next-hop exchange
    - multihop eBGP VPNv4 session between RRs
    - MPLS traffic between ASBRs
    - common RDs/RTs between ASNs (unless RT rewrite is used)
    - change next-hop on each VPNv4 RR for the eBGP session (default)
        - 2 LSPs from one PE to the other PE
    - next-hop-unchanged on each VPNv4 RR for the eBGP session
        - 1 LSP from one PE to the other PE
    - eBGP session between ASBRs with directly connected interfaces
        - next-hop-self on each ASBR for the iBGP sessions
    - multihop (loopback) eBGP session between ASBRs with loopbacks
        - static routes for peer's loopback on each ASBR
        - LDP between ASBRs
        - MPLS static label binding for peer's loopback pointing to interconnection on each ASBR

The transport label changes whenever the next-hop changes.

Inter-AS Option A

ASBR-1

IOS

```
ip vrf VPN1
 rd 1:100
 route-target 1:100
!
ip vrf VPN2
 rd 1:200
 route-target 1:200
!
interface FastEthernet0/0
 description ** Inter-AS NNI **
!
interface FastEthernet0/0.10
 description ** Customer VPN1 **
 encapsulation dot1q 10
 ip vrf forwarding VPN1
 ip address 10.10.10.1 255.255.255.0
!
interface FastEthernet0/0.20
 description ** Customer VPN2 **
 encapsulation dot1q 20
 ip vrf forwarding VPN2
 ip address 20.20.20.1 255.255.255.0
!
router bgp 1
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 update-source Loopback0
 neighbor 1.1.1.1 description iBGP-VPNv4
!
 address-family vpnv4
  neighbor 1.1.1.1 activate
  neighbor 1.1.1.1 send-community extended
  neighbor 1.1.1.1 next-hop-self
 exit-address-family
!
 address-family ipv4 vrf VPN1
  neighbor 10.10.10.2 remote-as 2
  neighbor 10.10.10.2 activate
 exit-address-family
!
 address-family ipv4 vrf VPN2
```

```
  neighbor 20.20.20.2 remote-as 2
  neighbor 20.20.20.2 activate
 exit-address-family
```

ASBR-2

IOS

```
ip vrf test1
 rd 2:100
 route-target 2:100
!
ip vrf test2
 rd 2:200
 route-target 2:200
!
interface FastEthernet0/0
 description ** Inter-AS NNI **
!
interface FastEthernet0/0.10
 description ** Customer VPN1 **
 encapsulation dot1q 10
 ip vrf forwarding VPN1
 ip address 10.10.10.2 255.255.255.0
!
interface FastEthernet0/0.20
 description ** Customer VPN2 **
 encapsulation dot1q 20
 ip vrf forwarding VPN2
 ip address 20.20.20.2 255.255.255.0
!
router bgp 2
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 2.2.2.2 description iBGP-VPNv4
!
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 2.2.2.2 next-hop-self
 exit-address-family
!
 address-family ipv4 vrf VPN1
  neighbor 10.10.10.1 remote-as 1
  neighbor 10.10.10.1 activate
 exit-address-family
!
 address-family ipv4 vrf VPN2
  neighbor 20.20.20.1 remote-as 1
  neighbor 20.20.20.1 activate
 exit-address-family
```

You can also use a different router-id per VRF, using the "`bgp router-id`" under each vrf address-family.

Inter-AS Option B

ASBR-1

IOS

```
interface FastEthernet0/0
 description ** Inter-AS NNI **
 ip address x.x.x.x
 mpls bgp forwarding
!
router bgp 1
 no bgp default route-target filter
 neighbor PE-1 remote-as 1
 neighbor PE-1 update-source Loopback0
 neighbor PE-1 description MP-iBGP with PE-1
 neighbor ASBR-2 remote-as 2
 neighbor ASBR-2 description MP-eBGP with ASBR-2
 no auto-summary
!
 address-family vpnv4
  neighbor PE-1 activate
  neighbor PE-1 send-community extended
  neighbor PE-1 next-hop-self
  neighbor ASBR-2 activate
  neighbor ASBR-2 send-community extended
 exit-address-family
```

ASBR-2

IOS

```
interface FastEthernet0/0
 description ** Inter-AS NNI **
 ip address x.x.x.x
 mpls bgp forwarding
!
router bgp 2
 no bgp default route-target filter
 neighbor PE-2 remote-as 2
 neighbor PE-2 update-source Loopback0
 neighbor PE-2 description MP-iBGP with PE-2
 neighbor ASBR-1 remote-as 1
 neighbor ASBR-1 description MP-eBGP with ASBR-1
!
 address-family vpnv4
  neighbor PE-2 activate
  neighbor PE-2 send-community extended
  neighbor PE-2 next-hop-self
  neighbor ASBR-1 activate
  neighbor ASBR-1 send-community extended
 exit-address-family
```

Inter-AS Option C

RR-1

IOS

```
router bgp 1
 no synchronization
 neighbor PE-1 remote-as 1
 neighbor PE-1 update-source Loopback0
 neighbor PE-1 description MP-iBGP with PE-1
 neighbor ASBR-1 remote-as 1
 neighbor ASBR-1 update-source Loopback0
 neighbor ASBR-1 description MP-iBGP with ASBR-1
 neighbor RR-2 remote-as 2
 neighbor RR-2 ebgp-multihop 255
 neighbor RR-2 update-source Loopback0
 neighbor RR-2 description MP-eBGP with RR-2
 no auto-summary
!
 address-family vpnv4
  neighbor PE-1 activate
  neighbor PE-1 send-community extended
  neighbor PE-1 route-reflector-client
  neighbor ASBR-1 activate
  neighbor ASBR-1 send-community extended
  neighbor ASBR-1 route-reflector-client
  neighbor RR-2 activate
  neighbor RR-2 send-community extended
  neighbor RR-2 next-hop-unchanged
  exit-address-family
```

ASBR-1

IOS

```
interface FastEthernet0/0
 description ** Inter-AS NNI **
 ip address x.x.x.x
 mpls bgp forwarding
!
route-map PE2-TO-IGP permit 10
 match ip address PE-2
!
router IGP 100
 redistribute bgp 1 route-map PE2-TO-IGP
!
router bgp 1
 no synchronization
 network PE-1 mask 255.255.255.255
!
 neighbor RR-1 remote-as 1
 neighbor RR-1 update-source Loopback0
 neighbor RR-1 description MP-iBGP to RR-1
 neighbor ASBR-2 remote-as 2
 neighbor ASBR-2 send-label
 veighbor ASBR-2 description MP-eBGP to ASBR-2
 no auto-summary
!
 address-family vpnv4
  neighbor RR-1 activate
  neighbor RR-1 send-community extended
 exit-address-family
```

ASBR-2

IOS

```
interface FastEthernet0/0
 description ** Inter-AS NNI **
 ip address x.x.x.x
 mpls bgp forwarding
!
route-map PE1-TO-IGP permit 10
 match ip address PE-1
!
router IGP 200
 redistribute bgp 2 route-map PE1-TO-IGP
!
router bgp 2
 network PE-2 mask 255.255.255.255
!
```

```
 neighbor RR-2 remote-as 2
 neighbor RR-2 update-source Loopback0
 neighbor RR-2 description MP-iBGP to RR-2
 neighbor ASBR-1 remote-as 1
 neighbor ASBR-1 send-label
 neighbor ASBR-1 description MP-eBGP to ASBR-1
!
 address-family vpnv4
  neighbor RR-2 activate
  neighbor RR-2 send-community extended
 exit-address-family
```

RR-2

IOS

```
router bgp 2
 neighbor PE-2 remote-as 2
 neighbor PE-2 update-source Loopback0
 neighbor PE-2 description MP-iBGP with PE-2
 neighbor ASBR-2 remote-as 2
 neighbor ASBR-2 update-source Loopback0
 neighbor ASBR-2 description MP-iBGP with ASBR-2
 neighbor RR-1 remote-as 1
 neighbor RR-1 ebgp-multihop 255
 neighbor RR-1 update-source Loopback0
 neighbor RR-1 description MP-eBGP with RR-1
!
 address-family vpnv4
  neighbor PE-2 activate
  neighbor PE-2 send-community extended
  neighbor PE-2 route-reflector-client
  neighbor ASBR-2 activate
  neighbor ASBR-2 send-community extended
  neighbor ASBR-2 route-reflector-client
  neighbor RR-1 activate
  neighbor RR-1 send-community extended
  neighbor RR-1 next-hop-unchanged
 exit-address-family
```

In IOS-XR, in order to send IPv4 prefixes with labels over a labeled BGP session, the IOS-XR router must be the originator of the prefixes. On the other hand, an IOS router can send labeled IPv4 prefixes over a labeled BGP session whether it's the originator or not of those prefixes.

If an output route-map is applied on a labeled BGP session, then labels will be added only to those prefixes that have the command "`set mpls-label`" under the relevant statement in the route-map. Generally, if a router is advertising IPv4 prefixes with labels, then you can use an output route-map (with the "`set mpls-label`" command) to specify which prefixes will be sent with a label.

You need to disable the default RT filter from the ASBRs, unless they have all the VRFs locally configured or they are VPNv4 RRs.

In most IOS software releases, the command "`mpls bgp forwarding`" is added automatically under the eBGP peering interface when a VPNv4 or labeled BGP session is configured between directly connected peers. **If you use loopbacks for peering, then you must manually configure it**. Always verify its existence, together with the interface's mpls operational state.

IOS

```
R1#sh mpls int
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/0.13     Yes (ldp)     No       No  No     Yes
FastEthernet0/0.30     No            No       Yes No     Yes
```

Generally, Cisco software requires a /32 route for each next-hop that should be label switched. In the Inter-AS B/C options, in IOS-XR you must add manually a /32 static route for the peer address of the interconnection in order to create a label for that. IOS creates automatically a /32 connected route when the relevant VPNv4 or labeled BGP session comes up.

IOS-XR

```
router static
 address-family ipv4 unicast
  10.10.10.2/32 GigabitEthernet0/2/1/2
```

IOS

```
Dec 29 15:45:30.703: %BGP-5-ADJCHANGE: neighbor 10.10.10.2 Up
Dec 29 15:45:30.707: CONN: add connected route, idb: FastEthernet0/0.30,
addr: 10.10.10.2, mask: 255.255.255.255
```

If you want to achieve load-sharing in a MPLS L3VPN environment with RRs, you can use a different RD per PE in combination with BGP multipath.

Inter-AS scenarios emulated in GNS3 might sometimes cause very large delays in data forwarding. Increase the ping/traceroute timeout in order to verify connectivity.

Static Label Bindings

In some cases you don't have the option of enabling LDP or having a VPNv4 or labeled BGP session between directly connected peers, but you still need to have the label switching functionality on their interconnection.

i.e. if you configure the following **static route in order to reach peer's loopback**:

IOS

```
ip route 19.19.19.19 255.255.255.255 12.1.19.19
```

IOS

```
R1#sh mpls forwarding-table 19.19.19.19 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
24         No Label   19.19.19.19/32   0             Fa0/0  12.1.19.19
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA02141C0008CA0417EC0000810000770800
        No output feature configured
```

then you need to also **add a static (outgoing) label binding** for that:

IOS

```
mpls static binding ipv4 19.19.19.19 255.255.255.255 output 12.1.19.19
implicit-null
```

IOS

```
R1#sh mpls static binding
19.19.19.19/32: Incoming label: none;
  Outgoing labels:
     12.1.19.19           implicit-null

R1#sh mpls forwarding-table 19.19.19.19 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
24         Pop Label  19.19.19.19/32   0             Fa0/0  12.1.19.19
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA02141C0008CA0417EC0000810000770800
        No output feature configured
```

At the same time, you must **enable MPLS on this interface without using LDP**:

IOS

```
R1#sh mpls int FastEthernet0/0
Interface              IP            Tunnel   BGP Static Operational
```

IOS

```
interface FastEthernet0/0
 mpls bgp forwarding
```

IOS

```
R1#sh mpls int FastEthernet0/0
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/0        No            No       Yes No     Yes
```

Static Label Bindings per Interface

- multiaccess interfaces
    - next-hop ip address required
    - label required
- point-to-point interfaces
    - interface required

The above differentiation per interface is applicable only on specific software releases. The multiaccess interface is the common one.

If you must configure specific static labels, then you must first define the label range (which will sometimes require a reload).

Implicit-null is used in the above example due to PHP (pop label) that must happen for the directly connected peer.

Inter-AS L3VPN

If you want to follow a Inter-AS L3VPN path (assuming control-plane has been setup correctly), then you can execute the following algorithm:

- first router (start PE)
    - Find the VPN label for the prefix
    - Find the Transport label(s) for the prefix's next-hop
- n router
    - Follow the Transport top label swaps until there is a "Pop Label" for next router
- n+1 router
    - Find the local VPN label for the prefix
        - If VPN label is "nolabel", then
            - router is the end PE
            - VPN is locally attached
        - If VPN label is other, then
            - router is an RR/ASBR
            - find the Transport label(s) for the prefix's new next-hop
            - go to "n router"
        - If VPN label doesn't exist, then
            - multiple Transport labels exist
            - go to "n router"

If the route is learned from IGP, the Transport label must be allocated through LDP/RSVP.
If the route is learned from BGP, the Transport label must be allocated through BGP.

Example

R6(PE1)=>R4(P1)=>XR1(ASBR1)=>R1(ASBR2)=>R3(P2)=>R2(PE3)

Start PE

IOS

```
R6#sh bgp vpnv4 unicast all 7.7.7.7/32
BGP routing table entry for 102:202:7.7.7.7/32, version 36
Paths: (1 available, best #1, table VPN_B)
  Not advertised to any peer
  100
    2.2.2.2 (metric 20) from 20.20.20.20 (20.20.20.20)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:102:202 0x8800:32768:0 0x8801:1:130560
        0x8802:65281:25600 0x8803:65281:1500 0x8806:0:0
      mpls labels in/out nolabel/26
```

VPN label is 26

IOS

```
R6#sh mpls forwarding-table 2.2.2.2 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
None       23         2.2.2.2/32       0             Fa0/0.46   20.4.6.4
        MAC/Encaps=18/26, MRU=1496, Label Stack{16 23}
        CA0611100000CA0115B000008100002E8847 0001000000017000
        No output feature configured
```

Transport label is 16/23, VPN label is 26

IOS

```
R4#sh mpls forwarding-table labels 16 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  19.19.19.19/32   18896         Fa0/0.419
20.4.19.19
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA02141C0008CA0611100000810001A38847
        No output feature configured
```

Transport label is 23, VPN label is 26

IOS

```
XR1#sh mpls forwarding-table labels 23 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
```

```
Label      Label      or Tunnel Id      Switched      interface
23         20         2.2.2.2/32        22628         Fa0/0.119  12.1.19.1
        MAC/Encaps=18/22, MRU=1500, Label Stack{20}
        CA0417EC0000CA02141C000881000077 8847 00014000
        No output feature configured
```

Transport label is 20, VPN label is 26

IOS

```
R1#sh mpls forwarding-table labels 20 detail
Local      Outgoing   Prefix            Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id      Switched      interface
20         19         2.2.2.2/32        24518         Fa0/0.13   10.1.3.3
        MAC/Encaps=18/22, MRU=1500, Label Stack{19}
        CA0711100000CA0417EC00008100000D8847 00013000
        No output feature configured
```

Transport label is 19, VPN label is 26

IOS

```
R3#sh mpls forwarding-table labels 19 detail
Local      Outgoing   Prefix            Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id      Switched      interface
19         Pop Label  2.2.2.2/32        85693         Fa0/0.23   10.2.3.2
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA0517EC0000CA071110000081000017 8847
        No output feature configured
```

VPN label is 26

IOS

```
R2#sh bgp vpnv4 unicast all 7.7.7.7/32
BGP routing table entry for 102:202:7.7.7.7/32, version 4
Paths: (1 available, best #1, table VPN_B)
  Advertised to update-groups:
     1
  Local
    40.2.7.7 from 0.0.0.0 (2.2.2.2)
      Origin incomplete, metric 156160, localpref 100, weight 32768,
valid, sourced, best
      Extended Community: RT:102:202 Cost:pre-bestpath:128:156160
        0x8800:32768:0 0x8801:1:130560 0x8802:65281:25600
0x8803:65281:1500
        0x8806:0:0
      mpls labels in/out 26/nolabel
```

End PE found

RT Rewrite

It is used mainly in Inter-AS topologies, when there is a need to keep different RTs between the ASes. It allows the ASBR (or any other router that's involved) to replace the peer ASN's RTs with their own.

Configuration Steps

- define the RTs to be replaced
- configure a route-map that matches the above RTs, deletes them and then adds the new RTs
- apply the route-map to the bgp neighbor session

IOS

```
ip extcommunity-list 1 permit rt 200:1
ip extcommunity-list 2 permit rt 200:2
!
route-map RT-REWRITE-ROUTEMAP permit 10
 match extcommunity 1
 set extcomm-list 1 delete
 set extcommunity rt 100:1 additive
 continue 20
!
route-map RT-REWRITE-ROUTEMAP permit 20
 match extcommunity 2
 set extcomm-list 2 delete
 set extcommunity rt 100:2 additive
!
route-map RT-REWRITE-ROUTEMAP permit 30
!
router bgp 100
 neighbor 10.10.10.2 remote-as 200
 !
 address family vpnv4
  neighbor 10.10.10.2 activate
  neighbor 10.10.10.2 send-community extended
  neighbor 10.10.10.2 route-map RT-REWRITE-ROUTEMAP in
```

Use the "additive" keyword when setting the new RT in order to not erase all other extended communities.

Use the "continue" statement (in ingress route-maps) when you need to rewrite more than one RTs in the same prefix.

CsC

CsC (Carrier supporting Carrier) is defined in RFC 4364.

Control-Plane

- The Customer Carrier PEs run BGP VPNv4 in order to exchange VPN labels
- The Customer Carrier routers run IGP+LDP (or iBGP+Label) in order to exchange all their internal BGP next-hops and their labels
- The CsC-PEs and CsC-CEs run eBGP (or IGP) in order to exchange BGP next-hop prefixes
- The CsC-PEs and CsC-CEs run eBGP+Label (or IGP+LDP) in order to exchange labels for the BGP next-hop prefixes
- The Backbone Carrier routers run IGP+LDP in order to exchange all their internal BGP next-hops and their labels

The Backbone Carrier offers a MPLS VPN service to the Customer Carrier which in turn offers a MPLS VPN or Internet service to its customers.

The Backbone Carrier doesn't need to know the final customer prefixes.

Using IGP+LDP in CsC is not as risky as with Inter-AS MPLS VPN Option 3 because:

- Customer Carrier internal routes are put into a specific VRF in the Backbone Carrier
- No Backbone Carrier internal routes are distributed into the Customer Carrier network

You can have multiple Backbone Carriers, using Inter-AS MPLS L3VPN for interconnection.

By default a CsC-PE runs PHP towards the CsC-CE. If using an ipv4-labeled PE-CE session, you can change this behavior (in order to keep the QoS consistent across providers) by using the "`neighbor x.x.x.x send-label` **`explicit-null`**" on the CsC-CE.

IOS-XR supports only the use of Labeled BGP as a PE-CE protocol in CsC topologies. LDP (+IGP) is not supported.

CsC Load Balancing

Load balancing between CsC-PE and CsC-CE can be achieved with:

- directly connected loopback peering for one pair of PE/CE
  - one eBGP session between neighbors
  - multiple static routes for each other's loopback
  - mpls forwarding on all directly connected physical interfaces
- eBGP multipath for multiple pairs of PEs/CEs
  - maximum-paths under bgp & vrf address family on PE

  - maximum-paths under bgp on CE

When using static routes, you also need to define the outgoing interface and the next-hop.

"`mpls bgp forwarding`" is not automatically added, because the BGP session is not between directly connected neighbors. You have to add it yourself.

Configuration

BC = Backbone Carrier (AS10)
CC = Customer Carrier (AS100)
C = Customer (AS200)

Backbone Carrier runs IS-IS or OSPF with MPLS/LDP in its core

Backbone Carrier (CsC-PE1) runs OSPF+LDP with Customer Carrier (CsC-CE1)
Backbone Carrier (CsC-PE2) runs eBGP+Label with Customer Carrier (CsC-CE2)

Customer Site 1 (C-CE1) runs OSPF with Customer Carrier (CC-PE1)
Customer Site 2 (C-CE2) runs ISIS with Customer Carrier (CC-PE2)

CC-PE (Customer Carrier PE serving the final customer site) and CsC-CE (Carrier supporting Carrier CE) functionalities can be collapsed into a single router.

CsC-PE1 and CsC-PE2 run iBGP VPNv4 in order to exchange Customer Carrier prefixes/labels
CsC-CE1 and CsC-CE2 run iBGP VPNv4 in order to exchange Customer prefixes/labels

IGP+LDP between CsC-PE1 and CsC-CE1

CsC-PE1 (IOS)

```
! for connectivity to BC core (IGP+LDP)
mpls ldp router-id Loopback0
mpls label protocol ldp
!
interface Ethernet0/2
 description ** Link to BC core **
 ip address x.x.x.x
 mpls ip
!
router isis/ospf x
!
! for connectivity to CsC-CE1 (OSPF+LDP)
vrf definition CC-VPN
 rd 10:X
 route-target 10:X
!
 address-family ipv4
 exit-address-family
!
interface Ethernet1/0
 description ** Link to CsC-CE1 **
 vrf forwarding CC-VPN
 ip address x.x.x.x
 mpls ip
!
router ospf 10 vrf CC-VPN
 redistribute bgp 10 subnets
 network x.x.x.x area 0
!
```

! for connectivity to BC-PE2 (iBGP VPNv4)

```
router bgp 10
 no bgp default ipv4-unicast
 neighbor BC-PE2 remote-as 10
 neighbor BC-PE2 update-source Loopback0
 !
 address-family vpnv4
  neighbor BC-PE2 activate
  neighbor BC-PE2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf CC-VPN
  redistribute ospf 10 vrf CC-VPN
 exit-address-family
```

CsC-CE1 (IOS)

! for connectivity to CsC-PE1 (OSPF+LDP)

```
mpls ldp router-id Loopback0
mpls label protocol ldp
!
interface Ethernet1/0
 description ** Link to CsC-PE1 **
 ip address x.x.x.x
 mpls ip
!
router ospf 10
 network x.x.x.x area 0
!
```

CC-PE1 (IOS)

! for connectivity to C-CE1 (OSPF+VRF)

```
vrf definition C-VPN
 rd 100:Y
 route-target 100:Y
 !
 address-family ipv4
 exit-address-family
!
interface Ethernet1/3
 description ** Link to C-CE1 **
 vrf forwarding C-VPN
 ip address y.y.y.y
!
router ospf 200 vrf C-VPN
 redistribute bgp 100 subnets
 network y.y.y.y area 0
!
! for connectivity to CC-PE2 (iBGP VPNv4)
router bgp 100
 no bgp default ipv4-unicast
 neighbor CC-PE2 remote-as 100
 neighbor CC-PE2 update-source Loopback0
 !
 address-family vpnv4
  neighbor CC-PE2 activate
  neighbor CC-PE2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf C-VPN
  redistribute ospf 200 vrf C-VPN
 exit-address-family
!
```

<u>BGP+Label between CsC-PE2 and CsC-CE2</u>

<u>CsC-PE2 (IOS-XR)</u>

```
! for connectivity to BC core (IGP+LDP)
mpls ldp router-id Loopback0
mpls label protocol ldp
!
router isis/ospf x
!
mpls ldp
 router-id x.x.x.x
 interface x
!
! for connectivity to CsC-CE2  (eBGP+Label)
vrf CC-VPN
 address-family ipv4 unicast
  import route-target
   10:X
  export route-target
   10:X
!
interface GigabitEthernet0/2/1/1
 description ** Link to CsC-CE2 **
 vrf CC-VPN
 ipv4 address x.x.x.x
!
router static
 vrf CC-VPN
  address-family ipv4 unicast
   CsC-CE2/32 GigabitEthernet0/2/1/1
!
router bgp 10
 address-family ipv4 unicast
 !
 vrf CC-VPN
  rd 10:X
  address-family ipv4 unicast
   network x.x.x.x
   allocate-label all
  !
  neighbor CsC-CE2
   remote-as 100
   address-family ipv4 unicast
    route-policy PASS-RPL in
    route-policy PASS-RPL out
    as-override
    send-extended-community-ebgp
   !
   address-family ipv4 labeled-unicast
    route-policy PASS-RPL in
    route-policy PASS-RPL out
    as-override
    send-extended-community-ebgp
!
route-policy PASS-RPL
  pass
end-policy
!
! for connectivity to BC-PE1 (iBGP VPNv4)
router bgp 10
 address-family vpnv4 unicast
 !
 neighbor BC-PE1
  remote-as 10
  update-source Loopback0
  address-family vpnv4 unicast
```

CsC-CE2 (IOS)

! for connectivity to CsC-PE2 (eBGP+Label)

```
interface Ethernet1/0
 description ** Link to CsC-PE2 **
 ip address x.x.x.x
 mpls bgp forwarding
!
router bgp 100
 no bgp default ipv4-unicast
 neighbor CsC-PE2 remote-as 10
 !
 address-family ipv4
  neighbor CsC-PE2 activate
  neighbor CsC-PE2 send-label
 exit-address-family
!
```

! for connectivity to C-CE2 (ISIS+VRF)

```
vrf definition C-VPN
 rd 100:Y
 route-target 100:Y
 !
 address-family ipv4
 exit-address-family
!
interface Ethernet1/3
 description ** Link to C-CE2 **
 vrf forwarding C-VPN
 ip address y.y.y.y
 ip router isis 200
!
router isis 200
 vrf C-VPN
  redistribute bgp 100
!
```

! for connectivity to CC-PE1 (iBGP VPNv4)

```
router bgp 100
 neighbor CC-PE1 remote-as 100
 neighbor CC-PE1 update-source Loopback0
 !
 address-family vpnv4
  neighbor CC-PE1 activate
  neighbor CC-PE1 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf C-VPN
  redistribute isis 200
 exit-address-family
```

IOS-XR configuration is similar to IOS, with the major difference of using the labeled unicast address-family instead of the send-label keyword.

Don't forget to create a /32 static route for the CsC-PE/CE next-hop in IOS-XR when using eBGP+Label. **Always verify the installation of labels for /32 next-hops.**

Verification

- Customer Carrier PEs must have a BGP VPNv4 route and a label for the VPN prefix
- Customer Carrier routers must have a label for the VPN prefix's next-hop
- CsC-PEs must have a BGP VPNv4 route and a label for the VPN prefix's next-hop
- Backbone Carrier routers must have a label for the next-hop of VPN prefix's next-hop

---

Example

Assume the following network:

R1-R2-R3-R4-R5-R6-R7-R8-R9-R10

where

Customer Carrier Network
Backbone Carrier Network

Then the following would happen for a VPN packet originating at R1 and terminating at R10.

- R1 (1.1.1.1) (Customer Carrier PE router) - vrf VPN
  - Transport label is 18, VPN label is 20
  - next-hop is R10 (10.10.10.10)
- R2 (2.2.2.2) (Customer Carrier P router)
  - Transport label is 20, VPN label is 20
  - next-hop is R10 (10.10.10.10)
- R3 (3.3.3.3) (CsC-CE)
  - Transport label is 26, VPN label is 20
  - next-hop is R10 (10.10.10.10)
- R4 (4.4.4.4) (CsC-PE) - vrf CSC
  - Transport label is 16/21, VPN label is 20
  - next-hop is R7 (7.7.7.7)
- R5 (5.5.5.5) Backbone Carrier P router)
  - Transport label is 16/21, VPN label is 20
  - next-hop is R7 (7.7.7.7)
- R6 (6.6.6.6) (Backbone Carrier P router)
  - Transport label is 21, VPN label is 20
  - next-hop is R7 (7.7.7.7)
- R7 (7.7.7.7) (CsC-PE) - vrf CSC

  - Transport label is 18, VPN label is 20
  - next-hop is R10 (10.10.10.10)
- R8 (8.8.8.8) (CsC-CE)
  - Transport label is 17, VPN label is 20
  - next-hop is R10 (10.10.10.10)
- R9 (9.9.9.9) (Customer Carrier P router)
  - Transport label is removed, VPN label is 20
  - next-hop is R10 (10.10.10.10)
- R10 (10.10.10.10) (Customer Carrier PE router) - vrf VPN
  - VPN label is removed, destination reached in next-hop

```
R1#trace vrf VPN 99.99.99.99

Type escape sequence to abort.
Tracing the route to 99.99.99.99

  1 20.1.2.2 [MPLS: Labels 18/20 Exp 0] 10 msec 10 msec 8 msec
  2 20.2.3.3 [MPLS: Labels 20/20 Exp 0] 7 msec 7 msec 7 msec
  3 20.3.4.4 [MPLS: Labels 26/20 Exp 0] 8 msec 8 msec 7 msec
  4 20.4.5.5 [MPLS: Labels 16/21/20 Exp 0] 7 msec 7 msec 8 msec
  5 20.5.6.6 [MPLS: Labels 16/21/20 Exp 0] 6 msec 5 msec 5 msec
  6 20.6.7.7 [MPLS: Labels 21/20 Exp 0] 3 msec 3 msec 3 msec
  7 20.7.8.8 [MPLS: Labels 18/20 Exp 0] 1 msec 1 msec 1 msec
  8 20.8.9.9 [MPLS: Labels 17/20 Exp 0] 1 msec 1 msec 1 msec
  9 20.9.10.10 [MPLS: Label 20 Exp 0] 1 msec 1 msec 1 msec
 10 30.10.10.99 1 msec 2 msec 2 msec
```

Verification in every hop

Customer Carrier PE router

```
R1#sh ip route vrf VPN 99.99.99.99

Routing Table: VPN
Routing entry for 99.99.99.99/32
  Known via "bgp 100", distance 200, metric 1, type internal
  Redistributing via rip
  Advertised by rip metric transparent
  Last update from 10.10.10.10 00:56:37 ago
  Routing Descriptor Blocks:
  * 10.10.10.10 (default), from 10.10.10.10, 00:56:37 ago
      Route metric is 1, traffic share count is 1
      AS Hops 0
      MPLS label: 20
      MPLS Flags: MPLS Required

R1#sh bgp vpnv4 unicast vrf VPN 99.99.99.99/32
```

```
BGP routing table entry for 100:1:99.99.99.99/32, version 11
Paths: (1 available, best #1, table VPN)
  Not advertised to any peer
  Local
    10.10.10.10 (metric 20) from 10.10.10.10 (10.10.10.10)
      Origin incomplete, metric 1, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      mpls labels in/out nolabel/20
```

VPN label is 20

```
R1#sh mpls forwarding-table 10.10.10.10 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
23         18         10.10.10.10/32   0             Fa0/0.12   20.1.2.2
        MAC/Encaps=18/22, MRU=1500, Label Stack{18}
        CA0113DC0000CA030794000881000 26C8847 00012000
        No output feature configured
```

```
R1#sh ip cef vrf VPN 99.99.99.99 det99.99.99.99/32, epoch 0, flags rib
defined all labels
  recursive via 10.10.10.10 label 20
    nexthop 20.1.2.2 FastEthernet0/0.12 label 18
```

Transport label is 18, VPN label is 20

From here you have 2 options:

- follow the next-hops
- follow the labels (i prefer this one)

Following the next-hops

Customer Carrier P router

```
R2#sh mpls forwarding-table 10.10.10.10 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
18         20         10.10.10.10/32   13707         Fa0/0.23   20.2.3.3
        MAC/Encaps=18/22, MRU=1500, Label Stack{20}
        CA0610240000CA0113DC00008100002E8847 00014000
        No output feature configured
```

Transport label is 20, VPN label is 20

CsC-CE

```
R3#sh mpls forwarding-table 10.10.10.10 detail
Local        Outgoing    Prefix            Bytes Label   Outgoing   Next Hop
Label        Label       or Tunnel Id      Switched      interface
20           26          10.10.10.10/32    15048         Fa0/0.34   20.3.4.4
        MAC/Encaps=18/22, MRU=1500, Label Stack{26}
        CA0207940008CA06102400008100001A38847 0001A000
        No output feature configured
```

Transport label is 26, VPN label is 20

CsC-PE

```
R4#sh mpls forwarding-table vrf CSC 10.10.10.10 detail
Local        Outgoing    Prefix            Bytes Label   Outgoing   Next Hop
Label        Label       or Tunnel Id      Switched      interface
26           21          10.10.10.10/32[V]16033         Fa0/0.45   20.4.5.5
        MAC/Encaps=18/26, MRU=1496, Label Stack{16 21}
        C20911080000CA0207940008810003338847 0001000000015000
        VPN route: CSC
        No output feature configured
```

2 Transport labels are used (use "`detail`" to see them)

```
R4#sh bgp vpnv4 unicast vrf CSC 10.10.10.10
BGP routing table entry for 200:1:10.10.10.10/32, version 12
Paths: (1 available, best #1, table CSC)
  Advertised to update-groups:
     3
  100
    7.7.7.7 (metric 4) from 7.7.7.7 (7.7.7.7)
      Origin incomplete, metric 20, localpref 100, valid, internal, best
      Extended Community: RT:200:1
      mpls labels in/out 26/21
```

VPN label (21) for Backbone Carrier is actually Transport label (21) for Customer Carrier

```
R4#sh mpls forwarding-table 7.7.7.7 detail
Local        Outgoing    Prefix            Bytes Label   Outgoing   Next Hop
Label        Label       or Tunnel Id      Switched      interface
16           16          7.7.7.7/32        0             Fa0/0.45   20.4.5.5
        MAC/Encaps=18/22, MRU=1500, Label Stack{16}
        C20911080000CA0207940008810003338847 00010000
        No output feature configured
```

Transport label is 16/21, VPN label is 20

Backbone Carrier P router

```
R5#sh mpls forwarding-table 7.7.7.7 detail
Local   Outgoing    Prefix            Bytes tag  Outgoing    Next Hop
```

```
tag    tag or VC   or Tunnel Id      switched   interface
16     16          7.7.7.7/32        44218      Fa0/0.56   20.5.6.6
        MAC/Encaps=18/22, MRU=1500, Tag Stack{16}
        C20811080000C20911080000810004E8847 00010000
        No output feature configured
    Per-packet load-sharing
```

Transport label is 16/21, VPN label is 20

Backbone Carrier P router

```
R6#sh mpls forwarding-table 7.7.7.7 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
16     Pop tag     7.7.7.7/32        42398      Fa0/0.67   20.6.7.7
        MAC/Encaps=18/18, MRU=1504, Tag Stack{}
        CA0415180000C20811080000810000118847
        No output feature configured
    Per-packet load-sharing
```

Transport label is 21, VPN label is 20

CsC-PE

```
R7#sh mpls forwarding-table vrf CSC 10.10.10.10 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
21         18         10.10.10.10/32[V]24156         Fa0/0.78   20.7.8.8
        MAC/Encaps=18/22, MRU=1500, Label Stack{18}
        CA0710240000CA041518000081000000D8847 00012000
        VPN route: CSC
        No output feature configured
```

Transport label is 18, VPN label is 20

CsC-CE

```
R8#sh mpls forwarding-table 10.10.10.10 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
18         17         10.10.10.10/32   24703         Fa0/0.89   20.8.9.9
        MAC/Encaps=18/22, MRU=1500, Label Stack{17}
        CA0013DC0000CA071024000081000238847 00011000
        No output feature configured
```

Transport label is 17, VPN label is 20

Customer Carrier P router

```
R9#sh mpls forwarding-table 10.10.10.10 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
17         Pop Label  10.10.10.10/32   23892         Fa0/0.910
20.9.10.10
```

```
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA0515180000CA0013DC000081000198847
        No output feature configured
```

Transport label is removed, VPN label is 20

Customer Carrier PE router

```
R10#sh mpls forwarding-table vrf VPN 99.99.99.99 detail
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label       or Tunnel Id     Switched      interface
20         No Label    99.99.99.99/32[V]1770          Fa0/0.1010
30.10.10.99
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        C20A0F840000CA05151800008100001D0800
        VPN route: VPN
        No output feature configured
```

VPN label is removed, destination reached

Following the labels

Customer Carrier P router

```
R2#sh mpls forwarding-table labels 18 detail
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label       or Tunnel Id     Switched      interface
18         20          10.10.10.10/32   13852         Fa0/0.23   20.4.6.4
        MAC/Encaps=18/22, MRU=1500, Label Stack{20}
        CA0610240000CA0113DC00008100002E8847 00014000
        No output feature configured
```

Transport label is 20, VPN label is 20

CsC-CE

```
R3#sh mpls forwarding-table labels 20 detail
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label       or Tunnel Id     Switched      interface
20         26          10.10.10.10/32   15338         Fa0/0.34
20.4.19.19
        MAC/Encaps=18/22, MRU=1500, Label Stack{26}
        CA0207940008CA06102400008100001A38847 0001A000
        No output feature configured
```

Transport label is 26, VPN label is 20

CsC-PE

```
R4#sh mpls forwarding-table labels 26 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
26         21         10.10.10.10/32[V]16645         Fa0/0.45   20.4.5.5
        MAC/Encaps=18/26, MRU=1496, Label Stack{16 21}
        C20911080000CA0207940008810003338847 0001000000015000
        VPN route: CSC
        No output feature configured
```

2 Transport labels are used (use "`detail`" to see them)

Transport label is 16/21, VPN label is 20

Backbone Carrier P router

```
R5#sh mpls forwarding-table labels 16 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
16     16          7.7.7.7/32        30421      Fa0/0.56   20.5.6.6
        MAC/Encaps=18/22, MRU=1500, Tag Stack{16}
        C20811080000C20911080000810004E8847 00010000
        No output feature configured
    Per-packet load-sharing
```

Transport label is 16/21, VPN label is 20

Backbone Carrier P router

```
R6#sh mpls forwarding-table labels 16 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
16     Pop tag     7.7.7.7/32        29337      Fa0/0.67   20.6.7.7
        MAC/Encaps=18/18, MRU=1504, Tag Stack{}
        CA0415180000C2081108000081000118847
        No output feature configured
    Per-packet load-sharing
```

Transport label is 21, VPN label is 20

CsC-PE

```
R7#sh mpls forwarding-table labels 21 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
21         18         10.10.10.10/32[V]17260         Fa0/0.78   20.7.8.8
        MAC/Encaps=18/22, MRU=1500, Label Stack{18}
        CA0710240000CA041518000081000D8847 00012000
        VPN route: CSC
        No output feature configured
```

Transport label is 18, VPN label is 20

CsC-CE

```
R8#sh mpls forwarding-table labels 18 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
18         17         10.10.10.10/32   17726         Fa0/0.89   20.8.9.9
        MAC/Encaps=18/22, MRU=1500, Label Stack{17}
        CA0013DC0000CA07102400008100002388847 00011000
        No output feature configured
```

Transport label is 17, VPN label is 20

Customer Carrier P router

```
R9#sh mpls forwarding-table labels 17 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
17         Pop Label  10.10.10.10/32   17200         Fa0/0.910
20.9.10.10
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA0515180000CA0013DC000081000198847
        No output feature configured
```

Transport label is removed, VPN label is 20

Customer Carrier PE router

```
R10#sh mpls forwarding-table labels 20 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
20         No Label   99.99.99.99/32[V]1770          Fa0/0.1010
30.10.10.99
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        C20A0F840000CA05151800008100001D0800
        VPN route: VPN
        No output feature configured
```

VPN label is removed, destination reached

6PE/6VPE

6PE is defined in RFC 4798.
6VPE is defined in RFC 4659.

- 6PE
  - customer's IPv6 prefixes are inside the global routing table
  - IPv6 labels/prefixes are exchanged using Labeled IPv6 over IPv4 iBGP between the PEs
- 6VPE
  - customer's IPv6 prefixes are inside a VRF
  - IPv6 labels/prefixes are exchanged using VPNv6 over IPv4 iBGP between the PEs

6PE

6PE is a technology that allows IPv6 customers to communicate with each other over an IPv4 MPLS Provider without any tunnel setup, by **having the customer IPv6 prefixes using a IPv4-mapped IPv6 address as next-hop inside the Provider's network** and using IPv4 LSPs between the 6PEs.

In 6PE, labels must be exchanged between the 6PEs for their IPv6 prefixes, which means that **a labeled IPv4 iBGP session must be activated under the IPv6 address family** (in IOS) or the **labeled IPv6 capability must be activated for the IPv4 peer 6PE** (in IOS-XR).

IOS

```
router bgp 100
 no bgp default ipv4-unicast
 neighbor 6PE-IPv4 remote-as 100
 neighbor 6PE-IPv4 update-source Loopback0
 neighbor CE-IPv6 remote-as X
!
 address-family ipv6
  neighbor 6PE-IPv4 activate
  neighbor 6PE-IPv4 send-label
  neighbor CE-IPv6 activate
```

IOS-XR

```
router bgp 100
 address-family ipv6 unicast
  allocate-label all
 !
 neighbor 6PE-IPv4
  remote-as 100
  update-source Loopback0
  address-family ipv6 labeled-unicast
  !
 neighbor CE-IPv6
  remote-as X
  address-family ipv6 unicast
```

The 6PE routers are dual-stack: IPv6 towards the CE and IPv4 towards the MPLS core.

You don't need to set the "`next-hop-self`" in the labeled IPv4 iBGP session for the IPv6 prefixes, because that happens automatically while creating the IPv4-mapped IPv6 address.

<u>MP_REACH_NLRI attribute</u>

- AFI = 2 (IPv6)
- SAFI= 4 (labels)
- Prefix =X:X:X:X::X
- Label = X
- next-hop = ::FFFF:IPv4-ADDRESS

The IPv4 address which is encoded into the next-hop must be present in the IPv4 routing table and a LSP must exist to this destination. If the above conditions are not both met, the IPv6 prefix is declared inaccessible.

```
R5#sh bgp ipv6 unicast 2001::1:1:1:1/128
BGP routing table entry for 2001::1:1:1:1/128, version 4
Paths: (1 available, no best path)
  Not advertised to any peer
  1
    ::FFFF:2.2.2.2 (inaccessible) from 2.2.2.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal
```

If a 6PE receives unlabeled IPv6 prefixes, then the 6PE marks these prefixes as unreachable in the IPv6 routing table, so that packets to this destination get dropped and not sent into MPLS core.

If RRs are to be used for 6PE connectivity, then they also must have labeled BGP enabled.

Control Plane

- CEs and 6PEs run eBGP (or IGP) in order to exchange IPv6 prefixes
- 6PEs run iBGP+Label in order to exchange labels and IPv6 prefixes
- Provider routers run IGP+LDP in order to exchange all their internal BGP IPv4 next-hops and their labels

6PE Example

Assume the following network:

R1-R2-R3-R4-R5-R6

where

IPv6 Customer
6PE Provider (with 6PE routers and IPv4 routers)

Then the following would happen for an IPv6 packet originating at R1 and terminating at R6.

- R1 (IPv6)
  - next-hop is R2 (IPv6 link-local)
- R2 (IPv4/IPv6 6PE)
  - next-hop is R5 (IPv4 loopback)
  - Transport label is 26, IPv6 label is 28
- R3 (IPv4)
  - next-hop is R5 (IPv4 loopback)
  - Transport label is 25, IPv6 label is 28
- R4 (IPv4)
  - next-hop is R5 (IPv4 loopback)
  - Transport label is removed, IPv6 label is 28
- R5 (IPv4/IPv6 6PE)
  - next-hop is R6 (IPv6 link-local)
  - IPv6 label is removed, IPv6 destination reached in next-hop
- R6 (IPv6)
  - IPv6 destination reached

The IPv6 label in 6PE can be considered like the VPN label in MPLS VPN.

The customer routers could be IPv4/IPv6, having IPv4 traffic being forwarded as usual and IPv6 traffic over 6PE.

Use extended traceroute if you want to source from an IPv6 address.

```
R1#trace
Protocol [ip]: ipv6
Target IPv6 address: 2001::6:6:6:6
Source address: 2001::1:1:1:1
...
Tracing the route to 2001::6:6:6:6

  1 2001:10:1:2::2 44 msec 72 msec 32 msec
  2 ::FFFF:10.2.3.3 [MPLS: Labels 26/28 Exp 0] 1 msec 1 msec 1 msec
  3 ::FFFF:10.3.4.4 [MPLS: Labels 25/28 Exp 0] 1 msec 1 msec 1 msec
  4 2001:10:5:6::5[MPLS: Label 28 Exp 0] 1 msec 1 msec  1 msec
  5 2001:10:5:6::6 1 msec 1 msec 1 msec
```

Although all traceroute answers are given by the closest interface to the source ip, in some software releases in the last 6PE router the traceroute answer is given by the IPv6 address on the PE-CE link.

You can use "`no mpls ip propagate-ttl`" on the first-hop 6PE if you want to hide the ::FFFF: IPv4 hops.

```
  1 2001:10:1:2::2 44 msec 72 msec 32 msec
  2 2001:10:5:6::5[MPLS: Label 28 Exp 0] 1 msec 1 msec  1 msec
  3 2001:10:5:6::6 1 msec 1 msec 1 msec
```

Verification in every hop

Customer IPv6 router

```
R1#sh ipv6 route 2001::6:6:6:6
Routing entry for 2001::6:6:6:6/128
  Known via "bgp 1", distance 20, metric 0, type external
  Route count is 1/1, share count 0
  Routing paths:
    FE80::C805:17FF:FEC8:1C, FastEthernet0/0
      MPLS label: nolabel
      Last updated 00:31:05 ago
```

Provider 6PE router

```
R2#sh bgp ipv6 unicast 2001::6:6:6:6/128
BGP routing table entry for 2001::6:6:6:6/128, version 3
Paths: (1 available, best #1, table default)
  Advertised to update-groups:
     1
  20
    ::FFFF:5.5.5.5(metric 4) from 5.5.5.5(5.5.5.5)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      mpls labels in/out nolabel/28
```

IOS-XR doesn't display the "::FFFF:" in front of the IPv4 next-hop (How Multi is MP-BGP in IOS-XR - Part #2).

IPv6 label is 28

```
R2#sh mpls forwarding-table 5.5.5.5 detail
Local      Outgoing    Prefix            Bytes Label   Outgoing    Next Hop
Label      Label       or Tunnel Id      Switched      interface
27         26          5.5.5.5/32        0             Fa0/0.23    10.2.3.3
        MAC/Encaps=18/22, MRU=1500, Label Stack{26}
        CA071AC40000CA0517C80000810000178847 0001A000
        No output feature configured
```

Transport label is 26

```
R2#sh ipv6 cef 2001::6:6:6:6/128 detail
2001::6:6:6:6/128, epoch 0, flags rib defined all labels
  recursive via 5.5.5.5 label 28
    nexthop 10.2.3.3 FastEthernet0/0.23 label 26
```

Transport label is 26, IPv6 label is 28

You can follow the next-hop or the labels (like in CsC), but let's follow the next-hop on this example.

Provider IPv4 Router

```
R3#sh mpls forwarding-table 5.5.5.5 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
26         25         5.5.5.5/32       16342         Fa0/0.34   10.3.4.4
        MAC/Encaps=18/22, MRU=1500, Label Stack{25}
        CA0113200000CA071AC40000810000248847 00019000
        No output feature configured
```

Transport label is 25, IPv6 label is 28

Provider IPv4 Router

```
R4#sh mpls forwarding-table 5.5.5.5 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
25         Pop Label  5.5.5.5/32       16125         Fa0/0.45 10.4.5.5
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA0214D00008CA0113200000810026B8847
        No output feature configured
```

Transport label is removed, IPv6 label is 28

Provider 6PE router

```
R5#sh mpls forwarding-table 2001::6:6:6:6/128 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
28         No Label   2001::6:6:6:6/128   \
                                       2244          PO2/0
point2point
        MAC/Encaps=4/4, MRU=4474, Label Stack{}
        0F0086DD
        No output feature configured
```

```
R5#sh bgp ipv6 unicast 2001::6:6:6:6/128
BGP routing table entry for 2001::6:6:6:6/128, version 3
Paths: (1 available, best #1, table default)
  Advertised to update-groups:
     2
  20
    2001:10:5:6::6 (FE80::C803:14FF:FED0:8) from 2001:10:5:6::6 (6.6.6.6)
      Origin IGP, metric 0, localpref 100, valid, external, best
      mpls labels in/out 28/nolabel

R5#sh bgp ipv6 unicast labels
   Network          Next Hop      In label/Out label
   2001::6:6:6:6/128
                    2001:10:5:6::6
                                    28/nolabel

R5#sh ipv6 route 2001::6:6:6:6/128
Routing entry for 2001::6:6:6:6/128
  Known via "bgp 100", distance 20, metric 0, type external
  Route count is 1/1, share count 0
  Routing paths:
    FE80::C803:14FF:FED0:8, POS2/0
      MPLS label: nolabel
      Last updated 00:43:37 ago
```

IPv6 label is removed, IPv6 destination reached in next-hop

6VPE

6VPE is a technology that allows IPv6 VPN customers to communicate with each other over an IPv4 MPLS Provider without any tunnel setup, by **having the customer VPNv6 prefixes using a v4-mapped IPv6 address as next-hop inside the provider's network** and using IPv4 LSPs between the 6VPEs.

In 6VPE, labels must be exchanged between the 6VPEs for their VPNv6 prefixes, which means that **the VPNv6 address-family must be activated on the IPv4 iBGP session between the 6VPEs**.

IOS

```
router bgp 100
 no bgp default ipv4-unicast
 neighbor 6VPE-IPv4 remote-as 100
 neighbor 6VPE-IPv4 update-source Loopback0
 !
 address-family vpnv6
  neighbor 6VPE-IPv4 activate
  neighbor 6VPE-IPv4 send-community extended
 exit-address-family
 !
 address-family ipv6 vrf VPN
```

```
  neighbor CE-IPv6 remote-as X
  neighbor CE-IPv6 activate
 exit-address-family
```

IOS-XR

```
router bgp 100
 address-family vpnv6 unicast
 !
 neighbor 6VPE-IPv4
  remote-as 100
  update-source Loopback0
  address-family vpnv6 unicast
  !
 vrf VPN
  rd 100:1
  address-family ipv6 unicast
  !
  neighbor CE-IPv6
   remote-as x
   address-family ipv6 unicast
```

The 6VPE routers are dual-stack: IPv6 towards the CE (inside a VRF) and IPv4 towards the MPLS core.

MP_REACH_NLRI attribute

- AFI = 2 (IPv6)
- SAFI= 128 (.)
- Prefix =X:X:X:X::X
- Label = X
- next-hop = ::FFFF:IPv4-ADDRESS

The IPv4 address which is encoded into the next-hop must be present in the IPv4 routing table and a LSP must exist to this destination. If the above conditions are not both met, the IPv6 prefix is declared inaccessible.

If RRs are to be used for 6VPE connectivity, then they also must have the VPNv6 address-family enabled.

Control Plane

- CEs and 6VPEs run eBGP (or IGP) in order to exchange IPv6 prefixes
- 6VPEs run VPNv6 iBGP in order to exchange labels and VPNv6 prefixes
- Provider routers run IGP+LDP in order to exchange all their internal BGP IPv4 next-hops and their labels

In IOS, if you enable the VPNv6 address-family on an existing VPNv4 peering, you might need to reset the whole BGP session for VPNv6 to come up.

6VPE Example

Assume the following network:

R1-R2-R3-R4-R5-R6

where

IPv6 VPN Customer
6VPE Provider (with 6VPE routers and IPv4 routers)

Then the following would happen for an IPv6 packet originating at R1 and terminating at R6.

- R1 (IPv6)
  - next-hop is R2 (IPv6 link-local)
- R2 (IPv4/IPv6 6VPE)
  - next-hop is R5 (IPv4 loopback)
  - Transport label is 18, VPNv6 label is 28
- R3 (IPv4)
  - next-hop is R5 (IPv4 loopback)
  - Transport label is 25, VPNv6 label is 28
- R4 (IPv4)
  - next-hop is R5 (IPv4 loopback)
  - Transport label is removed, VPNv6 label is 28
- R5 (IPv4/IPv6 6VPE)
  - next-hop is R6 (IPv6 link-local)
  - VPNv6 label is removed, IPv6 destination reached in next-hop
- R6 (IPv6)
  - IPv6 destination reached

Use extended traceroute if you want to source from a specific IPv6 address.

```
R1#trace 2001:6:6:6::6

Type escape sequence to abort.
Tracing the route to 2001:6:6:6::6
  1 2001:10:1:2::2 1 msec 1 msec 1 msec
  2 ::FFFF:10.2.3.3 [MPLS: Labels 18/28 Exp 0] 1 msec 1 msec 1 msec
  3 ::FFFF:10.3.4.4 [MPLS: Labels 25/28 Exp 0] 1 msec 1 msec 1 msec
  4 2001:10:5:6::5 [AS 100] [MPLS: Label 28 Exp 0] 1 msec 1 msec 1 msec
  5 2001:10:5:6::6 [AS 100] 1 msec 1 msec 1 msec
```

Although all traceroute answers are given by the closest interface to the source ip, in some software releases in the last 6VPE router the traceroute answer is given by the IPv6 address on the PE-CE link.

You can use "`no mpls ip propagate-ttl`" on the first-hop 6VPE if you want to hide the ::FFFF: IPv4 hops.

```
  1 2001:10:1:2::2 1 msec 1 msec 1 msec
  2 2001:10:5:6::5 [AS 100] [MPLS: Label 28 Exp 0] 1 msec 1 msec 1 msec
  3 2001:10:5:6::6 [AS 100] 1 msec 1 msec 1 msec
```

Verification in every hop

Customer IPv6 router

```
R1#sh ipv6 route 2001::6:6:6:6
Routing entry for 2001::6:6:6:6/128
  Known via "bgp 1", distance 20, metric 0, type external
  Route count is 1/1, share count 0
  Routing paths:
    FE80::C805:EFF:FE2C:1C, FastEthernet0/0
      MPLS label: nolabel
      Last updated 01:47:39 ago
```

Provider 6VPE router

```
R2#sh bgp vpnv6 unicast vrf VPN 2001::6:6:6:6/128
BGP routing table entry for [100:1]2001::6:6:6:6/128, version 6
Paths: (1 available, best #1, table VPN)
  Advertised to update-groups:
     3
  1
    ::FFFF:5.5.5.5 (metric 4) from 5.5.5.5 (5.5.5.5)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      mpls labels in/out nolabel/28
```

IOS-XR doesn't display the "::FFFF:" in front of the IPv4 next-hop (How Multi is MP-BGP in IOS-XR - Part #2).

VPNv6 label is 28

```
R2#sh mpls forwarding-table 5.5.5.5 detail
Local      Outgoing    Prefix           Bytes Label   Outgoing   Next Hop
Label      Label       or Tunnel Id     Switched      interface
19         18          5.5.5.5/32       0             Fa0/0.23   10.2.3.3
        MAC/Encaps=18/22, MRU=1500, Label Stack{18}
        CA070D4C0000CA050E2C00008100001788847 00012000
        No output feature configured
```

Transport label is 18

```
R2#sh ipv6 cef vrf VPN 2001::6:6:6:6 detail
2001::6:6:6:6/128, epoch 0, flags rib defined all labels
```

```
  recursive via 5.5.5.5 label 28
    nexthop 10.2.3.3 FastEthernet0/0.23 label 18
```

Transport label is 18, VPNv6 label is 28

You can follow the next-hop or the labels (like in CsC), but for easiness let's follow the labels this time.

Provider IPv4 Router

```
R3#sh mpls forwarding-table labels 18 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
18         25         5.5.5.5/32       45876         Fa0/0.34   10.3.4.4
        MAC/Encaps=18/22, MRU=1500, Label Stack{25}
        CA0119D80000CA070D4C000081000024 8847 00019000
        No output feature configured
```

Transport label is 25, VPNv6 label is 28

Provider IPv4 Router

```
R4#sh mpls forwarding-table labels 25 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
25         Pop Label  5.5.5.5/32       45343         Fa0/0.45   10.4.5.5
        MAC/Encaps=18/18, MRU=1504, Label Stack{}
        CA0217AC0008CA0119D8000081000 26B8847
        No output feature configured
```

Transport label is removed, VPNv6 label is 28

Provider 6VPE router

```
R5#sh mpls forwarding-table labels 28 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
28         No Label   2001::6:6:6:6/128[V]   \
                                       2820          PO2/0
point2point
        MAC/Encaps=4/4, MRU=4474, Label Stack{}
        0F0086DD
        VPN route: VPN
        No output feature configured

R5#sh bgp vpnv6 unicast vrf A 2001::6:6:6:6/128
BGP routing table entry for [100:1]2001::6:6:6:6/128, version 6
Paths: (1 available, best #1, table VPN)
```

```
  Advertised to update-groups:
     1
  1
    2001:10:5:6::6 (FE80::C803:17FF:FEAC:8) from 2001:10:5:6::6 (6.6.6.6)
      Origin incomplete, metric 0, localpref 100, valid, external, best
      Extended Community: RT:100:1
      mpls labels in/out 28/nolabel
```

Use prefix/size to verify when doing bgp lookups.

```
R5#sh ipv6 route vrf VPN 2001::20:20:20:20/128
Routing entry for 2001::6:6:6:6/128
  Known via "bgp 100", distance 20, metric 0, type external
  Route count is 1/1, share count 0
  Routing paths:
    FE80::C803:17FF:FEAC:8, POS2/0
      MPLS label: nolabel
      Last updated 02:08:07 ago
```

VPNv6 label is removed, IPv6 destination reached in next-hop

IPv6 prefixes over an IPv4 PE-CE BGP session

Usually when doing 6PE or 6VPE, the BGP session between the PE and the CE (which is used for exchanging the IPv6 prefixes) is IPv6 based.

If you want to exchange **IPv6 prefixes over an IPv4 PE-CE BGP session**, then you must **change the IPv6 next-hop** in the PEs in both directions (and then reset the session) Otherwise you might have the control plane showing everything is fine, but most probably data plane won't work.

IOS

```
R2#sh bgp vpnv6 unicast all
BGP table version is 1, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 26:65001 (default for vrf VPN)
*  2001:10:2:8::/64 ::FFFF:10.2.8.8          0             0 65001 i


R2#sh bgp vpnv6 unicast all 2001:10:2:8::/64
BGP routing table entry for [26:65001]2001:10:2:8::/64, version 0
```

```
Paths: (1 available, no best path)
  Not advertised to any peer
  65001
    ::FFFF:10.2.8.8 (inaccessible) from 10.2.8.8 (10.8.8.8)
      Origin IGP, metric 0, localpref 100, valid, external
      Extended Community: RT:26:65001
```

IOS

```
router bgp X
 address-family ipv6 vrf VPN
  neighbor CE-IPv4 route-map SET-IPV6NH-IN-ROUTEMAP in
  neighbor CE-IPv4 route-map SET-IPV6NH-OUT-ROUTEMAP out
 exit-address-family
!
route-map SET-IPV6NH-IN-ROUTEMAP permit 10
 set ipv6 next-hop CE-IPv6
!
route-map SET-IPV6NH-OUT-ROUTEMAP permit 10
 set ipv6 next-hop PE-IPv6
```

In general:

- PE=>CE (out)
  - `set ipv6 next-hop PE-IPv6`
- PE<=CE (in)
  - `set ipv6 next-hop CE-IPv6`

After applying the route-map and resetting the session:

IOS

```
R2#sh bgp vpnv6 unicast all 2001:10:2:8::/64
BGP routing table entry for [26:65001]2001:10:2:8::/64, version 14
Paths: (1 available, best #1, table VPN)
  Advertised to update-groups:
     4
  65001
    2001:10:2:8::8 from 10.2.8.8 (10.8.8.8)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Extended Community: RT:26:65001
      mpls labels in/out 27/nolabel
```

In IOS-XR, you won't even get an IPv4 BGP session up, if there are no IPv6 addresses configured on the PE-CE connection.

The default order of IPv6 next-hop operation is the following:

- use the IPv6 address defined in the route-map of the session
- use the IPv6 address used for peering (directly connected interface or loopback)

- use an IPv4-mapped IPv6 address

"no bgp default ipv6-nexthop" is also an option.

Also, **sometimes in the CE you must enable ebgp-multihop for the PE session (2 hops at least)**, otherwise the remote IPv6 routes won't be installed in the CE's BGP table and you'll be getting messages "DENIED due to: non-connected MP_REACH NEXTHOP" (enable "debug bgp ipv6 unicast updates" to see these). This is probably a **bug**, since the next-hop (PE) is actually connected to the CE router.

A correct output of "sh bgp vpnv6 unicast" from a PE will include:

- IPv6 routes from the directly connected CE
  - The next hop shown must be the directly connected CE IPv6 address (i.e. 2001:10:2:8::8)
- IPv6 routes from other PEs
  - The next hop shown must be the remote PE IPv4 address (in v4-mapped IPv6 address format like.::FFFF:1.1.1.1 in IOS, or just 1.1.1.1 in IOS-XR)

Example:

IOS

```
R2#sh bgp vpnv6 unicast all
BGP table version is 23, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 26:65001 (default for vrf X)
*>i2001:10:1:7::/64 ::FFFF:1.1.1.1           0    100      0 65001 i
*> 2001:10:2:8::/64 2001:10:2:8::8           0             0 65001 i
```

IOS-XR

```
GSR#sh bgp vpnv6 unicast
BGP router identifier 19.19.19.19, local AS number 26
...
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 26:65001 (default for vrf VPN)
*>i2001:10:1:7::/64   1.1.1.1                  0    100      0 65001 i
*> 2001:10:20:20::/64 2001:10:19:20::20
                               0             0 65001 i
```

When doing the opposite and trying to exchange IPv4 prefixes over an IPv6 BGP session, you have to set manually the IPv4 next-hop

IOS-XR requires to enable the VPNv6 address-family before enabling the IPv6 address-family under a VRF.

Hints

If you get issues with IPv6 next-hops being inaccessible when they point to eBGP interfaces (while they shouldn't), try to shut/no shut the interface.

Before

```
R2#sh bgp ipv6 unicast 2001:10:1:7::/64
BGP routing table entry for 2001:10:1:7::/64, version 0
Paths: (1 available, no best path)
  Not advertised to any peer
  26 26
    2001:10:19:20::19 (FE80::C802:15FF:FE18:8) (inaccessible) from
10.19.20.19 (19.19.19.19)
      Origin IGP, localpref 100, valid, external
```

After

```
R2#sh bgp ipv6 unicast 2001:10:1:7::/64
BGP routing table entry for 2001:10:1:7::/64, version 6
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  26 26
    2001:10:19:20::19 from 10.19.20.19 (19.19.19.19)
      Origin IGP, localpref 100, valid, external, best
```

This is probably another **bug**.

ATOM/L2VPN/VPLS

PW (PseudoWire) architecture is described in RFC 3985.
LDP for PWs is defined in RFC 4447.
EoMPLS (Ethernet over MPLS) is defined in RFC 4448.
L2TPv3 (Layer 2 Tunneling Protocol v3) is defined in RFC 3931.

L2VPN

- AToM
    - requires MPLS connectivity from end to end
    - source ip address is the one used for LDP
- L2TPv3
    - requires IP connectivity from end to end
    - source ip address is configured manually

If you are transferring Vlans and STP packets over the L2VPN connection, then you might need to disable STP for those specific vlans when using any type of PVST. Alternatively you can try to match vlans on either side.

L2TPv3

L2TPv3 is protocol 115.

Supported protocols over L2TPv3

- Frame-Relay
- Ethernet
- VLAN
- HDLC
- PPP
- IPv6

The L2TPv3 keepalive mechanism consists of an exchange of L2TP hello messages.

The interface keepalive feature is automatically disabled on the interface to which xconnect is applied, except for Frame-Relay encapsulation.

The IP local interface must be a loopback interface.

To convert an interface with AToM xconnect to L2TPv3 xconnect, remove the xconnect configuration from the interface and then configure L2TPv3.

Each L2TPv3 tunneled packet includes the entire Layer 2 frame of the payload:

- 20 bytes (IP packet header)
- 12 bytes (L2TPv3 session header)
- 4 bytes (if sequencing is enabled)
- payload (the complete frame received on the AC)

You can have either **dynamic or static pseudowires** (where no signaling is required for session establishment).

IOS

```
pseudowire-class L2TP-PWCLASS
 encapsulation l2tpv3
 ip local interface Loopback0
!
interface X
 xconnect 2.2.2.2 12 pw-class L2TP-PWCLASS
```

IOS-XR

```
l2vpn
 pw-class L2TP-PWCLASS
  encapsulation l2tpv3
   protocol l2tpv3
   ipv4 source 2.2.2.2
 !
 xconnect group R2-R1-XCONGRP
  p2p R2-R1-P2P
   interface X
   neighbor 1.1.1.1 pw-id 12
    pw-class L2TP-PWCLASS
```

In IOS-XR, you have to define the signaling protocol, while in IOS it's enabled by default.

IOS

```
R6#sh l2tun tunnel all

L2TP Tunnel Information Total tunnels 1 sessions 1

Tunnel id 1787404852 is up, remote id is 3119273419, 1 active sessions
  Remotely initiated tunnel
  Tunnel state is established, time since change 00:00:14
  Tunnel transport is IP  (115)
  Remote tunnel name is R2
    Internet Address 2.2.2.2, port 0
  Local tunnel name is R1
    Internet Address 1.1.1.1, port 0
  L2TP class for tunnel is l2tp_default_class
  Counters, taking last clear into account:
    0 packets sent, 0 received
    0 bytes sent, 0 received
    Last clearing of counters never
  Counters, ignoring last clear:
    0 packets sent, 0 received
    0 bytes sent, 0 received
  Control Ns 2, Nr 3
  Local RWS 1024 (default), Remote RWS 512
  Control channel Congestion Control is disabled
  Tunnel PMTU checking disabled
  Retransmission time 1, max 2 seconds
  Unsent queuesize 0, max 0
  Resend queuesize 0, max 1
  Total resends 1, ZLB ACKs sent 1
  Total out-of-order dropped pkts 0
  Total out-of-order reorder pkts 0
  Total peer authentication failures 0
  Current no session pak queue check 0 of 5
  Retransmit time distribution: 0 1 0 0 0 0 0 0 0
  Control message authentication is disabled
```

Static PWs require you to define sessions & cookie parameters under the neighbor where the PW is pointing to. Using static PWs allows the traffic to pass over the PW as soon as possible (avoiding session parameter negotiation).

IOS

```
pseudowire-class L2TP-CLASS
 encapsulation l2tpv3
 protocol none
 ip local interface Loopback0
!
interface POS2/0
 xconnect 2.2.2.2 12 encapsulation l2tpv3 manual pw-class L2TP-CLASS
  l2tp id 1 2
  l2tp cookie local 4 4660
  l2tp cookie remote 8 4660 22136
```

You need to define "protocol none" under the pw-class, otherwise you won't be able to do the static configuration.

IOS-XR

```
l2vpn
 xconnect group A-XCONGRP
```

```
  p2p A-P2P
   neighbor 1.1.1.1 pw-id 100
    l2tp static local cookie size 4 value 0x1234
    l2tp static local session 1
    l2tp static remote cookie size 8 value 0x1234 0x5678
    l2tp static remote session 2
```

Local/remote values must be reversed on the other side.

When you use a static L2TPv3 session, you cannot perform interworking.

Authentication

- L2TPv3 Control Message Hashing (**new method**)
  - hash is computed of entire message
  - all message types are hashed
- L2TP Control Channel Authentication (old method)
  - hash is computed of Challenge AVP message only
  - only SCCRP and SCCCN messages are hashed

IOS

```
l2tp-class OLDAUTH-L2TP-CLASS
 authentication
 hostname USER
 password PASS
!
l2tp-class NEWAUTH-L2TP-CLASS
 digest secret 0 PASS hash SHA1
!
pseudowire-class L2TP-PWCLASS
 encapsulation l2tpv3
 protocol l2tpv3 XXX-L2TP-CLASS
 ip local interface Loopback0
```

IOS-XR

```
l2tp-class OLDAUTH-L2TP-CLASS
 authentication
 hostname USER
 password PASS
!
l2tp-class NEWAUTH-L2TP-CLASS
 digest hash SHA1
 digest secret PASS
!
l2vpn
 pw-class L2TP-PWCLASS
 encapsulation l2tpv3
  protocol l2tpv3 class XXX-L2TP-CLASS
  ipv4 source 2.2.2.2
```

In IOS-XR, the L2TP class used for authentication (and other parameters) is configured outside the l2vpn configuration, in the global configuration.

ATOM

- A working LSP must exist between the PEs.
- If the configuration is correct, then a targeted LDP adjacency will be formed between the PEs and the pseudowire will come up (unless the PEs do not agree on some parameters).
    - The PW-ID must agree on both sides.
    - MTUs must agree on both ACs for the PW to come up (IOS-XR MTU includes the L2 header).
- The tLDP session is used to exchange labels for the pseudowire.
- The PW label is used to differentiate the local ACs, like the VPN label is used to differentiate the local VRFs.

Interfaces at both sides of a local AC must have a common encapsulation (ppp, hdlc, frame-relay) in order for the local AC to come up.

When you change the encapsulation of an interface, any xconnect configuration is removed.

You have to exit from the xconnect or pw-class config mode in order to have it activated.

If you have to change the MTU of the local AC in order to agree with the remote AC, then you must do the same for the interface on the other side of the local AC, to account for any IGPs passing over.

Besides dynamic pseudowires (based on LDP), you can also create static pseudowires by having the labels statically defined on each PE.

Logging

Configure the following if you want to enable logging of various PW events.

IOS

```
xconnect logging pseudowire status
```

IOS-XR

```
l2vpn
 logging
  bridge-domain
  pseudowire
  vfi
```

You can use the "clear xconnect" command if you want to reset a pseudowire.

IOS

```
R1#clear xconnect ?
  all          All xconnect entries
  interface    All xconnects by interface
  peer         All xconnects by Peer IP address
```

PPP Pseudowires

When using xconnect for PPP L2VPN connectivity, then the PPP process of the local AC is disabled.

no xconnect configured

```
Serial2/0 is up, line protocol is up
  Hardware is M4T
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP Open
```

xconnect configured

```
Serial2/0 is up, line protocol is up
  Hardware is M4T
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP Closed, crc 16, loopback not set
```

Frame-Relay

Each PW can be assigned to a single PVC/DLCI.

If you need to transfer all DLCIs of a port, then you have to use HDLC as the interface encapsulation and do the usual xconnect under the interface.

Frame-relay config on the other side of each AC doesn't have to agree, as long as the DLCIs are communicated correctly across the MPLS network.

IOS might require to have frame-relay switching enabled on the PE (IOS-XR and GNS3 don't require it).

A PVC that is transferred across the PW is shown as "switched" under "show frame-relay pvc", while another PVC that terminates locally is shown as "local".

Remember to configure appropriately the IGP network type on the interfaces, if the default isn't the right one.

Wireshark might not decode correctly the FRoMPLS encapsulation.

IOS

```
interface Serial2/0
 encapsulation frame-relay
 frame-relay intf-type dce
!
pseudowire-class FRoMPLS-PWCLASS
 encapsulation mpls
!
connect FRCON Serial2/0 100 l2transport
 xconnect 9.9.9.9 19 pw-class FRoMPLS-PWCLASS
```

IOS-XR

```
interface POS0/0/0/0.100 l2transport
 pvc 100
!
interface POS0/0/0/0
 encapsulation frame-relay
 frame-relay intf-type dce
!
l2vpn
 pw-class FRoMPLS-PWCLASS
  encapsulation mpls
 !
 xconnect group FR-XCONGRP
  p2p R9R1-P2P
   interface POS0/0/0/0.100
   neighbor 1.1.1.1 pw-id 19
    pw-class FRoMPLS-PWCLASS
```

IOS

```
R1#sh connection
ID   Name             Segment 1               Segment                State
==========================================================================
=======
2    FRCON            Se2/0 100               9.9.9.9 19             UP


R1#sh mpls l2transport vc 19
Local intf     Local circuit              Dest address    VC ID
Status
-------------  -------------------------- --------------- ---------- ----
------
Se2/0          FR DLCI 100                9.9.9.9         19         UP
```

Ethernet Pseudowires

- Tagged mode (vlan mode)
  - VC type 5
  - VC type 4 (if vlan tag is to be transfered too)
- Raw mode (port mode)
  - VC type 5

Usually the VC type is auto-negotiated while the PW is being setup.

Configuration

IOS

```
pseudowire-class EoMPLS-PWCLASS
 encapsulation mpls
!
interface X
 xconnect 9.9.9.9 19 pw-class EoMPLS-PWCLASS
```

IOS-XR

```
interface X l2transport
!
l2vpn
 pw-class EoMPLS-PWCLASS
  encapsulation mpls
!
 xconnect group R9R1-XCONGRP
  p2p R9R1-P2P
   interface X
   neighbor 1.1.1.1 pw-id 19
    pw-class EoMPLS-PWCLASS
```

In IOS-XR, you have to define as "`l2transport`" the interface used as AC in the xconnect. If there is existing configuration under the interface, it's better to remove the whole interface, make it L2 and then re-apply the old config (if it's still required.)

Verification

IOS

```
R1#sh mpls l2 vc
Local intf     Local circuit              Dest address    VC ID
Status
-------------  -------------------------- --------------- ---------- ----
------
Fa1/0          Ethernet                   9.9.9.9         19         UP

R1#sh mpls l2 vc detail
Local interface: Fa1/0 up, line protocol up, Ethernet up
```

```
  Destination address: 9.9.9.9, VC ID: 19, VC status: up
    Output interface: Fa0/0.12, imposed label stack {26 16100}
    Preferred path: not configured
    Default path: active
    Next hop: 10.1.2.2
  Create time: 00:20:14, last status change time: 00:20:14
  Signaling protocol: LDP, peer 9.9.9.9:0 up
    Targeted Hello: 1.1.1.1(LDP Id) -> 9.9.9.9, LDP is UP
    Status TLV support (local/remote)    : enabled/supported
      LDP route watch                    : enabled
      Label/status state machine         : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last local SSS circuit status rcvd: No fault
      Last local SSS circuit status sent: No fault
      Last local  LDP TLV    status sent: No fault
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 28, remote 16100
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: Off (configured: autosense)
  VC statistics:
    transit packet totals: receive 169, send 771
    transit byte totals:   receive 21479, send 80640
    transit packet drops:  receive 0, seq error 0, send 0
```

"imposed label stack" shows the complete label stack (Transport label + PW label) used by the local PE to reach the remote PE.

"MPLS VC labels" shows the PW labels exchanged by the PEs.

The status you see next to "Label/status state machine" translates to the following:

- L=Local, R=Remote
- r=ready, n=not ready
- u=up, d=down

In VC statistics, the send direction counts the traffic from the AC to the PW, while the receive direction counts the traffic from the PW to the AC.

IOS-XR

```
R9#sh l2vpn xconnect detail
Group R9R1-XCONGRP, XC R9R1-P2P, state is up; Interworking none
  AC: TenGigE0/0/0/9, state is up
    Type Ethernet
    MTU 1500; XC ID 0x40002; interworking none
    Statistics:
      packets: received 2324, sent 4514
      bytes: received 1745, sent 4668
  PW: neighbor 1.1.1.1, PW ID 19, state is up ( established )
    PW class EoMPLS-PWCLASS, XC ID 0xc0000001
    Encapsulation MPLS, protocol LDP
    Source address 9.9.9.9
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    Load Balance Hashing: src-dst-ip
    Flow Label flags configured (Tx=1,Rx=1), negotiated (Tx=0,Rx=0)

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------
------
      Label        16100                          28
      Group ID     0x1c0                          0x0
      Interface    TenGigE0/0/0/9                 ** AC **
      MTU          1500                           1500
      Control word disabled                       disabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x6                            0x2
                   (router alert label)           (router alert label)
                   (TTL expiry)
      ------------ ------------------------------ -----------------------
------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 3221225473
    Create time: 27/12/2013 03:23:01
    Last time status changed: 27/12/2013 03:26:49
    Statistics:
      packets: received 4514, sent 2324
      bytes: received 4668, sent 1745
```

IOS

```
R1#sh xconnect all
Legend:    XC ST=Xconnect State  S1=Segment1 State  S2=Segment2 State
  UP=Up       DN=Down            AD=Admin Down      IA=Inactive
  SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware

XC ST  Segment 1                         S1 Segment
2                         S2
------+---------------------------------+--+-----------------------------
----+--
```

```
UP     ac    Fa1/0(Ethernet)                UP mpls 9.9.9.9:19
    UP
```

IOS

```
R1#sh xconnect all detail
Legend:    XC ST=Xconnect State  S1=Segment1 State  S2=Segment2 State
  UP=Up       DN=Down            AD=Admin Down      IA=Inactive
  SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware

XC ST  Segment 1                         S1 Segment
2                         S2
------+---------------------------------+--+--------------------------
----+--
UP     ac    Fa1/0(Ethernet)                UP mpls 9.9.9.9:19
    UP
             Interworking: none                     Local  VC label 28
                                                    Remote VC label 16100
                                                    pw-class: EoMPLS-PWCLASS
```

IOS-XR

```
R9#sh l2vpn xconnect
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                 Segment 2
Group         Name     ST   Description           ST    Description
ST
------------------------   -----------------------   --------------------
--
R9R1-XCONGRP R9R1-P2P UP   Te0/0/0/9             UP    1.1.1.1    19
UP
--------------------------------------------------------------------------
--
```

When viewing the LFIB, the direction is the opposite from L3VPNs, pointing towards the AC this time.

IOS

```
R1#sh mpls forwarding-table
Local      Outgoing    Prefix           Bytes Label    Outgoing    Next Hop
Label      Label       or Tunnel Id     Switched       interface
28         No Label    l2ckt(19)        19658          Fa1/0
point2point
```

IOS-XR

```
R9#sh mpls forwarding
Local  Outgoing     Prefix              Outgoing      Next Hop          Bytes
Label  Label        or ID               Interface
Switched
```

```
------ ----------- ------------------ ------------ --------------- ------
------
16100  Pop         PW(1.1.1.1:19)     Te0/0/0/9    point2point     464883
```

You can also use **ping and traceroute in order to verify the PW** (although it won't help you much in case of a core problem). In IOS-XR, **mpls oam** must first be activated.

IOS

```
R1#ping mpls pseudowire 9.9.9.9 19 segment 1 verbose repeat 1
Sending 1, 100-byte MPLS Echos to 9.9.9.9,
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
!    size 100, reply addr 10.8.9.9, return code 3[Labels: 16100 Exp: 0]
     Rx Interface: 10.8.9.9
    local 1.1.1.1 remote 9.9.9.9 vc id 19

Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
```

IOS

```
R1#traceroute mpls pseudowire 9.9.9.9 19 segment
Tracing MS-PW segments within range [1-1] peer address 9.9.9.9 and
timeout 2 seconds

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
! 1 10.8.9.9 1 ms [Labels: 16100 Exp: 0]
    local 1.1.1.1 remote 9.9.9.9 vc id 19
```

Static PWs

You can also use **static pseudowires (without tLDP between PEs)** if you manually define the local/remote labels to be used.

IOS

```
mpls label range 100 200 static 201 300
!
interface X
 xconnect 1.1.1.1 22 encapsulation mpls manual pw-class STATICPW-PWCLASS
  mpls label 201 202
```

IOS-XR

```
l2vpn
 xconnect group XCONGRP
  p2p LS-P2P
   neighbor 1.1.1.1 pw-id 22
    mpls static label local 201 remote 202
```

In IOS-XR, labels 16 to 15999 are reserved (by default) for static L2VPN pseudowires.

Interworking

Interworking is actually a transforming function that is required to interconnect two heterogeneous ACs, by providing the translation between the different L2 encapsulations.

There are 3 main modes:

- Ethernet (bridged)
  - ethernet frames are extracted from the AC and sent over the PW
  - non-ethernet frames are dropped
  - ethernet frames with vlan tags, have their tags removed and then send over the PW
- Vlan
  - ethernet frames are extracted from the AC and sent over the PW
  - non-ethernet frames are dropped
  - ethernet frames with vlan tags, are send with their tags over the PW
- IP (routed)
  - ipv4 packets are extracted from the AC and sent over the PW
  - non-ipv4 packets are dropped

Supported modes/technologies:

- Ethernet/Vlan to Frame-Relay
  - MPLS/L2TPv3
  - Ethernet/IP
- Ethernet/Vlan to PPP

    - MPLS
    - IP
- Ethernet to Vlan
    - MPLS/L2TPv3
    - Ethernet/IP
- Frame-Relay to PPP
    - MPLS/L2TPv3
    - IP

When doing interworking with frame-relay, **you can't use inverse-arp** (because there is no ip address on the other side of the local AC, neither at the other side of the remote AC), so you'll have to either **use static mapping** or **change the frame-relay configuration to point-to-point**.

In frame-relay interworking the PE router reports the PVC status to the CE router, based on the status of the pseudowire with the remote PE. Also the status of the local PVC detected by the PE router will be reflected into the pseudowire status towards the remote PE.

For ethernet, the PE acts like a proxy-ARP server to all ARP requests from the CE router. Sometimes you might need to clear the arp table on the CE after making changes on the PE.

For PPP, proxy IPCP is automatically enabled on the PE router when IP interworking is configured on the pseudowire.

Most of the times, you must **configure all IGPs for point-to-point operation** between the CE routers when configuring an Ethernet to non-Ethernet interworking setup.

Although it might not be applicable in the lab, generally you should take care of the AC MTU so that it doesn't exceed the core MTU.

IOS

```
pseudowire-class EoMPLS-IW-PWCLASS
 encapsulation mpls
 interworking ip
```

IOS-XR

```
l2vpn
 xconnect group A-XCONGRP
  p2p A-P2P
   interworking ipv4
```

The rest of the configuration is the same as in normal L2VPN setup.

Interworking doesn't apply in VPLS.

PW Redundancy

IOS

Ethernet

```
interface FastEthernet1/0
 xconnect 9.9.9.9 19 pw-class EoMPLS-IW-PWCLASS
  backup peer 8.8.8.8 18 pw-class EoMPLS-IW-PWCLASS
```

Frame-Relay

```
connect FRCON Serial2/0 100 l2transport
 xconnect 9.9.9.9 19 pw-class FRoMPLS-PWCLASS
  backup peer 8.8.8.8 18 pw-class FRoMPLS-PWCLASS
```

The "backup peer" command is configured on the PE which will handle the redundancy.

Configure the following if you want to enable logging of the PW redundancy events.

IOS

```
xconnect logging redundancy
```

The PW status is signaled between the PEs, if it's supported (it's enabled by default).

IOS

```
R1#sh xconnect all
Legend:    XC ST=Xconnect State  S1=Segment1 State  S2=Segment2 State
  UP=Up       DN=Down            AD=Admin Down      IA=Inactive
  SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware

XC ST  Segment 1                         S1 Segment
2                        S2
------+---------------------------------+--+-------------------------------
----+--
UP pri ac    Fa1/0(Ethernet)             UP mpls 9.9.9.9:19
      UP
IA sec ac    Fa1/0(Ethernet)             UP mpls
5.5.5.5:15                  SB

R1#sh mpls l2transport vc

Local intf     Local circuit              Dest address    VC ID
Status
-------------  -------------------------- --------------- ---------- ----
------
Fa1/0          Ethernet                   5.5.5.5         15
STANDBY
Fa1/0          Ethernet                   9.9.9.9         19         UP
```

```
R1#sh mpls l2transport vc 15 detail
Local interface: Fa1/0 up, line protocol up, Ethernet up
  Destination address: 5.5.5.5, VC ID: 15, VC status: standby
    Output interface: Fa0/0.12, imposed label stack {17 27}
    Preferred path: not configured
    Default path: active
    Next hop: 10.1.2.2
  Create time: 00:04:30, last status change time: 00:03:36
  Signaling protocol: LDP, peer 5.5.5.5:0 up
    Targeted Hello: 1.1.1.1(LDP Id) -> 5.5.5.5, LDP is UP
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LrdRru
      Last local dataplane   status rcvd: No fault
      Last local SSS circuit status rcvd: DOWN(UP, standby)
      Last local SSS circuit status sent: No fault
      Last local  LDP TLV    status sent: DOWN(standby)
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 32, remote 27
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: On (configured: autosense)
  VC statistics:
    transit packet totals: receive 116, send 0
    transit byte totals:   receive 8221, send 0
    transit packet drops:  receive 0, seq error 0, send 0

R1#ping mpls pseudowire 5.5.5.5 15
%Total number of MS-PW segments is less than segment number; Adjusting
the segment number to 1
Sending 5, 100-byte MPLS Echos to 5.5.5.5,
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 72/78/88 ms
```

You can configure the topology as "dual-homed" under the pw-class, if you want to have both primary and backup pseudowires appear as active simultaneously. In any case, traffic will continue to pass through the primary PW only.

IOS

```
pseudowire-class EOMPLS-PWCLASS
 encapsulation mpls
 status peer topology dual-homed
```

Keep in mind that the xconnect output is based on the configuration, while the mpls l2vc output is based on the result.

IOS

```
R1#sh xconnect all
Legend:    XC ST=Xconnect State  S1=Segment1 State  S2=Segment2 State
  UP=Up       DN=Down            AD=Admin Down      IA=Inactive
  SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware

XC ST  Segment 1                         S1 Segment
2                         S2
------+---------------------------------+--+-----------------------------
----+--
UP pri ac   Fa1/0(Ethernet)              UP mpls 9.9.9.9:19
            UP
IA sec ac   Fa1/0(Ethernet)              UP mpls
5.5.5.5:15                   SB

R1#sh mpls l2 vc

Local intf     Local circuit              Dest address    VC ID
Status
-------------  -------------------------- --------------- ---------- ----
------
Fa1/0          Ethernet                   5.5.5.5         15         UP
Fa1/0          Ethernet                   9.9.9.9         19         UP
```

Links

- IETF - RFC 4447

IOS-XR l2vpn configuration

Everything must be configured under the "l2vpn" hierarchy, including the pw-classes.

VPWS

IOS-XR

```
l2vpn
 xconnect group A-XCONGRP
  p2p A-P2P
   interface X
   !
   neighbor Y pw-id k
    pw-class Z
```

VPLS

IOS-XR

```
l2vpn
 bridge group A-BRGRP
  bridge-domain A-BRDOM
   interface X1
   interface X2
   !
   vfi A-VFI
    neighbor Y1 pw-id k1
     pw-class Z
    neighbor Y2 pw-id k2
     pw-class Z
```

xconnect/bridge groups are used solely for configuration grouping; you can put all bridge-domains under the same bridge group and all the p2ps under the same xconnect group.

You can have a neighbor directly under the bridge-domain, if you want to make this an access pseudowire. For core pseudowires (which will be 99% the case) you need to define the neighbor under the vfi.

Below you can find some useful commands for verification and troubleshooting.

Check AToM (AC/PW details, status, MTU, traffic, etc.)

IOS-XR

```
ASR9k#sh l2vpn xconnect ?
  brief          Encapsulation brief
  detail         Detailed information
  encapsulation  Filter on encapsulation
  group          Filter on group
  groups         All groups information
  interface       Filter on interface
  mp2mp           MP2MP Information
  mspw            ms_pw Information
  neighbor        Filter on neighbor
  private         Private information
  pw-class        Filter on pseudowire class
  state           Filter on xconnect state
  summary         Summary information
  type            Filter on xconnect type
  |               Output Modifiers
```

Check VPLS (AC/PW details, status, MTU, traffic, split-horizon, L2 features like storm-control, unicast/multicast/broadcast flooding, etc.)

IOS-XR

```
ASR9k#sh l2vpn bridge-domain ?
  autodiscovery   BGP/Radius autodiscovery information
  bd-name         Filter on bridge domain name
  brief           Brief information
  detail          Detailed information
  group           Filter on bridge domain group name
  hardware        Hardware information
  interface       Filter on interface
  neighbor        Filter on neighbor
  objects         Display bridge-domain objects information
  pbb             Provider backbone bridge information
  private         Private information
  summary         Bridge domain summary information
  |               Output Modifiers
```

Check VPLS (mac-addresses and various L2 features like DHCP snooping, ARP inspection, etc.)

IOS-XR

```
ASR9k#sh l2vpn forwarding bridge-domain ?
  WORD          Specify Bridge group name:Bridge domain name
  debug         Include debug information
  detail        Detailed information
  hardware      Read from hardware
  location      Specify a location
  mac-address   MAC address information
  mroute        multicast route information
  pbb           Provider backbone bridge information
  private       Include private information
```

L2transport ethertype

When configuring a l2transport interface in IOS-XR you have the following options regarding its tagging/ethertype:

- dot1q single vlan
  - ethertype 0x8100 (default)
- dot1q double vlan (QinQ)
  - ethertype 0x8100 (default)
  - ethertype 0x9100
  - ethertype 0x9200
- dot1ad single vlan
  - ethertype 0x88a8 (default)
- dot1ad double vlan
  - ethertype 0x88a8 (default)

The above ethertype refers always to the outer vlan.

dot1q refers to 802.1q, while dot1ad refers to 802.1ad (which is the standard).

H-VPLS

Instead of having a full-mesh of core pseudowires from end to end, you partition the network into smaller access domains (containing U-PEs and CEs) and a core domain (containing N-PEs and Ps).

N=Network
U= User

- core domain
  - core PWs between all N-PEs
- access domains
  - access PWs between each U-PE and a core N-PE

You can also have simple L2 ethernet circuits instead of access PWs.

Configuration-wise the idea is the same as above:

- CE
  - no change
- U-PE
  - one (or two) p2p PW from the U-PE to one (or two) N-PE
- N-PE
  - full-mesh of p2p PWs between all N-PEs
- P
  - just MPLS switching

Example N-PE Config

IOS-XR

```
l2vpn
 bridge group A-BRGRP
  bridge-domain A-BRDOM
   neighbor U-PE1 pw-id k1
    pw-class Z
   !
   vfi A-VFI
    neighbor N-PE2 pw-id k1
     pw-class Z
    neighbor N-PE3 pw-id k2
     pw-class Z
```

The above N-PE is responsible for switching the traffic from the access PW (coming from U-PE1) to the core PWs (going to N-PE2 and N-PE3) and vice versa, based on the usual mac learning.

Remember that access PWs go outside the VFI and core PWs go inside the VFI, while everything goes under the bridge-domain.

Local Switching

Vlan to Vlan

IOS

```
interface FastEthernet0/0.1
 encapsulation dot1q 1
!
interface FastEthernet0/0.2
 encapsulation dot1q 2
!
connect VLAN2VLAN FastEthernet0/0.1 FastEthernet0/0.2
```

IOS-XR

```
interface TenGigE0/0/0/0.1 l2transport
 dot1q vlan 100
!
interface TenGigE0/0/0/0.2 l2transport
 dot1q vlan 200
!
l2vpn
 xconnect group XCONGRP
  p2p LS-P2P
   interface TenGigE0/0/0/0.1
   interface TenGigE0/0/0/0.2
```

Frame-Relay to Frame-Relay

IOS

```
frame-relay switching
!
interface Serial2/0
 encapsulation frame-relay
 frame-relay interface-dlci 100 switched
 frame-relay intf-type dce
!
interface Serial2/1
 encapsulation frame-relay
 frame-relay interface-dlci 200 switched
 frame-relay intf-type dce
!
connect FR2FR serial2/0 100 serial2/1 200
```

L2TPv3 Local Switching

IOS

```
pseudowire-class L2TP-1-CLASS
 encapsulation l2tpv3
 ip local interface Loopback0
!
pseudowire-class L2TP-11-CLASS
 encapsulation l2tpv3
 ip local interface Loopback1
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface Loopback1
 ip address 11.11.11.11 255.255.255.255
!
interface FastEthernet0/0
 xconnect 11.11.11.11 111 pw-class L2TP-1-CLASS
!
interface FastEthernet0/1
 xconnect 1.1.1.1 111 pw-class L2TP-11-CLASS
```

2 loopbacks are required for L2TPv3 local switching to work.

IOS

```
R1#sh xconnect all
Legend:    XC ST=Xconnect State  S1=Segment1 State  S2=Segment2 State
  UP=Up       DN=Down            AD=Admin Down      IA=Inactive
  SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware

XC ST  Segment 1                         S1 Segment
2                         S2
------+---------------------------------+--+-----------------------------
----+--
UP     ac   Fa0/1(Ethernet)              UP l2tp
1.1.1.1:111                  UP
UP     ac   Fa0/0(Ethernet)              UP l2tp
11.11.11.11:111              UP

R1#sh l2tun tunnel sum

L2TP Tunnel Information Total tunnels 2 sessions 2

LocTunID   RemTunID   Remote Name   State  Remote Address  Sessn L2TP
Class/
                                                           Count VPDN

Group
1280397094 3308679294 R1            est    11.11.11.11     1
l2tp_default_cl
3308679294 1280397094 R1            est    1.1.1.1         1
l2tp_default_cl
```

L2VPN/AToM over GRE

Configuration Steps

- create a GRE tunnel with destination the remote L2VPN PE
- enable MPLS on the GRE tunnel
- route the PW through the GRE tunnel (using IGP, static route, preferred-path)

R2 (IOS)

```
interface Tunnel0
 ip unnumbered Loopback0
 mpls ip
 tunnel source 2.2.29.2
 tunnel destination 2.2.79.7
!
ip route 2.2.0.7 255.255.255.255 Tunnel0
!
```

```
interface Ethernet1/0
 xconnect 2.2.0.7 27 pw-class PWCLASS
```

R7 (IOS)

```
interface Tunnel0
 ip unnumbered Loopback0
 mpls ip
 tunnel source 2.2.79.7
 tunnel destination 2.2.29.2
!
ip route 2.2.0.2 255.255.255.255 Tunnel0
!
interface Ethernet1/0
 xconnect 2.2.0.2 27 pw-class PWCLASS
```

IOS-XR

```
interface tunnel-ip0
 ipv4 address 10.10.10.1/24
 tunnel mode gre ipv4
 tunnel source 11.11.11.1
 tunnel destination 12.12.12.2
!
mpls ldp
 interface tunnel-ip0
!
l2vpn
 xconnect group XCONGRP
  p2p P2P
   neighbor 2.2.2.2 pw-id 12
```

The other end of the tunnel is also the remote PE where the PW terminates.

You must also enable an IGP on the tunnel (or create a static route for a different hop), in order to drive the PW destination across it. Be careful of recursive routing (tunnel destination must not go through the tunnel itself).

In latest IOS-XR, you can also use the **preferred tunnel-path** feature in order to map the PW to a specific GRE tunnel, instead of pointing the PW to the remote PE router..

IOS-XR

```
l2vpn
 pw-class GRE-PWCLASS
  encapsulation mpls
  preferred-path interface tunnel-ip0
```

Only PE-to-PE tunnels are supported when using preferred-path.

IOS supports something similar, but in very specific hardware and software.

PW Switching or Multisegment PW

An L2VPN multisegment PW (MS-PW, also known as switched PW) is a set of two or more PW segments that function as a single PW. They can be inside a single AS or they can span multiple AS.

The end routers are called terminating PE routers (T-PEs) and the switching routers are called S-PE routers.

S-PE's role is to:

- terminate the tunnels of the preceding and succeeding PW segments
- switch the control and data planes of the preceding and succeeding PW segments

An MS-PW is declared to be up when all the single-segment PWs are up.

Interworking must be configured only at the edge PEs, not on the S-PEs.

In an MS-PW spanning PE-1 <=> S-PE-2 <=> S-PE-3 <=> PE-4, the configuration would be the following:

IOS

PE-1
```
interface X
 xconnect S-PE-2 12 encapsulation mpls
```

S-PE-2
```
l2 vfi MS123-VFI point-to-point
 neighbor PE-1 12 encapsulation mpls
 neighbor S-PE-3 23 encapsulation mpls
```

S-PE-3
```
l2 vfi MS234-VFI point-to-point
 neighbor S-PE-2 23 encapsulation mpls
 neighbor PE-4 34 encapsulation mpls
```

PE-4
```
interface X
 xconnect S-PE-3 34 encapsulation mpls
```

BGP-Based VPLS Autodiscovery

It enables each VPLS PE router to discover "automatically" the other PE routers that are part of the same VPLS domain using BGP.

BGP uses the L2VPN RIB to store endpoint provisioning information, which is updated each time a L2 VFI is configured. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, the endpoint information is used to configure a PW mesh between these neighbors.

In older releases, VPLS Autodiscovery is not supported with L2TPv3 or Inter-AS topologies.

You can configure both autodiscovered and manually configured PWs in the same VFI, but not for the same peer PE. If you need to do that, then configure appropriately the RT values to prevent each peer from receiving discovery data for that VPLS.

You will want to configure extra static VPLS neighbors for:

- PEs that do not participate in the autodiscovery process (in order to join the VPLS)
- PEs that have been configured with the Tunnel Selection feature
- U-PEs in H-VPLS configurations that do not participate in the autodiscovery process and have split-horizon forwarding disabled

The same RT is not allowed in multiple VFIs in the same PE router.

In older releases, the BGP peer address must be a /32 address bound to the peer's LDP router-id.

Split-horizon should not be disabled on autodiscovered neighbors.

Route reflectors can be used to reflect the BGP VPLS prefixes without having VPLS explicitly configured on them.

IOS

```
l2 vfi VPLSAUTO-VFI autodiscovery
 vpn id 100
!
router bgp 1
 no bgp default ipv4-unicast
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
!
 address-family l2vpn vpls
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
 exit-address-family
```

The same vpn id must be configured on all PEs that are part of the same VPN.

It's good practice to also use "`bgp update-delay 1`" in order to minimize the initial update time.

There are various options you can configure under the l2 vfi, with the RT being the most important.

IOS

```
l2 vfi VPLSAUTO-VFI autodiscovery
 vpn id X
 vpls-id X:Y
 rd X:Y
 route-target import/export X:Y
 l2 router-id x.x.x.x
```

Default values:

- **vpls-id**: ASN:vpn-id
- **rd**: ASN:vpn-id
- **route-target**: lower 6 bytes of rd and vpls-id
- **l2 router-id**: LDP router-id

VPLS ID is a BGP extended community value that identifies the VPLS domain. It must be the same between PEs of the same VPN and unique between different VFIs of the same PE. In case of Inter-AS VPLS, it must be **defined manually** and be the same on all PEs of the same VPLS domain, since ASNs are different.

RSVP/MPLS-TE

MPLS-TE (MPLS Traffic Engineering) Requirements are described in RFC 2702.
RSVP (Resource Reservation Protocol) is defined in RFC 2205.
RSVP-TE (RSVP Traffic Engineering extension) is defined in RFC 3209.
OSPFv2 extensions for TE are defined in RFC 3630.
OSPFv3 extensions for TE are defined in RFC 5329.
ISIS extensions for TE are defined in RFC 5305.

RSVP

RSVP messages are sent hop-by-hop between RSVP-enabled routers as "raw" IP packets with protocol number 46.

Two main messages:

- **PATH**
  - sent with the same source and destination addresses as the actual data (i.e. tunnel src/dst), in order to be routed correctly even through non-RSVP routers
  - objects
    - Label Request Object
    - Explicit Route Object (ERO)
    - Record Route Object (RRO)
    - Session Attribute Object (SAO)
    - Sender TSpec
- **RESV**
  - sent hop-by-hop by having each RSVP router forward a RESV message to the previous RSVP hop (learned by the PATH message)
  - objects
    - Label Object
    - Record Route Object (RRO)

Each RESV message follows the exact reverse path of the initial PATH message

As a general case (but not applicable to MPLS-TE), if the PATH message arrives at a router that does not understand RSVP, that router should forward the message as a normal packet without interpreting the contents of the message and will not reserve resources for the flow.

Assume the following network:

R1-R2-R3-R4

where an MPLS-TE tunnel is going to be built from R1 to R4.

Then the following actions will happen when an RSVP reservation is asked:

1. R1
    1. checks reservation attributes
    2. sends PATH message with src/dst R1/R4 to next-hop (R2)
2. R2
    1. receives PATH message with src/dst R1/R4 from previous-hop (R1)
    2. checks reservation attributes
    3. sends PATH message with src/dst R1/R4 to next-hop (R3)
3. R3
    1. receives PATH message with src/dst R1/R4 from previous-hop (R2)
    2. checks reservation attributes
    3. sends PATH message with src/dst R1/R4 to next-hop (R4)
4. R4
    1. receives PATH message with src/dst R1/R4 from previous-hop (R3)
    2. checks reservation attributes
    3. makes the reservation
    4. sends RESV message with src/dst R4/R3 to previous-hop (R3)
5. R3
    1. receives RESV message with src/dst R4/R3 from R4
    2. sends RESV message with src/dst R3/R2 to previous-hop (R2)
6. R2
    1. receives RESV message with src/dst R3/R2 from R3
    2. sends RESV message with src/dst R2/R1 to previous-hop (R1)
7. R1
    1. receives RESV message with src/dst R2/R1 from R2
    2. creates the tunnel (up/up)

In case of MPLS-TE, one reservation attribute could be the transport label required in order to build the LSP tunnel.

Labels are allocated downstream (R1=>R4) using PATH messages and propagated upstream (R4=>R1) using RESV messages.

tunnel head-end = RSVP sender
tunnel tail-end = RSVP receiver

The request to bind labels to a specific LSP tunnel is initiated by the tunnel ingress node (known as **RSVP sender**) using a PATH message.

The reply to this label allocation PATH message is sent from the egress node (known as **RSVP receiver**) back upstream towards the sender using a RESV message, following the path state across all nodes created by the initial PATH message in reverse order.

Each intermediate node that receives a RESV message containing a label, uses that label for outgoing traffic associated with this specific LSP tunnel. If the node is not the sender, it allocates a new label and places that new label into a new RESV message which it sends upstream to the previous-hop, until it reaches the sender and a complete LSP is built.

When using "`sh ip rsvp int`" to verify the rsvp bandwidth on an interface, all rsvp enabled interfaces are shown, even the ones that are shut down.

RSVP hellos are not enabled by default, since they are not needed for normal MPLS TE operation.

RSVP & Fast Failure Detection

When RSVP signals a TE LSP and there is a failure somewhere along the path, the failure can remain undetected for as long as two minutes (timeout of RSVP session).

Two solutions can be used for triggering fast detection in case of a failure:

- RSVP hellos
- BFD

RSVP hellos

IOS

```
ip rsvp signalling hello
!
interface X
 ip rsvp signalling hello
```

IOS-XR

not supported

IOS-XR supports only BFD for fast detection (< 1sec). RSVP hellos can be used for simple reroute (detection > 10sec) or GR.

The same type of "`rsvp signalling hello`" must be configured on the other side too for fast detection to take place.

IOS

```
R4#sh ip rsvp hello
Hello:
  RSVP Hello for Fast-Reroute/Reroute: Enabled
    Statistics: Disabled
  BFD for Fast-Reroute/Reroute: Disabled
  RSVP Hello for Graceful Restart: Disabled
```

```
R4#sh ip rsvp hello client lsp sum
Local           Remote          tun_id lsp_id subgrp_orig     subgrp_id
FLAGS
2.2.2.2         19.19.19.19     0      46     0.0.0.0         0
0x24

R4#sh ip rsvp hello client lsp det
  Hello Client LSPs (all lsp tree)

  Tun Dest:   19.19.19.19  Tun ID: 0  Ext Tun ID: 2.2.2.2
  Tun Sender: 2.2.2.2  LSP ID: 46
    Lsp flags: 0x24
    Lsp RR DN nbr: 20.4.5.5 FRR


R4#sh ip rsvp hello client nbr
Local           Remote          Type    NBR_STATE    HI_STATE    LSPs
20.4.5.4        20.4.5.5        RR      Normal       Up             1

R4#sh ip rsvp hello client nbr det
  Hello Client Neighbors

        Remote addr 20.4.5.5,    Local addr  20.4.5.4
    Nbr State: Normal    Type: Reroute
    Nbr Hello State: Up
    LSPs protecting: 1
    I/F: Fa0/0.45
```

In IOS, there are 3 different commands to set the **hello refresh interval**:

- `ip rsvp signalling hello refresh interval xxx`
    - refresh interval: 10-30000 ms
- `ip rsvp signalling hello fast-reroute refresh interval xxx`
    - refresh interval: 10-30000 ms
- `ip rsvp signalling hello reroute refresh interval xxx`
    - refresh interval: 1000-30000 ms

The first two ones are the same and are the ones that actually enable/configure FRR. The number of refresh misses is configured likewise.

BFD

Check "BFD" for more information.

In both cases in IOS, RSVP hellos must be configured both globally on the router and on the specific interface in order to be operational.

MPLS-TE Configuration

- enable MPLS-TE globally
- enable MPLS-TE & RSVP under each interested interface
- enable MPLS-TE extensions under the IGP
- create the MPLS-TE tunnel

Minimum configuration includes the following:

IOS

```
! global
mpls traffic-eng tunnels

! per interface
interface X
 mpls traffic-eng tunnels
 ip rsvp bandwidth


! per IGP
router isis X
 metric-style wide
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng level x
!
router ospf X
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area x


! per tunnel
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination x.x.x.x
 tunnel mpls traffic-eng path-option 1 dynamic
```

IOS-XR

```
! global
mpls ldp

! per interface
rsvp
 interface X
!
```

```
mpls traffic-eng
 interface X


! per IGP
router ospf X
 area 0
  mpls traffic-eng
 mpls traffic-eng router-id Loopback0
!
router isis X
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level x
  mpls traffic-eng router-id Loopback0


! per tunnel
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination x.x.x.x
 path-option 1 dynamic
```

Is IOS-XR you must also enable MPLS LDP (even when LDP is not going to be used), otherwise data-plane won't work and you'll see unresolved CEF entries.

If you don't set the router-id under the IGP, you won't be able to create any TE paths for the tunnels.

You have to configure "`ip rsvp bandwidth`" in order to set the reserved bandwidth per interface (tunnels can work without setting it, as long as their required bandwidth is 0).

Default bandwidth is 75% in IOS and 0 in IOS-XR (IOS-XR doesn't support the setting of bandwidth as a percentage).

You won't see the RSVP bandwidth of an interface as allocated, unless the tunnel that passes through it is fully operational.

RSVP bandwidth is reserved only on the egress interface of a router (based on the direction of the tunnel).

Don't forget to enable wide metrics if using IS-IS (IOS gives you a warning).

In IOS-XR every tunnel type has a specific interface name. For MPLS-TE tunnels, the name "**tunne-te**" is used.

Each TE tunnel is unidirectional, so if you want to make it behave like bidirectional, then you need to configure a reverse tunnel from the tail-end to the head-end too. Only "`ip unnumbered`" should be used in TE tunnels due to their unidirectional nature.

Most tunnel changes require to shut/no shut the tunnel interface in order to have immediate effect.

Setting the (signaled) bandwidth on the tunnel is not necessary for normal operation, but it might help to verify the TE path when checking the RSVP interface bandwidth on the path.

MPLS TE in Cisco software supports only a single IGP process/instance.

For extra logging information, you can configure the following:

IOS

```
R1(config)#mpls traffic-eng logging lsp ?
  path-errors          Log LSP Path Error traps
  preemption           Log LSP Preemption traps
  reservation-errors   Log LSP Reservation Error traps
  setups               Log LSP Establishment Traps
  teardowns            Log LSP Teardown Traps

R1(config)#mpls traffic-eng logging tunnel ?
  lsp-selection  Log Tunnel LSP Selection traps
  path           Log Tunnel Path-related traps
```

IOS-XR

```
C12k(config-if)#logging events ?
  link-status  Enable interface and line-protocol state change alarms
  lsp-status   Enable interface LSP state change alarms

C12k(config-if)#logging events lsp-status ?
  insufficient-bandwidth  Enable Syslog for setup/reopt failure due to
bandwidth
  reoptimize              Enable interface LSP REOPT change alarms
  reroute                 Enable interface LSP REROUTE change alarms
  state                   Enable interface LSP UP/DOWN change alarms
  switchover              Enable interface LSP SWITCHOVER change alarms
```

MPLS-TE Verification

When using MPLS-TE in place of LDP for a L3VPN connection, then the transport label is provided by RSVP.

IOS

```
R2#sh mpls traffic-eng tunnels tunnel 0 | i Label
  InLabel  :  -
  OutLabel : FastEthernet0/0.24, 16
```

```
R2#sh mpls forwarding-table 19.19.19.19 detail
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
None       16         19.19.19.19/32   0             Tu0
point2point
        MAC/Encaps=18/22, MRU=1500, Label Stack{16}, via Fa0/0.24
        CA0611400000CA0509F00000810000188847 00010000
        No output feature configured
```

Don't forget to use the complete prefix when checking the LFIB in the PE, otherwise "`sh mpls forwarding-table`" won't show you anything about it.

The VPN label is still provided by BGP VPNv4.

IOS

```
R2#sh bgp vpnv4 unicast all 20.20.20.20/32 | i label
      mpls labels in/out nolabel/17
```

You can also check the labels assigned by RSVP in the intermediate routers and follow the path like in LDP labels.

IOS

```
R4#sh mpls traffic-eng tunnels | i Label
  InLabel  : FastEthernet0/0.24, 16
  OutLabel : FastEthernet0/0.45, 26

R5#sh mpls traffic-eng tunnels | i Label
  InLabel  : FastEthernet0/0.45, 26
  OutLabel : FastEthernet0/0.519, implicit-null
```

Keep in mind that "`sh mpls forwarding-table`" will show you the tunnel source prefix (which originated the RSVP messages) when referring to TE tunnels.

IOS

```
R4#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         26         2.2.2.2 0 [1]    7658          Fa0/0.45   20.4.5.5

R5#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
26         Pop Label  2.2.2.2 0 [1]    7540          Fa0/0.519
20.5.19.19
```

The number "`0 [1]`" after the prefix are the "`Tun_Id`" and "`Tun_instance`" found in "`sh mpls traffic-eng tunnels`" output.

```
R5#sh mpls traffic-eng tunnels | i Tun
LSP Tunnel R2_t0 is signalled, connection is up
       Src 2.2.2.2, Dst 19.19.19.19, Tun_Id 0, Tun_Instance 1
```

You can use "clear ip rsvp reservation *" if you want to clear the RSVP label bindings (i.e. after you change the mpls label range).

You can use the following command in order to find the optimal path for a tunnel, based on current resources:

IOS

```
R2#sh mpls traffic-eng tunnels suboptimal constraints current | b Short
  Shortest Availability-Constrained Path Info:
    Path Weight: 30 (TE)
    Explicit Route: 20.2.4.2 20.2.4.4 20.4.5.4 20.4.5.5
                    20.5.19.5 20.5.19.19 19.19.19.19
```

Other options include:

```
R2#sh mpls traffic-eng tunnels suboptimal constraints ?
  current  path lookup constrained by available resources
  max      path lookup constrained by network's maximum potential
resources
  none     path lookup without any constraints
```

The same info is available if you use the "sh mpls traffic-eng tunnels tunnel X" command, explicitly specifying the tunnel.

An easy way to find the address (and label when using FRR) of each hop, is to enable record-route.

IOS

```
interface Tunnel0
 tunnel mpls traffic-eng record-route
```

IOS-XR

```
interface tunnel-te0
 record-route
```

IOS

```
R2#sh mpls traffic-eng tunnels | b Resv
    RSVP Resv Info:
      Record   Route:  20.3.4.3 20.4.5.4 20.5.19.5 20.5.19.19
```

Besides various show commands, you can use "debug mpls traffic-eng tunnels" in order to check for any error messages while setting up the tunnel path. Other possible debugs are the following:

```
R2#debug mpls traffic-eng tunnels ?
  auto-bw            MPLS Traffic Engineering tunnel auto-bw
  errors             MPLS Traffic Engineering tunnel errors
  events             MPLS Traffic Engineering tunnel system events
  fast-reroute       MPLS Traffic Engineering tunnel fast reroute
  labels             MPLS Traffic Engineering tunnel labels
  path-protection    MPLS Traffic Engineering tunnel path protection
  reoptimize         MPLS Traffic Engineering tunnel reoptimization
  signalling         MPLS Traffic Engineering tunnel signalling
  state              MPLS Traffic Engineering tunnel state machine
  timers             MPLS Traffic Engineering tunnel timers
```

Some of these debugs are replicated in the RSVP debug commands.

If you enable "`debug mpls traffic-eng tunnels signalling detail`" and there is an error in RSVP communication somewhere in the path, you'll get the following message, which will guide you to the hop with the problem:

```
TE-SIG-HE: Tunnel0 [71]->1.1.1.1: received Path Error from 20.2.4.2:
Routing Problem: No route available toward destination
```

Traffic steering & LSP attributes

You can use the following methods in order to steer the tunnel traffic through a specific TE path:

- use explicit path-option with specific addresses included/excluded
- use dynamic path option and change the interface IGP metric
- use dynamic path option and change the interface TE metric (administrative-weight)
- use dynamic path option and change the interface RSVP bandwidth or the tunnel bandwidth
- use dynamic path option and change the interface/tunnel affinity/attributes
- use a combination of explicit path-option and the above attributes

explicit-path

When using explit-paths in order to define the TE path, you can use either loopbacks or next-hop addresses as next-address. Also there is no need to define the last-hop, which is the tail-end. However, it's good practice to define next-hop addresses (plus the last-hop) if you want to be as strict as possible about the path.

i.e. for a path R2-R3-R4-R5-R6-R7 you can simply create the following explicit path on R2:

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng path-option 1 explicit name R3-R4-R5-R6
!
ip explicit-path name R3-R4-R5-R6 enable
 next-address R3
 next-address R4
 next-address R5
 next-address R6
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 2.2.2.2
 path-option 1 explicit name R3-R4-R5-R6
!
explicit-path name R3-R4-R5-R6
 index 1 next-address strict ipv4 unicast R3
 index 2 next-address strict ipv4 unicast R4
 index 3 next-address strict ipv4 unicast R5
 index 4 next-address strict ipv4 unicast R6
```

While editing the explicit-path, you can use "`index x`" in order to change a specific entry, or "`list`" to view all current entries.

When checking the output of "`sh mpls traffic-eng topology`" you can find the exact path (ip addresses from all ingress/egress interfaces) followed under the "Explicit Route" object (ERO).

When using the "`exclude-address`" option, the path is considered **semi-dynamic**. It works like a dynamic path (based on IGP), but specific hops are avoided.

bandwidth

You can define the tunnel required/signaled bandwidth and the interface available bandwidth in order to steer the traffic through a path that has enough bandwidth to accommodate the tunnel's need.

This is purely a control-plane reservation, in order to allow the TE tunnels to be established. In order to actually reserve that amount of bandwidth for the TE tunnel traffic, you have to configure the appropriate QoS too.

IOS

```
interface FastEthernet0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth 100000
!
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng bandwidth 99000
 tunnel mpls traffic-eng path-option 1 dynamic
```

IOS-XR

```
rsvp
 interface X
  bandwidth 100000
!
interface tunnel-te0
 ipv4 unnumbered Loopback0
 signalled-bandwidth 99000
 destination 2.2.2.2
 path-option 1 dynamic
```

You can use the **Path Option for Bandwidth Override** in order to provide a fallback path option that allows **overriding the bandwidth configured on the TE tunnel** (i.e a path option with bandwidth set to 0 removes the bandwidth constraint imposed by the constraint-based routing calculation).

When an LSP is signaled using a path option with a configured bandwidth, the bandwidth associated with the path option is signaled instead of the signaled bandwidth configured directly on the tunnel. You can have multiple path-options with different bandwidth requirements. All path-options should use the same type of bandwidth (global pool or sub-pool), otherwise reoptimization might not occur.

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 19.19.19.19
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 50000
 tunnel mpls traffic-eng path-option 1 explicit name R3-R4-R6-R5
 tunnel mpls traffic-eng path-option 2 dynamic bandwidth 25000
 tunnel mpls traffic-eng path-option 3 dynamic bandwidth 0
```

In the above example, 3 path-options are using 3 different bandwidth requirements:

- path-option 1 (explicit) requires 50000 kbps (default)
- path-option 2 (dynamic) requires 25000 kbps
- path-option 3 (dynamic) requires 0 kbps

With bandwidth override configured on a path option, the traffic engineering software attempts to reoptimize the bandwidth every 30 seconds.

TE metric

You can influence the path that the tunnel will follow by changing the TE metric (administrative-weight) **on the physical interfaces along the path**.

IOS

```
interface X
 mpls traffic-eng tunnels
 mpls traffic-eng administrative-weight 2
```

IOS-XR

```
mpls traffic-eng
 interface X
  admin-weight 2
```

Usually it's easier to increase the TE metric on the path that should not be preferred, instead of decreasing the metric on the path that should be preferred.

If TE metric is set to its max under a specific interface, then tunnel will never pass through that interface.

All TE metric changes require to shut/no shut the tunnel interface in order to have immediate effect.

Default TE metric is the IGP metric for that specific path. Whenever IGP metric changes, TE metric follows, unless it's explicitly configured, so it gets priority.

IOS

```
R1#sh mpls traffic-eng tunnel

P2P TUNNELS/LSPs:

Name: R1_t0                               (Tunnel0) Destination: 2.2.2.2
  Status:
    Admin: up         Oper: up     Path: valid       Signalling:
connected
    path option 1, type dynamic (Basis for Setup, path weight 23)
```

You can change this default path-selection policy (where TE metric is used) using the following command, either globally or under the specific tunnel:

IOS

```
mpls traffic-eng path-selection metric igp
!
interface Tunnel0
 tunnel mpls traffic-eng path-selection metric igp
```

IOS-XR

```
mpls traffic-eng
 path-selection metric igp
!
interface tunnel-te0
 path-selection metric igp
```

Use the following commands to verify the current metric type:

IOS

```
R1#sh mpls traffic-eng tunnels | i Metric
    Metric Type: TE (default)
R1#sh mpls traffic-eng tunnels | i Metric
    Metric Type: IGP (interface)
```

IOS-XR

```
GSR#sh mpls traffic-eng tunnels | i Metric
    Metric Type: TE (default)
```

path-options

You can define multiple path-options (as fall-back methods), with preference 1 having the highest priority. The currently active TE path can be found by checking the "`sh mpls traffic-eng tunnels`" command output.

IOS

```
interface Tunnel0
 tunnel mpls traffic-eng path-option 1 explicit name R3-R4-R5-R6
 tunnel mpls traffic-eng path-option 2 dynamic
```

IOS-XR

```
interface tunnel-te0
 path-option 1 explicit name R3-R4-R5-R6
 path-option 2 dynamic
```

IOS

```
R2#sh mpls traffic-eng tunnels

P2P TUNNELS/LSPs:

Name: R2_t0                              (Tunnel0) Destination: 7.7.7.7
  Status:
    Admin: up         Oper: up     Path: valid       Signalling:
connected
    path option 2, type dynamic (Basis for Setup, path weight 40)
    path option 1, type explicit R3-R4-R5-R6
```

...

You can also use "traceroute mpls traffic-eng" on a PE in order to verify the TE path unidirectionally:

IOS

```
R2#traceroute mpls traffic-eng tunnel 0
Tracing MPLS TE Label Switched Path on Tunnel0, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
  0 26.2.4.2 MRU 1500 [Labels: 22 Exp: 0]
L 1 26.2.4.4 MRU 1500 [Labels: 22 Exp: 0] 76 ms
L 2 26.4.5.5 MRU 1500 [Labels: 21 Exp: 0] 80 ms
L 3 26.5.6.6 MRU 1500 [Labels: 22 Exp: 0] 104 ms
L 4 26.3.6.3 MRU 1504 [Labels: implicit-null Exp: 0] 112 ms
! 5 26.3.19.19 140 ms
```

IOS-XR

```
GSR#traceroute mpls traffic-eng tunnel 0

Tracing MPLS TE Label Switched Path on tunnel-te0, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.

  0 2.2.78.8 MRU 1500 [Labels: 22 Exp: 0]
L 1 2.2.78.7 MRU 1500 [Labels: 23 Exp: 0] 41 ms
L 2 2.2.27.2 MRU 1504 [Labels: implicit-null Exp: 0] 126 ms
! 3 2.2.29.9 9 ms
```

MPLS traceroute in IOS-XR requires "mpls oam" to be enabled.

setup/hold priorities

Every LSP/tunnel has a setup and a hold priority, each one having values from 0 (best) to 7 (worst).

- setup priority
    - used when the LSP is being established
- hold priority
    - used when the LSP is already established

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng priority 4 4
 tunnel mpls traffic-eng path-option 1 dynamic
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 priority 4 4
 destination 2.2.2.2
 path-option 1 dynamic
```

Preemption

When an LSP is set up across a path where resources have been exhausted, its **setup priority is compared to the hold priority** of all the LSPs already using the resources, in order to determine if the new LSP can preempt an existing LSP and use its resources.

Do not use a hold priority that is worse than the setup priority, otherwise you might end up with continuous preemption (the software will probably prohibit you from doing so).

Using a high hold priority is recommended for stable environments.

When having primary and backup tunnels, it's good practice to assign better priorities to primary tunnels.

IOS-XR supports soft-preemption, which tries to preempt tunnels with the least possible disruption. You have to enable it under the tunnel and on every node.

LSP attributes

It provides the ability to configure values for several LSP-specific path-options for TE tunnels. One or more TE tunnels can specify specific path-options by referencing an LSP attribute list. You can configure LSP attributes lists with different sets of attributes for each path-option

Attributes that can be used:

- affinity (IOS, IOS-XR)
- auto-bw (IOS)
- bandwidth (IOS, IOS-XR)
- lockdown (IOS)
- priority (IOS)
- protection (IOS)
- record-route (IOS)

LSP attribute list values referenced by the path-option take precedence over the values configured on the tunnel interface.

IOS

```
mpls traffic-eng lsp attributes LSP-ATTR-1
 bandwidth 22222
 priority 6 6
!
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 19.19.19.19
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 50000
 tunnel mpls traffic-eng path-option 1 explicit name R3-R4-R6-R5
attributes LSP-ATTR-1
```

IOS-XR

```
mpls traffic-eng
 attribute-set path-option LSP-ATTR-1
  signalled-bandwidth 22222
!
interface tunnel-te0
 ipv4 unnumbered Loopback0
 signalled-bandwidth 50000
 destination 2.2.2.2
 path-option 1 dynamic attribute-set LSP-ATTR-1
```

IOS

```
R2#sh mpls traffic-eng lsp attributes
LIST LSP-ATTR-1
    bandwidth 22222
    priority 6 6
```

```
R2#sh mpls traffic-eng tunnels

P2P TUNNELS/LSPs:

Name: R2_t0                               (Tunnel0) Destination:
19.19.19.19
  Status:
    Admin: up         Oper: up     Path: valid       Signalling:
connected
    path option 1, type explicit R3-R4-R6-R5 (Basis for Setup, path
weight 50)

  Config Parameters:
    Bandwidth: 22222    kbps (Global)  Priority: 6  6   Affinity:
0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute announce: enabled  LockDown: disabled Loadshare: 22222
bw-based
    auto-bw: disabled
...
```

If when applying the LSP attributes to a path-option you get a warning like "`% Setup priority is inconsistent with popt x`", then you have to check the x path-option for a possible inconsistency.

TE LSP reoptimization

Tunnel reoptimization is the signaling of an new LSP that is more optimal (i.e. with a lower metric) than the current LSP a TE tunnel is using and the switching over of the tunnel's data to use this new LSP.

The new more optimal TE LSP is always established and the data moved onto it before the original LSP is torn down (using RSVP shared explicit reservation style), in order to ensure that no data packets are lost during the transition to the new LSP.

The TE LSPs reoptimization process is triggered under the following circumstances:

- periodically (based on a timer)
- manually (using the command "`mpls traffic-eng reoptimize`")
- after a network event (i.e. link-up)

Regardless of how reoptimization is triggered, the head-end router reoptimizes a tunnel only if it can find a better path than the one the tunnel currently uses. If there is no better path in the TE database (topology), then no new LSP is signaled and reoptimization does not occur.

You can use the "`lockdown`" keyword with any tunnel path-option if you want to disable reoptimization for a specific path-option of a tunnel.

Links

- IETF - RFC 4736

TE Tunnels for VPN

There is no need to have a single end-to-end LSP tunnel between two PEs. You can have:

- multiple TE tunnels (one after the other) between the PEs, using "stitching" on the intermediate P routers
- a combination of TE tunnels plus LDP sessions

When doing PE-to-P TE tunnels you must be careful of the PHP happening one hop before the P, exposing the bottom VPN label to it. One solution is to create an extra layer of labels between the PE and the P (**using a tLDP session across the tunnel**) in order to avoid exposing the VPN label and expose a transport label instead. Then, traffic can be forwarded from the P to the end PE and have the VPN label popped there without any issue.

Generally, for VPN traffic the following types of TE tunnels be used:

- **PE-to-PE tunnels**
  - high number of LSPs
  - no extra LDP required
- **P-to-P tunnels**
  - low number of LSPs
  - LDP over IP Tunnel or LDP over TE Tunnel is required
- **PE-to-P tunnels**
  - medium number of LSPs
  - tLDP over TE Tunnel is required

TE Routing & Traffic Selection

You can route through the TE tunnel using one of the following:

- **static route** (for the tunnel destination)
  - supported on inter-area tunnels
- **static route** (for a loopback on the tail-end)
  - supported on inter-area tunnels
- **autoroute announce**
  - the tunnel destination is announced automatically into the IGP of the head-end only
  - not supported on inter-area tunnels
- **autoroute destination**

    - the tunnel destination is announced automatically as a static route
    - supported on inter-area tunnels
- **forwarding adjacency**
    - the TE link is advertised like a normal link into the IGP of all routers
- **policy-based routing (PBR)**
    - traffic is forwarded across a TE tunnel based on a route-map criteria
    - supported on inter-area tunnels
- **policy-based tunnel selection (PBTS)**
    - traffic is forwarded across TE tunnels based on a single EXP value per tunnel
    - supported on inter-area tunnels
- **class-based tunnel selection (CBTS)**
    - traffic is forwarded across TE tunnels based on one or more EXP values per tunnel
    - supported on inter-area tunnels

MPLS L3VPN through MPLS TE Tunnel

If the BGP next-hop (for MPLS-VPN) is different from the MPLS-TE router-ID of the tail-end (i.e. if this is not a PE-PE TE tunnel), then you need to enable LDP over the tunnel (i.e. enable "`mpls ip`" on the TE tunnel interface).

static route

IOS

```
interface Tunnel0
 tunnel destination x.x.x.x
!
ip route x.x.x.x 255.255.255.255 Tunnel0
```

IOS-XR

```
interface tunnel-te0
 destination y.y.y.y
!
router static
 address-family ipv4 unicast
  y.y.y.y/32 tunnel-te0
```

You can also use a static route for a loopback on the tail-end, and then use that loopback for BGP next-hop under a specific VRF.

IOS

PE-1

```
vrf definition VPN
 address-family ipv4
  bgp next-hop Loopback100
 exit-address-family
```

```
!
interface Loopback100
 ip address 100.100.100.1 255.255.255.255
!
ip route 100.100.100.2 255.255.255.255 Tunnel0
```

PE-2

```
vrf definition VPN
 address-family ipv4
  bgp next-hop Loopback100
 exit-address-family
!
interface Loopback100
 ip address 100.100.100.2 255.255.255.255
!
ip route 100.100.100.1 255.255.255.255 Tunnel0
```

autoroute announce

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng path-option 1 dynamic

R1#sh ip route 2.2.2.2
Routing entry for 2.2.2.2/32
  Known via "ospf 1", distance 110, metric 4, type intra area
  Last update from 2.2.2.2on Tunnel0, 00:03:39 ago
  Routing Descriptor Blocks:
  * 2.2.2.2, from 2.2.2.2, 00:03:39 ago, via Tunnel0
      Route metric is 4, traffic share count is 1
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 autoroute announce
 destination 2.2.2.2
 path-option 1 dynamic

GSR#sh route 2.2.2.2
Routing entry for 2.2.0.9/32
  Known via "isis 1", distance 115, metric 30, type level-1
  Installed Jan 16 13:29:24.723 for 01:11:17
  Routing Descriptor Blocks
    2.2.2.2, from 2.2.2.2, via tunnel-te0
      Route metric is 30
  No advertising protos.
```

autoroute destination

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng autoroute destination
 tunnel mpls traffic-eng path-option 1 dynamic
```

```
R1#sh ip route 2.2.2.2
Routing entry for 2.2.2.2/32
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Tunnel0
      Route metric is 0, traffic share count is 1
```

IOS-XR

not supported

If there is another static route manually configured, then traffic is load-balanced (unless a different metric is used).

If both "autoroute announce" and "autoroute destination" are configured, then "autoroute destination" becomes active, due to the static route having precedence over IGP in the RIB.

It's obvious that if you want to route other than the tunnel destination traffic through the tunnel, the autoroute options won't help you.

forwarding adjacency

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 10.0.0.1
 tunnel mpls traffic-eng forwarding-adjacency
 tunnel mpls traffic-eng path-option 1 dynamic
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 10.0.0.2
 forwarding-adjacency
 path-option 1 dynamic
```

LSP tunnel must be bidirectional.

If you don't set the IGP metric under the tunnel interface, the default IGP metric will be used (which might not provide the expected results). In any case you have to be very careful of loops.

Autoroute announce should not be used together with forwarding-adjacency.

Although a bidirectional tunnel is required, there is no actual IGP adjacency established over the TE tunnel.

An easy way to verify the existence of the tunnel in the ISIS topology is to use the following command:

IOS

```
R2#sh isis topology | i System|TE
System Id            Metric     Next-Hop             Interface   SNPA
R5                   19         XR1                  Tu0         *MPLS
TE-Tunnel
R6                   19         XR1                  Tu0         *MPLS
TE-Tunnel
XR1                  9          XR1                  Tu0         *MPLS
TE-Tunnel
```

Policy-Based Routing (PBR)

IOS

```
interface FastEthernet1/0
 ip policy route-map TO-TUNNEL-ROUTEMAP
!
route-map TO-TUNNEL-ROUTEMAP permit 10
 set interface Tunnel0
```

PBR for VRF traffic is not supported. You can use static route + vrf bgp next-hop in order to forward each VRF to a different TE tunnel.

CBTS (Class-Based Tunnel Selection)

It's EXP-based tunnel selection between multiple tunnels to same destination.

CBTS does not allow load-balancing of a given EXP value in multiple tunnels.

CBTS is not supported with AToM and MPLS TE Automesh.

If the input QoS policy modifies the EXP value, CBTS uses the modified EXP value for its tunnel selection.
If the output QoS policy modifies the EXP value, CBTS uses the original EXP value for its tunnel selection.

Tunnel Selection

All comparisons are made per tunnel prefix (static route, autoroute).
Only tunnels with the best metric are used for CBTS.

- specific EXP is configured on a single tunnel and there is a single default EXP tunnel
  - traffic with that EXP is forwarded into the specific EXP tunnel
  - traffic with other EXPs is forwarded into the default EXP tunnel
- specific EXP is configured on multiple tunnels and there is a single default EXP tunnel
  - traffic with that EXP is forwarded into one of the specific EXP tunnels (based on lowest EXP and lowest tunnel ID)
  - traffic with other EXPs is forwarded into the default EXP tunnel
- specific EXPs are configured on multiple tunnels and there is no default EXP tunnel
  - traffic with a specific EXP is forwarded into the specific EXP tunnel
  - traffic with other EXPs is forwarded into the EXP tunnel with the lowest EXP value
- specific EXP is not configured on any tunnel and there is a single default EXP tunnel
  - traffic with any EXP is forwarded into the default EXP tunnel
- specific EXP is not configured on any tunnel and there are multiple default EXP tunnels
  - traffic with any EXP is forwarded (arbitrarily) into one of the default EXP tunnels
- specific EXP is not configured on any tunnel and there is no default EXP tunnel
  - traffic with any EXP is load-balanced across all tunnels

Since static route has precedence over (dynamic) autoroute, if two tunnels for the same prefix use static route and autoroute accordingly, then only the one with the static route is chosen for the CBTS, regardless of the EXP values configured in them. This is also applicable and on same-type tunnels (i.e. autoroute ones) with different metrics, where only the one with the best metric is used by CBTS.

Configuration

- Tunnel master bundles tunnel members
- Tunnel selection (autoroute, etc.) configured on tunnel master
- Bundle members configured with EXP values to carry

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng path-option 1 dynamic
 tunnel mpls traffic-eng exp 3 5
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng path-option 1 dynamic
 tunnel mpls traffic-eng exp default
!
interface Tunnel1000
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2.2
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng exp-bundle master
 tunnel mpls traffic-eng exp-bundle member Tunnel0
 tunnel mpls traffic-eng exp-bundle member Tunnel1
```

IOS

```
R1#sh mpls traffic-eng exp

Destination: 2.2.2.2

   Master: Tunnel1000        Status:  up

   Members       Status                Conf Exp                  Actual
Exp

   Tunnel0       up   (Active)         3 5                       3 5
   Tunnel1       up   (Active)         Default                   0 1 2 4 6
7


(D)  : Destination is different
(NE): Exp values not configured on tunnel
```

IOS-XR

not supported

Policy-Based Tunnel Selection (PBTS)

It's EXP-based tunnel selection between multiple tunnels to same destination.

Similar to CBTS, but only one class per tunnel is supported and no tunnel-bundle is required.

PBTS has precedence over CBTS.

Configuration

- Tunnels configured via policy-class with one EXP value to carry
- Tunnel without policy-class acts as default

IOS
not supported

IOS-XR
```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 2.2.2.2
 policy-class 5
 path-option 1 dynamic
!
interface tunnel-te1
 ipv4 unnumbered Loopback0
 destination 2.2.2.2
 policy-class default
 path-option 1 dynamic
```

Advanced MPLS-TE

FRR (Fast ReRoute) extensions for RSVP-TE are defined in RFC 4090.
Inter-Area MPLS TE is described in RFC 4105.
Inter-AS MPLS TE is described in RFC 4216.

LSP Protection

- path protection - long term
- local protection (FRR) - short term
  - link protection
  - node protection

Backup tunnels are usually used for a short period of time, until the head-end recomputes and signals a new path. When the new path is established and traffic is switched over to it, the backup path is torn down.

PLR = Point of Local Repair
MP = Merge Point

The constraints of the backup path are signaled by the head-end to the PLR using the Fast Reroute Object (FRO) of RSVP.

- many-to-one backup (facility backup)
  - an extra label is pushed by the PLR
  - MP advertises an implicit-null label (PHP)
  - traffic arrives at the MP with the same label as the main path
- one-to-one backup (detour backup)
  - the top label is swapped by the PLR
  - MP advertises a normal label (label swap)
  - traffic arrives at the MP with a different label from the main path

Expect to see only the many-to-one backup being used.

The failure and the local restoration of a main path are signaled by the PLR to the head-end using an RSVP Path-Error message (code="Notify", subcode="Tunnel locally repaired") and an RRO flag.

FRR (Fast Reroute)

MPLS-TE/FRR is a mechanism for protecting MPLS TE LSPs from link and node failures by **locally repairing the LSPs at the point of failure**. It allow traffic to continue to flow on the LSPs, while their head-end routers attempt to establish new end-to-end LSPs to replace them.

FRR repairs locally the protected LSPs by rerouting them over backup tunnels that bypass these failed links or nodes.

- NHOP backup tunnels
    - backup tunnels that bypass only a link of the LSP's path
    - they provide link protection by **rerouting the LSP's traffic to the next-hop**
    - can be specified by simply **excluding the protected link** (interface address)
- NNHOP backup tunnels
    - backup tunnels that bypass a node of the LSP's path
    - they provide node (& link) protection by **rerouting the LSP's traffic to the next-next-hop**
    - can be specified by simply **excluding the protected node** (loopback address)

FRR prefers NNHOP over NHOP backup tunnels, when both are available.

Configuration Steps

- PE (head-end)
    - enable FRR under the tunnel
- PLR (point of local repair)
    - configure a backup tunnel with a path that avoids the link/node to be protected
    - enable the backup tunnel under the link to be protected
    - enable RSVP hellos or BFD under the link to be protected (for faster detection)

PE (head-end)

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 19.19.19.19
 tunnel mpls traffic-eng path-option 1 explicit name X
 tunnel mpls traffic-eng fast-reroute
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 19.19.19.19
 fast-reroute
 path-option 1 explicit name X
```

You also have the following options if you want to influence the backup tunnel selection on the PLR:

IOS

```
R2(config-if)#tunnel mpls traffic-eng fast-reroute ?
  bw-protect    bandwidth protection desired
  node-protect  node protection desired
```

IOS-XR

```
CRS(config-if)#fast-reroute protect ?
  bandwidth  Enable bandwidth protection request
  node       Enable node protection request
```

IOS-XR by default tries to provide the best available protection, even if the above two options aren't activated.

PLR (point of local repair)

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 5.5.5.5
 tunnel mpls traffic-eng path-option 1 explicit name AVOID-X
!
ip explicit-path name AVOID-X enable
 exclude-address x.x.x.x
!
interface FastEthernet0/0
 mpls traffic-eng backup-path Tunnel0
 ip rsvp signalling hello
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 5.5.5.5
 path-option 1 explicit name AVOID-X
!
explicit-path name AVOID-X
 index 1 exclude-address ipv4 unicast x.x.x.x
!
mpls traffic-eng
 interface TenGigE0/0/0/0
  backup-path tunnel-te 0
```

The same "`rsvp signalling hello`" type (RSVP hellos or BFD) must be configured on the other side too for fast detection to take place.

IOS-XR supports only BFD for fast detection (< 1sec). RSVP hellos can be used for simple reroute (detection > 10sec) or GR.

Verification

PE (head-end)

IOS

```
R2#sh mpls traffic-eng tunnels protection
P2P TUNNELS:
R2_t0
  LSP Head, Tunnel0, Admin: up, Oper: up
  Src 2.2.2.2, Dest 19.19.19.19, Instance 46
  Fast Reroute Protection: Requested
    Outbound: Unprotected: no backup tunnel assigned
      LSP signalling info:
        Original: out i/f: Fa0/0.23, label: 18, nhop: 20.2.3.3
                  nnhop: 4.4.4.4; nnhop rtr id: 4.4.4.4
  Path Protection: None
```

PLR (point of local repair)

before the activation of the backup tunnel

IOS

```
R4#sh mpls traffic-eng tunnels backup
R4_t0
  LSP Head, Admin: up, Oper: up
  Tun ID: 0, LSP ID: 3, Source: 4.4.4.4
  Destination: 5.5.5.5
  Fast Reroute Backup Provided:
    Protected i/fs: Fa0/0.45
    Protected LSPs/Sub-LSPs: 1, Active: 0
    Backup BW: any pool unlimited; inuse: 0 kbps
    Backup flags: 0x0
```

```
R4#sh mpls traffic-eng fast-reroute database role middle

P2P LSP midpoint frr information:
LSP identifier                 In-label Out intf/label   FRR intf/label
Status
---------------------------    -------- --------------   --------------
------
2.2.2.2 0 [46]                 16       Fa0/0.45:18      Tu0:18
Ready
```

```
R4#sh mpls traffic-eng fast-reroute database role middle detail
FRR Database Summary:
  Protected interfaces    : 1
  Protected LSPs/Sub-LSPs : 1
  Backup tunnels          : 1
  Active interfaces       : 0

P2P LSPs:

 Tun ID: 0, LSP ID: 46, Source: 2.2.2.2
 Destination: 19.19.19.19
  State        : Ready
  InLabel      : 16
  OutLabel     : Fa0/0.45:18
  FRR OutLabel : Tu0:18
```

after the the activation of the backup tunnel

```
R4#sh mpls traffic-eng tunnels backup
R4_t0
  LSP Head, Admin: up, Oper: up
  Tun ID: 0, LSP ID: 3, Source: 4.4.4.4
  Destination: 5.5.5.5
  Fast Reroute Backup Provided:
    Protected i/fs: Fa0/0.45
    Protected LSPs/Sub-LSPs: 1, Active: 1
    Backup BW: any pool unlimited; inuse: 0 kbps
    Backup flags: 0x0


R4#sh mpls traffic-eng fast-reroute database role middle

P2P LSP midpoint frr information:
LSP identifier                 In-label Out intf/label   FRR intf/label
Status
---------------------------    -------- --------------   --------------
------
2.2.2.2 0 [46]                 16       Fa0/0.45:18      Tu0:18
Active

R4#sh mpls traffic-eng fast-reroute database role middle detail
FRR Database Summary:
  Protected interfaces    : 1
  Protected LSPs/Sub-LSPs : 1
  Backup tunnels          : 1
  Active interfaces       : 1

P2P LSPs:
 Tun ID: 0, LSP ID: 46, Source: 2.2.2.2
 Destination: 19.19.19.19
  State        : Active
  InLabel      : 16
  OutLabel     : Fa0/0.45:18
  FRR OutLabel : Tu0:18
```

When using MPLS-TE/FRR for node protection, a label_recording flag is included into the SAO in order to signal that the RRO should also include labels (because these labels are actually used by the PLR for NNHOP backup tunnels), although it's not shown in the tunnel config.

IOS

```
R2#sh mpls traffic-eng tunnels | b Resv
    RSVP Resv Info:
      Record   Route:  3.3.3.3(20) 4.4.4.4(20)
                       5.5.5.5(20) 19.19.19.19(0)
```

If you explicitly enable it in the config, then you get the address of the physical interfaces too.

```
R2#sh mpls traffic-eng tunnels | b Resv
    RSVP Resv Info:
      Record   Route:  3.3.3.3(21) 20.3.4.3(21)
                       4.4.4.4(21) 20.4.5.4(21)
                       5.5.5.5(21) 20.5.19.5(21)
                       19.19.19.19(0) 20.5.19.19(0)
```

The path when using the backup tunnel is not reflected in the above output.

You can also use the following command on the PLR in order to verify the FRR operation.

IOS

```
R4#sh ip rsvp sender detail
...
  Tun Dest:   19.19.19.19  Tun ID: 0  Ext Tun ID: 2.2.2.2
  Tun Sender: 2.2.2.2  LSP ID: 25
...
  Fast-Reroute Backup info:
    Inbound  FRR: Not active
    Outbound FRR: Ready -- backup tunnel selected
      Backup Tunnel: Tu1        (label 19)
      Bkup Sender Template:
        Tun Sender: 20.3.4.4  LSP ID: 25
      Bkup FilerSpec:
        Tun Sender: 20.3.4.4, LSP ID: 25
```

or the following command on the head-end.

```
R2#sh ip rsvp reservation detail
...
Reservation:
  Tun Dest:   19.19.19.19  Tun ID: 0  Ext Tun ID: 2.2.2.2
  Tun Sender: 2.2.2.2  LSP ID: 25
  Next Hop: 20.2.3.3 on FastEthernet0/0.23
  Label: 20 (outgoing)
  Reservation Style is Shared-Explicit, QoS Service is Controlled-Load
  Resv ID handle: 0A000415.
  Created: 13:32:42 UTC Sat Jan 11 2014
  Average Bitrate is 20M bits/sec, Maximum Burst is 1K bytes
  Min Policed Unit: 0 bytes, Max Pkt Size: 1500 bytes
  RRO:
    3.3.3.3/32, Flags:0x20 (No Local Protection, Node-id)
      Label subobject: Flags 0x1, C-Type 1, Label 20
    20.3.4.3/32, Flags:0x0 (No Local Protection)
      Label subobject: Flags 0x1, C-Type 1, Label 20
    4.4.4.4/32, Flags:0x25 (Local Prot Avail/Has BW/to NHOP, Node-id)
      Label subobject: Flags 0x1, C-Type 1, Label 19
    20.4.5.4/32, Flags:0x5 (Local Prot Avail/Has BW/to NHOP)
      Label subobject: Flags 0x1, C-Type 1, Label 19
    5.5.5.5/32, Flags:0x20 (No Local Protection, Node-id)
      Label subobject: Flags 0x1, C-Type 1, Label 19
    20.5.19.5/32, Flags:0x0 (No Local Protection)
      Label subobject: Flags 0x1, C-Type 1, Label 19
    19.19.19.19/32, Flags:0x20 (No Local Protection, Node-id)
      Label subobject: Flags 0x1, C-Type 1, Label 0
    20.5.19.19/32, Flags:0x0 (No Local Protection)
      Label subobject: Flags 0x1, C-Type 1, Label 0
  Status:
  Policy: Accepted. Policy source(s): MPLS/TE
```

Then the backup tunnel gets activated, the output will change to:

IOS

```
R2#sh ip rsvp reservation detail
...
    4.4.4.4/32, Flags:0x23 (Local Prot Avail/In Use/to NHOP, Node-id)
      Label subobject: Flags 0x1, C-Type 1, Label 16
    20.3.4.4/32, Flags:0x3 (Local Prot Avail/In Use/to NHOP)
      Label subobject: Flags 0x1, C-Type 1, Label 16
```

Path Protection

Apart from having link & node protection using FRR, you also have the option of enabling **path protection end-to-end for a tunnel**. You can have multiple backup path options per primary path option.

It's not as fast as FRR, but it's still faster than having a second path-option or relying on IGP, because **it pre-calculates the whole backup path** (like FRR).

It can be used with intra-area, inter-area and inter-AS scenarios.

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 2.2.2
 tunnel mpls traffic-eng path-option 1 explicit name PATH1
 tunnel mpls traffic-eng path-option protect 1 explicit name PATH1-BAK
```

```
R1#sh mpls traffic-eng tunnels protection

P2P TUNNELS:
R1_t0
  LSP Head, Tunnel0, Admin: up, Oper: up
  Src 1.1.1.1, Dest 2.2.2.2, Instance 3
  Fast Reroute Protection: None
  Path Protection: Requested
```

You can also view the details about SRLGs:

```
R1#sh mpls traffic-eng tunnels tunnel0

Name: R1_t0                             (Tunnel0) Destination: 2.2.2.2
  Status:
    Admin: up         Oper: up     Path: valid       Signalling:
connected
    path option 1, type explicit PATH1 (Basis for Setup, path weight 20)
    Path Protection: 2 Common Link(s), 1 Common Node(s)
      Link Sharing Detail:
        P2P Links:             0
        Multiacces Links:      2
           Both interfaces:         2
           1 interface:             0
           0 interfaces:            0 (only media is shared)
    path protect option 1, type explicit PATH1-BAK (Basis for Protect,
path weight 20)
```

Latest IOS software releases support enhanced path protection (up to 8 secondary path options).

IOS-XR

```
interface tunnel-te0
 path-protection
 path-option 1 explicit name R2-R3-R4
 path-option 2 dynamic
!
interface tunnel-te1
 path-protection
 path-option 1 explicit name R2-R3-R4 protected-by 2
 path-option 2 explicit name R2-R5-R6
```

Path-protection is not supported on C12k platform for IOS-XR.

In IOS-XR you can either have explicit path as primary and dynamic as backup, or explicit path as primary and backup. IOS-XR provides automatic path diversity.

Backup Tunnel Selection

You can have multiple backup tunnels protecting the same or different LSPs, each one terminating on the same destination or a different one.

The backup tunnel is configured on the PLR, under the physical interface to be protected.

IOS

```
interface X
 mpls traffic-eng backup-path Tunnel0
 mpls traffic-eng backup-path Tunnel1
```

IOS-XR

```
mpls traffic-eng
 interface X
  backup-path tunnel-te 0
  backup-path tunnel-te 1
```

The selection priority between multiple backup tunnels is based on the following table.

Preference	Backup Tunnel Destination	Bandwidth Pool	Bandwidth Amount
1 (Best)	Next-next hop	Sub-pool or global-pool	Limited
2	Next-next hop	Any	Limited
3	Next-next hop	Sub-pool or global-pool	Unlimited
4	Next-next hop	Any	Unlimited
5	Next-hop	Sub-pool or global-pool	Limited
6	Next-hop	Any	Limited
7	Next-hop	Sub-pool or global-pool	Unlimited
8 (Worst)	Next-hop	Any	Unlimited

The general idea is the following:

- NNHOP tunnels have priority over NHOP tunnels
- backup-bw tunnels have priority over no backup-bw tunnels
- specific pool tunnels have priority over any pool tunnels

After a backup tunnel has been chosen for an LSP, various conditions may change on the network, that will cause the router to reevaluate this choice. This promotion can happen immediately in case of a backup tunnel going up/down, or periodically every 5 mins. This time be changed by using the command "`mpls traffic-eng fast-reroute timers`". Set it to 0 if you want to disable promotion for LSPs.

backup protection algorithm

If the main tunnel doesn't have any bandwidth configured, then it can use only backup tunnels that don't have any backup-bandwidth configured.

In the following example, main tunnel (that doesn't have any signaled bandwidth configured) can use only backup tunnel #2.

main tunnel

```
interface Tunnel0
```

backup tunnel #1

```
interface Tunnel1
 tunnel mpls traffic-eng backup-bw 30000
```

backup tunnel #2

```
interface Tunnel2
```

If the main tunnel has a bandwidth configured, then it can use only backup tunnels that either don't have any backup-bandwidth configured or have sufficient backup-bandwidth configured.

In the following example, main tunnel (that has a bandwidth of 35M configured) can use only backup tunnel #2 and #3.

main tunnel

```
interface Tunnel0
 tunnel mpls traffic-eng bandwidth 35000
```

backup tunnel #1

```
interface Tunnel1
 tunnel mpls traffic-eng backup-bw 30000
```

backup tunnel #2

```
interface Tunnel2
```

backup tunnel #3

```
interface Tunnel3
 tunnel mpls traffic-eng backup-bw 40000
```

Generally, the backup-bandwidth is assumed to be unlimited when it's not configured. Is this case no bandwidth protection is provided.

If you don't configure a global-pool or a sub-pool, then any pool is assumed.

When using global-pool or sub-pool in order to **limit the type of backup-bandwidth** (and not the amount), then you have to explicitly **define as unlimited the backup-bandwidth**, like below:

IOS

```
interface Tunnel1
 tunnel mpls traffic-eng backup-bw global-pool unlimited
 tunnel mpls traffic-eng backup-bw sub-pool unlimited
```

IOS-XR

```
interface tunnel-te1
 backup-bw global-pool unlimited
 backup-bw sub-pool unlimited
```

If there are two or more **backup tunnels with backup-bandwith** configured that have sufficient available backup-bandwith, then the one with the **least amount of currently available backup bandwidth** is chosen, in order to keep the largest amount available for other LSPs (main tunnels) with bigger needs.

If there are two or more **backup tunnels with no backup-bandwith** configured, then the one with the **least amount of currently used backup bandwidth** is chosen, in order to distribute the LSPs (main tunnels) as evenly as possible. If the LSP (main tunnel) doesn't have any bandwidth configured, the backup tunnel with the fewest protecting LSPs is chosen.

If you want to change the above backup protection preemption algorithm, you can use the following command on the PLR:

IOS

```
R2(config)#mpls traffic-eng fast-reroute backup-prot-preempt ?
  optimize-bw  Reduce bandwidth wastage (default: minimize LSPs
preempted)
```

Keep in mind that backup tunnels can have their **signaled-bandwidth** configured, just like any other tunnel, in order to have the LSRs on the backup path do the appropriate admission control (in production networks this might not be recommended in order to avoid reservation of resources for a tunnel rarely used). On the other hand, the **backup-bandwidth** is used solely by the PLR (head-end of he backup tunnel) locally. Theoretically, **both signaled-bandwidth and backup-bandwidth should be the same for proper operation.**

In general:

- signaled-bandwidth on the main tunnel
  - xxx (global pool is assumed)
  - sub-pool xxx
- backup-bandwidth on the backup tunnel
  - xxx (any pool is assumed)
  - global-pool xxx
  - sub-pool xxx

You need to enable RDM for RSVP if you are using sub-pools.

TE auto-bandwidth

TE auto-bandwidth samples the average output rate for each tunnel configured for automatic bandwidth adjustment and periodically (i.e. once per day) adjusts the tunnel's signaled-bandwidth to be the largest sample for the tunnel since the last adjustment.

Auto-bandwidth basically performs the following steps:

- every X interval, traffic over a tunnel is measured
- every Y interval, the largest sample from the above measurement is collected
- if this sample value exceeds a Z threshold, then it's used to set the new tunnel bandwidth

Configuration Steps

- globally
  - enable statistics collection & auto-bw adjustments (enabled by default)
  - frequency: define the interval between tunnel bw statistics collection (5m by default)
- per tunnel
  - frequency: define the interval between auto-bw adjustments (24h by default)

  - max-bw: define the max auto-bw allowed (no limit by default)
  - min-bw: define the min auto-bw allowed (no limit by default)
  - adjustment-threshold: define the delta threshold over which the new bw is actually signaled/used (10% by default)

IOS

```
mpls traffic-eng auto-bw timers frequency 60
!
interface Tunnel0
 load-interval 30
 tunnel mpls traffic-eng bandwidth 20
 tunnel mpls traffic-eng auto-bw frequency 300 adjustment-threshold 1
max-bw 100 min-bw 1
```

IOS-XR

```
mpls traffic-eng
 auto-bw collect frequency 1
!
interface tunnel-te0
 load-interval 30
 signalled-bandwidth 20
 auto-bw
  bw-limit min 1 max 100
  adjustment-threshold 1
  application 5
```

IOS-XR auto-bw intervals are defined in mins, while IOS ones are defined in secs.

It's good practice to have: tunnel load-interval < global sampling frequency < tunnel adjust frequency.

IOS

```
R2#sh mpls traffic-eng tunnels
...
  Config Parameters:
    Bandwidth: 20     kbps (Global)  Priority: 7  7   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute announce: enabled  LockDown: disabled Loadshare: 1
bw-based
    auto-bw: (300/155) 7  Bandwidth Requested: 1
          Adjustment Threshold: 1%
          Samples Missed: 0  Samples Collected: 2
```

after the auto-bw adjustment interval timer expires, the largest sample (**7**) is now requested/used

```
R2#sh mpls traffic-eng tunnels
...
  Config Parameters:
    Bandwidth: 20    kbps (Global)  Priority: 7  7   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute announce: enabled  LockDown: disabled Loadshare: 7
bw-based
    auto-bw: (300/283) 0  Bandwidth Requested: 7
          Adjustment Threshold: 1%
          Samples Missed: 0  Samples Collected: 0
```

Don't forget to also configure the load-interval on the tunnel interface accordingly.

Traffic samples for tunnels that were down during the sampling period, are ignored.

If auto-bw is configured for a tunnel, the "`tunnel mpls traffic-eng bandwidth`" command sets only the initial tunnel bandwidth. Setting the initial bandwidth is not required for auto-bw to work.

The latest requested/signaled bandwidth is automatically saved to the tunnel running config.

In latest software releases you can also set an **overflow** or **underflow threshold** in order to make the bandwidth adjustment earlier than the configured interval (so you don't have to wait for all the samples to be collected). That way you can set a large auto-bw adjustment frequency and use these two thresholds in such a way that you can **react quickly to traffic changing conditions**.

Auto-Tunnels

3 types:

- backup auto-tunnels
    - no need to configure manual backup tunnels
    - no need to assign backup tunnels to a protected interface
- primary one-hop auto-tunnels
    - no need to configure tunnels to directly connected neighbors with FRR for protection
- mesh-group auto-tunnels
    - no need to configure tunnel to all interested PEs with FRR for protection

backup auto-tunnels

It must be configured on the PLR, like normal backup tunnels. After activation, **dynamic backup NHOP/NNHOP tunnels are created automatically whenever an LSP requests FRR protection**.

If there is a static configured backup tunnel for a specific interface, then backup auto-tunnels won't be created for this.

IOS

```
mpls traffic-eng auto-tunnel backup
```

IOS-XR (4.0.0)

```
mpls traffic-eng
 auto-tunnel backup
```

Extra options available:

IOS

```
R3(config)#mpls traffic-eng auto-tunnel backup ?
  config      Config commands to apply to all backup auto-tunnels
  nhop-only   Automatically create n-hop backup tunnels only
  srlg        Shared Risk Link Groups influence backup tunnel path
selection
  timers      Configure timers for backup auto-tunnels
  tunnel-num  Configure tunnel I/F numbers for backup auto-tunnels
```

You can enable "debug mpls traffic-eng auto-tunnel backup all" if you want to see the actual cli commands used to create the backup auto-tunnels.

IOS

```
R3#debug mpls traffic-eng auto-tunnel backup all
MPLS TE Auto-Tunnel Backup all debugging is on

TE_AUTO_TUN: backup CLI command:
interface tunnel65436
no logging event link-status
ip unnumbered Loopback0
tunnel destination 6.6.6.6
tunnel mode mpls traffic-eng
end

TE_AUTO_TUN: CLI command:
ip explicit-path name __dynamic_tunnel65436
  index 1 exclude-address 20.3.6.3

TE_AUTO_TUN: backup CLI command:
interface tunnel65436
tunnel mpls traffic-eng path-option 1 exp name __dynamic_tunnel65436
end

TE_AUTO_TUN: backup CLI command:
interface tunnel65437
no logging event link-status
```

```
ip unnumbered Loopback0
tunnel destination 5.5.5.5
tunnel mode mpls traffic-eng
end

TE_AUTO_TUN: CLI command:
ip explicit-path name __dynamic_tunnel65437
  index 1 exclude-address 6.6.6.6

TE_AUTO_TUN: backup CLI command:
interface tunnel65437
tunnel mpls traffic-eng path-option 1 exp name __dynamic_tunnel65437
end


R3#sh mpls traffic-eng tunnels backup
R3_t65436
  LSP Head, Admin: up, Oper: up
  Tun ID: 65436, LSP ID: 1, Source: 3.3.3.3
  Destination: 6.6.6.6
  Fast Reroute Backup Provided:
    Protected i/fs: Fa0/0.36
    Protected LSPs/Sub-LSPs: 0, Active: 0
    Backup BW: any pool unlimited; inuse: 0 kbps
    Backup flags: 0x0
R3_t65437
  LSP Head, Admin: up, Oper: up
  Tun ID: 65437, LSP ID: 1, Source: 3.3.3.3
  Destination: 5.5.5.5
  Fast Reroute Backup Provided:
    Protected i/fs: Fa0/0.36
    Protected LSPs/Sub-LSPs: 1, Active: 0
    Backup BW: any pool unlimited; inuse: 0 kbps
    Backup flags: 0x0
```

primary one-hop auto-tunnels

It must be configured on every router you want to create a TE tunnel to all the TE-enabled routers that are one hop away. Each auto-tunnel has "`autoroute announce`" and "`fast-reroute`" activated by default, but you can add some extra configuration too.

That way you can create automatically multiple tunnels covering all possible paths in a network.

IOS

```
mpls traffic-eng auto-tunnel primary onehop
```

IOS-XR

not supported

Extra options available:

IOS

```
R1(config)#mpls traffic-eng auto-tunnel primary ?
  config      Config commands to apply to all primary auto-tunnels
  onehop      Automatically create tunnel to all next-hops
  timers      Configure timers for backup auto-tunnels
  tunnel-num  Configure tunnel I/F numbers for primary auto-tunnels
```

You can enable "`debug mpls traffic-eng auto-tunnel primary all`" if you want to see the actual cli commands used to create the primary one-hop auto-tunnels.

IOS

```
R1#debug mpls traffic-eng auto-tunnel primary all
MPLS TE Auto-Tunnel Primary all debugging is on

TE_AUTO_TUN: Found a new router id 2.2.2.2 off of FastEthernet0/0
TE_AUTO_TUN: primary CLI command:
interface tunnel65336
no logging event link-status
ip unnumbered Loopback0
tunnel destination 2.2.2.2
tunnel mode mpls traffic-eng
end

TE_AUTO_TUN: primary CLI command:
interface tunnel65336
tunnel mpls traffic-eng autoroute announce
tunnel mpls traffic-eng fast-reroute
end

TE_AUTO_TUN: CLI command:
ip explicit-path name __dynamic_tunnel65336
  index 1 next-address 10.1.2.1

TE_AUTO_TUN: primary CLI command:
interface tunnel65336
tunnel mpls traffic-eng path-option 1 exp name __dynamic_tunnel65336
end

TE_AUTO_TUN: Found  Down Tunnel65336 to router id 2.2.2.2 out
FastEthernet0/0
```

auto-tunnel mesh groups

It must be configured on the PE routers. After activation, it creates a mesh of TE tunnels to all the PE routers which are either configured for the **same mesh-group** (assuming OSPF is used), or their **TE Id is matched by an ACL**. The address that are used in the ACL must exist in the TE database in order to have the auto-tunnel created..

IOS

```
mpls traffic-eng auto-tunnel mesh
!
access-list 1 permit 20.20.20.20
access-list 1 permit 30.30.30.30
!
interface Auto-Template1
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination access-list 1
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng path-option 1 dynamic
```

The interface auto-template is configured like a normal TE tunnel.

You can also use explicit paths, which use the exclude-address keyword.

IOS-XR (4.1.1)

```
mpls traffic-eng
 auto-tunnel mesh
  group 99
   destination-list 1
!
ipv4 prefix-list 1
 10 permit 20.20.20.20/32
 20 permit 30.30.30.30/32
```

You cannot configure Inter-Area or Inter-AS tunnels for auto-tunnel mesh groups.

You cannot configure a static route to route traffic over auto-tunnel mesh group TE tunnels. You should use only the autoroute for tunnel selection.

In IOS-XR there is no need to configure a dynamic path-option; it's enabled by default.

When using mesh-group numbers (instead of an ACL) to match the PEs, you have to enable IGP flooding of mesh information under the IGP of the PEs.

IOS

```
interface Auto-Template1
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination mesh-group 99
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng path-option 1 dynamic
!
router ospf 1
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 mpls traffic-eng mesh-group 99 Loopback0 area 0


R2#sh mpls traffic-eng topology | i mg
Area mg-id's:
: mg-id 99    1.1.1.1 :

R2#sh mpls traffic-eng auto-tunnel mesh

Auto-Template1:

 Using mesh-group 99 to clone the following tunnel interfaces:

  Destination          Interface
  -----------          ---------

  1.1.1.1              Tunnel64336

Mesh tunnel interface numbers: min 64336 max 65335
```

IS-IS does not support IGP distribution of mesh group information.You have to use ACLs in this case.

You can also define manually the min/max tunnel numbers used.

SRLG (Shared Risk Link Groups)

When enabled it allows you to define links that belong to the same group, because they share a risk (i.e two links using the same fiber path).

The idea is to have backup tunnels avoid using links in the same SRLG as the interfaces they are protecting. Otherwise, when the protected link fails the backup tunnel will fail too.

Configuration

- assign interfaces to SRLGs
- enable backup auto-tunnels globally (IOS) or per interface (IOS-XR)
- configure backup auto-tunnels to avoid SRLGs (always or if possible)

IOS

```
interface X
 mpls traffic-eng srlg 1
 mpls traffic-eng srlg 2
!
interface Y
 mpls traffic-eng srlg 2
 mpls traffic-eng srlg 3
!
mpls traffic-eng auto-tunnel backup
mpls traffic-eng auto-tunnel backup srlg exclude force
```

IOS-XR (4.1.0)

```
srlg
 interface X
  value 1
  value 2
 !
 interface Y
  value 2
  value 3
!
mpls traffic-eng
 interface X
  auto-tunnel backup
   exclude srlg preferred
```

You can have multiple SRLG values per interface.

IOS default is "preferred", while IOS-XR default is "force".

Only backup auto-tunnels can be used.

Links

- IETF - RFC 4203
- IETF - RFC 5307

Inter-Area MPLS TE

Inter-Area MPLS TE Tunnels are not supported by the default configuration.

Two solutions:

- per domain: ERO loose-hop expansion
- distributed: Path Computation Element (PCE)

ERO loose-hop expansion

A loosely routed explicit path must be defined for the tunnel LSP **that identifies each ABR** the LSP should traverse using the "**`next-address loose`**" command. The head-end router and the ABRs along the specified explicit path expand automatically the loose hops, each one computing the strict path segment to the next ABR or tunnel destination.

MPLS TE is configured as usually, with the following change:

- You have to use loose next-hops on the TE Tunnels towards the ABRs

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 4.4.4.4
 tunnel mpls traffic-eng path-option 1 explicit name EXPPATH
!
ip explicit-path name EXPPATH enable
 next-address loose 2.2.2.2
 next-address loose 3.3.3.3
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 4.4.4.4
 path-option 1 explicit name EXPPATH
!
explicit-path name EXPPATH
 index 1 next-address loose ipv4 unicast 2.2.2.2
 index 2 next-address loose ipv4 unicast 3.3.3.3
```

Specifying the tunnel tail-end in a loosely routed path is optional. You need to define only the ABRs as next-address loose and each ABR will automatically find the strict next-address for the next ABR or for the final destination.

You can use "`debug mpls traffic-eng path`" and "`debug mpls traffic-eng tunnels`" debugs on the head-end and the ABRs to troubleshoot any issues.

IOS

```
R1#debug mpls traffic-eng path ?
  <0-65535>  Show debug output for this tunnel only
  api        Path calculation API events
  dump       Path calculation detail info
  errors     Path calculation errors events
  lookup     Path calculation lookup events
  spf        Path calculation SPF events
  verify     Path calculation verify events

R1#debug mpls traffic-eng tunnels ?
  auto-bw          MPLS Traffic Engineering tunnel auto-bw
  errors           MPLS Traffic Engineering tunnel errors
  events           MPLS Traffic Engineering tunnel system events
  fast-reroute     MPLS Traffic Engineering tunnel fast reroute
  labels           MPLS Traffic Engineering tunnel labels
  path-protection  MPLS Traffic Engineering tunnel path protection
  reoptimize       MPLS Traffic Engineering tunnel reoptimization
  signalling       MPLS Traffic Engineering tunnel signalling
  state            MPLS Traffic Engineering tunnel state machine
  timers           MPLS Traffic Engineering tunnel timers
```

As an alternative for an inter-area tunnel crossing different areas, you could configure a sequence of intra-area tunnels, each one crossing one of the areas between source and destination routers, in such a way that the traffic arriving on one tunnel is forwarded into the next tunnel and so on.

OSPF

- An ABR has visibility into the TE topology of all areas where it's attached to.
- An area 0 router has visibility into the TE topology of area 0 only.
- An non-transit area router has visibility into the TE topology of that area only.

Always keep in mind that first you have to enable MPLS TE for all interested areas.

You can use the following abbreviations when filtering the output in order to quickly verify the OSPF area & TE topology per router. Just compare the number of nodes and links on the output vs the ones on the diagram.

IOS

```
R3#sh mpls traffic-eng top | i TE Id
IGP Id: 1.1.1.1, MPLS TE Id:1.1.1.1 Router Node  (ospf 1  area 1)
IGP Id: 2.2.2.2, MPLS TE Id:2.2.2.2 Router Node  (ospf 1  area 1)
IGP Id: 3.3.3.3, MPLS TE Id:3.3.3.3 Router Node  (ospf 1  area 0)
IGP Id: 3.3.3.3, MPLS TE Id:3.3.3.3 Router Node  (ospf 1  area 1)
IGP Id: 4.4.4.4, MPLS TE Id:4.4.4.4 Router Node  (ospf 1  area 0)
IGP Id: 4.4.4.4, MPLS TE Id:4.4.4.4 Router Node  (ospf 1  area 1)
IGP Id: 5.5.5.5, MPLS TE Id:5.5.5.5 Router Node  (ospf 1  area 0)
IGP Id: 6.6.6.6, MPLS TE Id:6.6.6.6 Router Node  (ospf 1  area 0)

R1#sh mpls traffic-eng top | i TE Id|Address
IGP Id: 1.1.1.1, MPLS TE Id:1.1.1.1 Router Node  (ospf 1  area 1)
          frag_id 1, Intf Address:10.1.2.1
IGP Id: 2.2.2.2, MPLS TE Id:2.2.2.2 Router Node  (ospf 1  area 1)
          frag_id 9, Intf Address:20.2.3.2
          frag_id 10, Intf Address:20.2.4.2
          frag_id 2, Intf Address:10.1.2.2
IGP Id: 3.3.3.3, MPLS TE Id:3.3.3.3 Router Node  (ospf 1  area 1)
          frag_id 5, Intf Address:20.2.3.3
IGP Id: 4.4.4.4, MPLS TE Id:4.4.4.4 Router Node  (ospf 1  area 1)
          frag_id 5, Intf Address:20.2.4.4
```

ISIS

In case of ISIS & MPLS TE, the L1/L2 router is considered as the ABR.

- A L1 router has visibility into the TE topology of L1 only
- A L2 router has visibility into the TE topology of L2 only
- A L1/L2 router has visibility into the TE topology of both L1/L2

Always keep in mind that first you have to enable MPLS TE for all interested levels.

You can use the following abbreviations when filtering the output in order to quickly verify the ISIS L1/L2 and TE topology per router. Just compare the number of nodes and links on the output vs the ones on the diagram.

IOS

```
R5#sh mpls traffic-eng top | i TE Id
IGP Id: 0000.0000.0002.00, MPLS TE Id:10.0.0.2 Router Node  (isis  level-
1)
IGP Id: 0000.0000.0005.00, MPLS TE Id:10.0.0.5 Router Node  (isis  level-
1)
IGP Id: 0000.0000.0005.00, MPLS TE Id:10.0.0.5 Router Node  (isis  level-
2)
IGP Id: 0000.0000.0019.00, MPLS TE Id:10.0.0.19 Router Node  (isis
level-2)
IGP Id: 0000.0000.0020.00, MPLS TE Id:10.0.0.20 Router Node  (isis
level-1)
IGP Id: 0000.0000.0020.00, MPLS TE Id:10.0.0.20 Router Node  (isis
level-2)

R5#sh mpls traffic-eng top | i TE Id|Address
IGP Id: 0000.0000.0002.00, MPLS TE Id:10.0.0.2 Router Node  (isis  level-
1)
          frag_id 0, Intf Address:10.0.220.2
          frag_id 0, Intf Address:10.0.25.2
IGP Id: 0000.0000.0005.00, MPLS TE Id:10.0.0.5 Router Node  (isis  level-
1)
          frag_id 0, Intf Address:10.0.25.5
IGP Id: 0000.0000.0005.00, MPLS TE Id:10.0.0.5 Router Node  (isis  level-
2)
          frag_id 0, Intf Address:10.0.195.5
          frag_id 0, Intf Address:10.0.205.5
IGP Id: 0000.0000.0019.00, MPLS TE Id:10.0.0.19 Router Node  (isis
level-2)
          frag_id 0, Intf Address:10.0.195.19
          frag_id 0, Intf Address:10.19.20.19, Nbr Intf
Address:10.19.20.20
IGP Id: 0000.0000.0020.00, MPLS TE Id:10.0.0.20 Router Node  (isis
level-1)
          frag_id 0, Intf Address:10.0.220.20
IGP Id: 0000.0000.0020.00, MPLS TE Id:10.0.0.20 Router Node  (isis
level-2)
          frag_id 0, Intf Address:10.0.205.20
          frag_id 0, Intf Address:10.19.20.20, Nbr Intf
Address:10.19.20.19
```

IOS-XR

```
R2#sh mpls traffic-eng topology | i TE Id
IGP Id: 0000.0000.0002.00, MPLS TE Id:10.0.0.2 Router Node  (isis  level-
1)
IGP Id: 0000.0000.0005.00, MPLS TE Id:10.0.0.5 Router Node  (isis  level-
1)
IGP Id: 0000.0000.0020.00, MPLS TE Id:10.0.0.20 Router Node  (isis
level-1)

R2#sh mpls traffic-eng topology | i TE Id|Address
IGP Id: 0000.0000.0002.00, MPLS TE Id:10.0.0.2 Router Node  (isis  level-
1)
          frag_id 0, Intf Address:10.0.220.2
          frag_id 0, Intf Address:10.0.25.2
IGP Id: 0000.0000.0005.00, MPLS TE Id:10.0.0.5 Router Node  (isis  level-
1)
          frag_id 0, Intf Address:10.0.25.5
IGP Id: 0000.0000.0020.00, MPLS TE Id:10.0.0.20 Router Node  (isis
level-1)
          frag_id 0, Intf Address:10.0.220.20, Nbr Intf
Address:10.0.220.2
```

Links

- IETF - RFC 4105

Inter-AS MPLS TE

Inter-AS MPLS TE Tunnels are not supported by the default configuration. **A loosely routed explicit path** must be defined for the tunnel LSP **that identifies each ASBR** the LSP should traverse using the "**next-address loose**" command. The head-end router and the ASBRs along the specified explicit path expand automatically the loose hops, each one computing the strict path segment to the next ASBR or tunnel destination.

MPLS TE is configured as usually, with the following two changes:

- You have to use loose next-hops on the TE Tunnels
- You have to define some manual TE parameters on the ASBRs for their neighbors

ASBR Forced Link Flooding is the process that allows links to be installed (as point-to-point) into the TE database, although no IGP is running on them. In order to activate that functionality you just need to configure a link as passive for MPLS TE and provide the neighbor's TE-Id and ip address.

- OSPF
    - uses **opaque LSAs (Type 10)** containing a single link TLV
    - only information about the link that has changed is flooded
- IS-IS
    - uses **Type 22 TLVs**
    - information about all links on a node is flooded

There is no need for full communication (i.e. eBGP, static) to exist between the AS for an Inter-AS LSP to become functional, unless a backup LSP is required too.

Head-end

IOS

```
interface Tunnel0
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 9.9.9.9
 tunnel mpls traffic-eng path-option 1 explicit name R4-R5-EXPPATH
!
ip explicit-path name R4-R5-EXPPATH enable
 next-address loose R4
 next-address loose R5
```

IOS-XR

```
interface tunnel-te0
 ipv4 unnumbered Loopback0
 destination 9.9.9.9
 path-option 1 explicit name R4-R5-EXPPATH
!
```

```
explicit-path name R4-R5-EXPPATH
 index 1 next-address loose ipv4 unicast R4
 index 2 next-address loose ipv4 unicast R%
```

ASBR #1

IOS

```
interface FastEthernet0/0
 ip address 10.4.5.4 255.255.255.0
 mpls traffic-eng tunnels
 mpls traffic-eng passive-interface nbr-te-id 5.5.5.5 nbr-if-addr
10.4.5.5
 ip rsvp bandwidth
```

IOS-XR

PCE or verbatim option required

ASBR#2

IOS

```
interface FastEthernet0/0
 ip address 10.4.5.5 255.255.255.0
 mpls traffic-eng tunnels
 mpls traffic-eng passive-interface nbr-te-id 4.4.4.4 nbr-if-addr
10.4.5.4
 ip rsvp bandwidth
```

IOS-XR

PCE or verbatim option required

Don't forget to enable "`mpls traffic-eng tunnels`" and "`ip rsvp bandwidth`" under the inter-as interfaces, like in intra-as MPLS TE scenarios.

You cannot configure IGP enabled interfaces as MPLS-TE passive interfaces.

You can configure multiple passive neighbors under the same interface. If there are multiple neighbors and the AS use different IGPs, then you need to also use the "`nbr-igp-id`" keyword.

The TE topology should include all MPLS TE enabled interfaces and for the inter-as links the nbr-te-id should be shown as "Nbr Intf Address", like in all p2p links.

IOS

```
R1#sh mpls traffic-eng topology | i Address
...
          frag_id 0, Intf Address:10.2.4.4
          frag_id 0, Intf Address:10.3.4.4
          frag_id 0, Intf Address:10.4.5.4, Nbr Intf Address:5.5.5.5
```

The IGP metrics are set to max (shown as "invalid" for ISIS and "MaxLinkMetric" for OSPF), because these routes aren't into RIB, but only into MPLS TE.

IOS sets the TE metric the same as the IGP metric, IOS-XR doesn't.

Configure a TE metric if you want to enable reoptimization for this tunnel.

IOS

```
R9#sh mpls traffic-eng top | i Addr|metric
...
          frag_id 7, Intf Address:10.5.6.6
          TE metric:1, IGP metric:1, attribute flags:0x0
          frag_id 6, Intf Address:10.4.5.5, Nbr Intf Address:4.4.4.4
          TE metric:invalid, IGP metric:invalid, attribute flags:0x0

R1#sh mpls traffic-eng top | i Addr|metric
...
          frag_id 0, Intf Address:10.3.4.4
          TE metric:10, IGP metric:10, attribute flags:0x0
          frag_id 0, Intf Address:10.4.5.4, Nbr Intf Address:5.5.5.5
          TE metric:MaxLinkMetric, IGP metric:MaxLinkMetric, attribute
flags:0x0
```

Use "sh ip ospf database opaque-area" and "sh isis database verbose" to view the IGP details about the MPLS TE passive links.

You can also use the following commands on the ASBRs to verify the inter-as link.

IOS

```
R4#show mpls traffic-eng link-management igp-neighbors
...
Link ID::  Fa0/0.45
    Neighbor ID:  5-5-5-5-0-0-0 (force, non-igp IP: 20.4.5.5)
                  up, Sources: Passive
...
Link ID::  Fa0/0.24
    Neighbor ID:  0000.0000.0004.01 (area: isis  level-2, IP: 0.0.0.0)
                  up, Sources: IGP


R4#show mpls traffic-eng link-management advertisements | i Address
    Link IP Address:      20.3.4.4
    Link IP Address:      20.4.5.4
    Link IP Address:      20.4.6.4
    Link IP Address:      20.2.4.4
```

When viewing the Explicit Route (under the RSVP Path information) of an Inter-AS TE tunnel, loose hops that belong to other AS will be shown with a "*" next to them.

IOS

```
R1#sh mpls traffic-eng tunnels
...
    RSVP Path Info:
      My Address: 10.1.2.1
      Explicit Route: 10.1.2.2 20.2.3.2 20.2.3.3 3.3.3.3
                      6.6.6.6*
```

FRR & backup tunnels

Local protection within a domain operates like intra-domain in the Inter-AS LSPs.

When using FRR on the primary Inter-AS tunnel and an Inter-AS backup tunnel for ASBR link/node protection:

- The explicit path of the backup tunnel must also use loose hops
- The backup tunnel destination must be included in the RRO of the primary tunnel
- You need to somehow (i.e. eBGP, static) **create a route on the MP router** (and all other intra-as routers towards the ASBR) **for the backup tunnel's egress ip address**, so the RESV messages can be sent from the MP (tail-end of the backup tunnel in one domain) to the PLR (head-end of the backup tunnel in the other domain)

When you're having ip connectivity issues, it might be good practice to enable "`debug ip icmp`" and watch out for "host unreachable" messages.

Links

- IETF - RFC 4216
- IETF - RFC 4920
- IETF - RFC 4561

MPLS DiffServ-TE

DiffServ-TE allows bandwidth reservations on a per-class basis.

8 Class Types: CT0 - CT7 (**CT0 and CT1 are currently used**)

These 8 CTs can be combined with the 8 setup/hold priorities creating 64 possible combinations (called TE classes), from where only 8 are actually used (TE0 - TE7). All routers in a network must have a consistent configuration of all the TE classes.

IOS

```
mpls traffic-eng ds-te te-classes
 te-class 0 class-type 0 priority 0
 te-class 1 class-type 1 priority 7
```

IOS-XR

```
mpls traffic-eng
 ds-te te-classes
  te-class 0 class-type 0 priority 0
  te-class 1 class-type 1 priority 7
```

CT0 is usually used for best effort or pre-DE-TE LSPs and is implicitly signaled
CT1 is usually used for special classes like voice and video and is explicitly signaled

Usually CTs map to scheduler queues for the appropriate QoS actions.

CSPF is enhanced to handle per-CT reservation requirements
IGP is enhanced to carry per-CT available bandwidth at different priorities
RSVP is enhanced to carry a new Class Type Object (CTO) in the PATH message

Bandwidth Constraint Models

BC (Bandwidth Constraint) = the percentage of a link's bandwidth that a CT can use

BC0 = global pool (for best-effort & DiffServ traffic)
BC1 = sub-pool (for strict bandwidth guarantee traffic)

The BC model determines the available bandwidth for each CT at each priority.

Two BC models:

- MAM (Maximum Allocation Model)
  - RFC 4125
  - the link bandwidth is divided among the different CTs
  - it completely isolates the different CTs

    - different priorities on different CTs do not have any effect
    - bandwidth is wasted due to not being able to share unused bandwidth between CTs
- RDM (Russian Dolls Model)
    - RFC 4127
    - CTs can share bandwidth on a link
    - no isolation between CTs
    - preemption (using priorities) is required to guarantee bandwidth to a CT
    - bandwidth is utilized more efficiently between CTs

RDM is the default BC model, after enabling IETF DS-TE.

IOS

```
mpls traffic-eng ds-te mode ietf
```

IOS-XR

```
mpls traffic-eng
 ds-te mode ietf
```

The same mode should be used in all MPLS-TE routers.

IOS

```
R3#sh mpls traffic-eng top | i BC
BC Model Id: RDM
          BC Model Id: RDM
          BC0 (max_reservable): 75000 (kbps)
          BC0 (max_reservable_bw_global): 0 (kbps)
          BC1 (max_reservable_bw_sub): 0 (kbps)
```

IOS-XR

```
GSR#sh mpls traffic top | i BC
My_BC_Model_Type: RDM
      BC Model ID:RDM
      BC0:50000 (kbps) BC1:0 (kbps)
      BC Model ID:RDM
      BC0:50000 (kbps) BC1:0 (kbps)
      BC Model ID:RDM
      BC0:50000 (kbps) BC1:0 (kbps)
```

Configuration Steps

- sub-pool (BC1) tunnel
    - choose a specific PHB for classifying the strict guarantee traffic
    - limit the traffic before it enters the tunnel in order to avoid oversubscription
    - mark the traffic with the right EXP while entering the tunnel
    - map the EXP into the appropriate queue at each outgoing interface of every tunnel hop
    - reserve the right percentage of the total sub-pool bandwidth on each outgoing link
- global pool (BC0) tunnel

  - choose a specific PHB for each class
  - mark each class with the right EXP while entering the tunnel
  - map each EXP into the appropriate queue at each outgoing interface of every tunnel hop
  - reserve the right percentage of the total global pool bandwidth on each outgoing link

IOS

```
interface Tunnel1
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 6.6.6.6
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 444
 tunnel mpls traffic-eng path-option 1 explicit name R4-R5
!
interface FastEthernet0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth 99999 sub-pool 9999
```

After switching to IETF DS-TE the above configuration becomes:

IOS

```
interface Tunnel1
 ip unnumbered Loopback0
 tunnel mode mpls traffic-eng
 tunnel destination 6.6.6.6
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth 444 class-type 1
 tunnel mpls traffic-eng path-option 1 explicit name R4-R5
!
interface FastEthernet0/0
 mpls traffic-eng tunnels
 ip rsvp bandwidth rdm bc0 99999 bc1 9999
```

IOS-XR supports either sub-pool or class-type configuration by default.

Links

- IETF - RFC 3564
- IETF - RFC 4124
- IETF - RFC 3270

P2MP LSPs

- One ingress router
- Many egress routers
- Replication (one=>many) happens at branch routers (ingress router can be a branch router too)
- The path of the P2MP LSP is determined by the branch routers
- All sub-LSPs of the same P2MP LSP employ the same constraints/policies
- Traffic is unidirectional, like in P2P LSPs

Two options in control plane:

- LDP
    - no TE support
    - LSP is initiated by the egress routers
    - Egress routers advertise each one a label for the P2MP FEC towards the branch router
    - Branch routers advertise a single label for the P2MP FEC (for multiple received labels from egress routers) toward the ingress router
    - Ingress routers need to keep state only for the directly connected routers
- RSVP
    - TE support
    - LSP is initiated by the ingress router
    - Sub-LSPs (like normal LSPs) are created from the ingress router towards each egress router
    - Each sub-LSP has its own PATH/RESV messages, which contain the P2MP Session Object (so that all intermediate routers know the parent LSP where these sub-LSPs belong to)
    - Ingress router sends a PATH message (hop-by-hop) to each egress router (towards the branch router)
    - Branch routers send multiple PATH messages (one for each sub-LSP) towards the multiple egress routers
    - Egress routers receive the PATH messages and each one sends a RESV message towards the branch router
    - Branch routers receive multiple RESV messages (one for each sub-LSP) from each egress router (of a different branch), each one with a different label
    - Branch routers send multiple RESV messages (one for each sub-LSP) towards the ingress router, each one with the same label (since they belong to the same parent P2MP LSP).
    - Ingress router receives multiple RESV messages with the same label from each branch router
    - Ingress routers need to keep state for all egress routers (due to sub-LSPs)

P2MP TE

Limitations

- Inter-area and inter-as topologies are not supported, only intra-as
- Node and path protection are not supported, only link protection
- PIM Sparse is not supported, only SSM
- Many OAM features are not supported
- Multiple path-options per destination are not supported, one one path-option per destination
- PHP is not supported on the tail-end, explicit-null or a normal label must be used
- Only one backup auto-tunnel is created, to be used for link-protection

Multicast

- Head-end router and tail-end routers exchange PIM messages with the locally attached CEs
- MPLS core doesn't need to run PIM
- PIM doesn't need to be enabled across the MPLS TE tunnel

You can use dynamic paths or explicit paths. If you configure expicit paths that lead to multiple sub-LSPs entering a middle router through different interfaces but exiting through a common interface, then the sub-LSP won't be established (remerge event).

The following apply to all sub-LSPs of the same P2MP LSP:

- andwidth parameters
- hold/setup priorities
- administrative weight
- attributes/affinity

Because a sub-LSP cannot be represented as a regular interface on the tail-end router, a special LSP virtual interface (**LSPVIF**) is automatically created. The LSPVIF represents the originating interface for all IP multicast traffic arriving to the P2MP TE tail-end router. This LSPVIF interface is used for the RPF check.

MPLS-TE should be configured and working between all routers being involved in the P2MP LSPs.

head-end (IOS)

```
ip multicast-routing
!
ip pim ssm default
!
interface Tunnel0
 ip unnumbered Loopback0
 ip pim passive
 ip igmp static-group 232.20.20.20 source 7.7.7.7
 ip igmp static-group 232.19.19.19 source 7.7.7.7
 ip igmp static-group 232.8.8.8 source 7.7.7.7
 ip igmp static-group 232.2.2.2 source 7.7.7.7
 tunnel mode mpls traffic-eng point-to-multipoint
 tunnel destination list mpls traffic-eng name TAIL-END-ACL
!
mpls traffic-eng destination list name TAIL-END-ACL
 ip 2.2.2.2 path-option 1 dynamic
 ip 19.19.19.19 path-option 1 dynamic
 ip 20.20.20.20 path-option 1 dynamic
```

For every multicast group you'll need a static igmp under the TE tunnel.

Path-option per sub-LSP can be either dynamic or explicit.

mid-point (IOS)

just normal MPLE-TE config, no multicast required

support of signaling extensions is required

tail-end (IOS)

```
ip multicast-routing
ip multicast mpls traffic-eng
!
ip pim ssm default
!
ip mroute 7.7.7.7 255.255.255.255 1.1.1.1
```

In order to avoid failing on the RPF check on the tail-end router, you must configure a static mroute for the multicast source pointing to the head-end router.

In IOS-XR, the interface "**tunnel-mte**" is used for P2MP LSPs.

head-end (IOS-XR)

```
multicast-routing
 address-family ipv4
  interface tunnel-mte0
   enable
!
router igmp
 interface tunnel-mte0
  static-group 232.2.2.2 7.7.7.7
  static-group 232.8.8.8 7.7.7.7
  static-group 232.19.19.19 7.7.7.7
  static-group 232.20.20.20 7.7.7.7
!
interface tunnel-mte0
 ipv4 unnumbered Loopback0
 destination 1.1.1.1
  path-option 1 dynamic
 !
 destination 2.2.2.2
  path-option 1 dynamic
 !
 destination 19.19.19.19
  path-option 1 dynamic
```

mid-point (IOS-XR)

just normal MPLE-TE config, no multicast required

support of signaling extensions is required tail-end (IOS-XR)

```
multicast-routing
 address-family ipv4
  core-tree-protocol rsvp-te
  static-rpf 7.7.7.7 32 mpls 1.1.1.1
```

IOS-XR 4.2 is required for P2MP LSPs. IOS-XR on C12k doesn't support P2MP LSPs.

IOS-XR doesn't support pings on loopbacks belonging to VRFs under a mVPN topology. Use a physical interface instead.

"multicast-intact" might be required if P2MP and P2P LSPs are created between two nodes.

For P2MP TE FRR protection, the "ip routing protocol purge interface" command is recommended on every penultimate hop router. Otherwise, the traffic can be lost for a few seconds during a FRR cutover event.

It is assumed that all routers outside the MPLS-TE core are using multicast as usual. So the head-end is having a normal PIM connectivity to the source and the tail-ends are having normal PIM connectivity to the receivers.

Verification

Keep all possible loggings enabled in order to aid in troubleshooting.

IOS

```
mpls traffic-eng logging lsp path-errors
mpls traffic-eng logging lsp reservation-errors
mpls traffic-eng logging lsp preemption
mpls traffic-eng logging lsp setups
mpls traffic-eng logging lsp teardowns
mpls traffic-eng logging tunnel path change
```

When the tail-end router creates the LSPVIF interface, you'll get the following log:

```
%MPLS_TE-5-LSP: Sub-LSP 1.1.1.1[1:3]->2.2.2.2_0: UP
%LINEPROTO-5-UPDOWN: Line protocol on Interface Lspvif0, changed state to
up
```

head-end (IOS)

```
R1#sh mpls traffic-eng tunnels

P2P TUNNELS/LSPs:

P2MP TUNNELS:
```

```
Tunnel0     (p2mp),  Admin: up, Oper: up
  Name: R1_t0

  Tunnel0 Destinations Information:

    Destination     State SLSP UpTime
    2.2.2.2         Up    01:47:12
    19.19.19.19     Up    01:47:28
    20.20.20.20     Up    01:48:35

    Summary: Destinations: 3 [Up: 3, Proceeding: 0, Down: 0 ]
      [destination list name: TAIL-TE-ACL]

  History:
    Tunnel:
      Time since created: 1 hours, 51 minutes
      Time since path change: 1 hours, 48 minutes
      Number of LSP IDs (Tun_Instances) used: 1
    Current LSP: [ID: 1]
      Uptime: 1 hours, 48 minutes

  Tunnel0 LSP Information:
    Configured LSP Parameters:
      Bandwidth: 0        kbps (Global)  Priority: 7  7   Affinity:
0x0/0xFFFF
      Metric Type: TE (default)

  Session Information
    Source: 1.1.1.1, TunID: 0

    LSPs
      ID: 1 (Current), Path-Set ID: 0xA2000001
        Sub-LSPs: 3, Up: 3, Proceeding: 0, Down: 0

      Total LSPs: 1

P2MP SUB-LSPS:

 LSP: Source: 1.1.1.1, TunID: 0, LSPID: 1
     P2MP ID: 0, Subgroup Originator: 1.1.1.1
     Name: R1_t0
     Bandwidth: 0, Global Pool

  Sub-LSP to 20.20.20.20, P2MP Subgroup ID: 1, Role: head
    Path-Set ID: 0xA2000001
    OutLabel : FastEthernet0/0.13, 29
    Next Hop : 12.1.3.3
    Explicit Route: 12.1.3.3 12.3.4.4 12.4.20.20 20.20.20.20
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE
  Sub-LSP to 19.19.19.19, P2MP Subgroup ID: 2, Role: head
    Path-Set ID: 0xA2000001
    OutLabel : FastEthernet0/0.13, 29
    Next Hop : 12.1.3.3
    Explicit Route: 12.1.3.3 12.3.5.5 12.5.6.6 12.6.19.19
                    19.19.19.19
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE

  Sub-LSP to 2.2.2.2, P2MP Subgroup ID: 3, Role: head
    Path-Set ID: 0xA2000001
    OutLabel : FastEthernet0/0.13, 29
    Next Hop : 12.1.3.3
    Explicit Route: 12.1.3.3 12.3.5.5 12.5.6.6 12.2.6.2
                    2.2.2.2
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE
```

head-end (IOS)

```
R1#show mpls traffic-eng tunnels brief
Signalling Summary:
    LSP Tunnels Process:            running
    Passive LSP Listener:           running
    RSVP Process:                   running
    Forwarding:                     enabled
    Periodic reoptimization:        every 3600 seconds, next in 3463
seconds
    Periodic FRR Promotion:         Not Running
    Periodic auto-bw collection:    every 300 seconds, next in 163
seconds

P2P TUNNELS/LSPs:
Displayed 0 (of 0) heads, 0 (of 0) midpoints, 0 (of 0) tails

P2MP TUNNELS:
                         DEST        CURRENT
INTERFACE    STATE/PROT  UP/CFG     TUNID  LSPID
Tunnel0      up/up       3/3        0      1
Displayed 1 (of 1) P2MP heads

P2MP SUB-LSPS:
SOURCE        TUNID  LSPID  DESTINATION    SUBID  STATE UP IF      DOWN IF
1.1.1.1       0      1      20.20.20.20    1      Up    head
Fa0/0.13
1.1.1.1       0      1      19.19.19.19    2      Up    head
Fa0/0.13
1.1.1.1       0      1      2.2.2.2        3      Up    head
Fa0/0.13
```

```
Displayed 3 P2MP sub-LSPs:
          3 (of 3) heads, 0 (of 0) midpoints, 0 (of 0) tails
```

head-end (IOS)

```
R1#show mpls traffic-eng forwarding path-set brief
Sub-LSP Identifier
src_lspid[subid]->dst_tunid              InLabel Next Hop      I/F
PSID
---------------------------              ------- ------------- ------ ---
-
1.1.1.1_1[1]->20.20.20.20_0              none    12.1.3.3      Fa0/0.
A2000001
1.1.1.1_1[2]->19.19.19.19_0              none    12.1.3.3      Fa0/0.
A2000001
1.1.1.1_1[3]->2.2.2.2_0                  none    12.1.3.3      Fa0/0.
A2000001
```

head-end (IOS)

```
R1#show mpls traffic-eng forwarding path-set detail

  LSP: Source: 1.1.1.1, TunID: 0, LSPID: 1
  Destination: 20.20.20.20, P2MP Subgroup ID: 1
    Path Set ID: 0xA200000
    OutLabel : FastEthernet0/0.13, 29
    Next Hop : 12.1.3.3

  LSP: Source: 1.1.1.1, TunID: 0, LSPID: 1
  Destination: 19.19.19.19, P2MP Subgroup ID: 2
    Path Set ID: 0xA200000
    OutLabel : FastEthernet0/0.13, 29
    Next Hop : 12.1.3.3

  LSP: Source: 1.1.1.1, TunID: 0, LSPID: 1
  Destination: 2.2.2.2, P2MP Subgroup ID: 3
    Path Set ID: 0xA200000
    OutLabel : FastEthernet0/0.13, 29
    Next Hop : 12.1.3.3
```

The same commands can also be used on all the intermediate routers.

head-end (IOS)

```
R1#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E -
Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(7.7.7.7, 232.20.20.20), 01:03:22/stopped, flags: sTI
  Incoming interface: Serial2/0.100, RPF nbr 192.168.78.7
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 01:03:22/00:02:59

(7.7.7.7, 232.2.2.2), 01:17:17/stopped, flags: sTI
  Incoming interface: Serial2/0.100, RPF nbr 192.168.78.7
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 01:17:17/00:02:59

(7.7.7.7, 232.19.19.19), 01:17:11/stopped, flags: sTI
  Incoming interface: Serial2/0.100, RPF nbr 192.168.78.7
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 01:17:11/00:02:59

(7.7.7.7, 232.8.8.8), 01:17:27/stopped, flags: sTI
  Incoming interface: Serial2/0.100, RPF nbr 192.168.78.7
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 01:17:27/00:02:59
```

mid-point (IOS)

```
R3#sh mpls forwarding-table | i 1.1.1.1|Prefix
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
23         Pop Label  1.1.1.1/32       41241         Fa0/0.13   12.1.3.1
29         29         1.1.1.1 0 [1]    2684          Fa0/0.34   12.3.4.4
           16         1.1.1.1 0 [1]    2684          Fa0/0.35   12.3.5.5
```

On every mid-point router you should see more than one label for the same TE LSP, each one pointing to a different outgoing interface.

mid-point (IOS)

```
R3#sh mpls traffic-eng tunnels

P2P TUNNELS/LSPs:

P2MP TUNNELS:
```

```
P2MP SUB-LSPS:

 LSP: Source: 1.1.1.1, TunID: 0, LSPID: 1
     P2MP ID: 0, Subgroup Originator: 1.1.1.1
     Name: R1_t0
     Bandwidth: 0, Global Pool

  Sub-LSP to 20.20.20.20, P2MP Subgroup ID: 1, Role: midpoint
    Path-Set ID: 0x84000001
    InLabel  : FastEthernet0/0.13, 29
    Prev Hop : 12.1.3.1
    OutLabel : FastEthernet0/0.34, 29
    Next Hop : 12.3.4.4
    Explicit Route: 12.3.4.4 12.4.20.20 20.20.20.20
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE

  Sub-LSP to 19.19.19.19, P2MP Subgroup ID: 2, Role: midpoint
    Path-Set ID: 0x84000001
    InLabel  : FastEthernet0/0.13, 29
    Prev Hop : 12.1.3.1
    OutLabel : FastEthernet0/0.35, 16
    Next Hop : 12.3.5.5
    Explicit Route: 12.3.5.5 12.5.6.6 12.6.19.19 19.19.19.19
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE

  Sub-LSP to 2.2.2.2, P2MP Subgroup ID: 3, Role: midpoint
    Path-Set ID: 0x84000001
    InLabel  : FastEthernet0/0.13, 29
    Prev Hop : 12.1.3.1
    OutLabel : FastEthernet0/0.35, 16
    Next Hop : 12.3.5.5
    Explicit Route: 12.3.5.5 12.5.6.6 12.2.6.2 2.2.2.2
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE
```

Sub-LSPs following the same branch (egress interface) will use the same label.

tail-end (IOS)

```
R2#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E -
Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(7.7.7.7, 232.2.2.2), 00:28:39/stopped, flags: sLTI
  Incoming interface: Lspvif0, RPF nbr 1.1.1.1, Mroute
  Outgoing interface list:
    FastEthernet1/0, Forward/Sparse-Dense, 00:28:39/00:01:20
```

tail-end (IOS)

```
R2#sh mpls traffic-eng tunnels

P2P TUNNELS/LSPs:

P2MP TUNNELS:

P2MP SUB-LSPS:

 LSP: Source: 1.1.1.1, TunID: 0, LSPID: 1
     P2MP ID: 0, Subgroup Originator: 1.1.1.1
     Name: R1_t0
     Bandwidth: 0, Global Pool

  Sub-LSP to 2.2.2.2, P2MP Subgroup ID: 3, Role: tail
    Path-Set ID: 0x40000001
    InLabel  : FastEthernet0/0.26, 33
    Prev Hop : 12.2.6.6
    OutLabel :  -
    Explicit Route:  NONE
    Record   Route (Path):  NONE
    Record   Route (Resv):  NONE
```

Links

- IETF - RFC 4461
- IETF - RFC 4875
- IETF - RFC 6388

Multicast

PIM-DM (Protocol Independent Multicast- Dense Mode) is defined in RFC 3973.
PIM-SM (PIM - Sparse Mode) is defined in RFC 4601.
PIM-SSM (PIM - Source Specific Multicast) is defined in RFC 4607.
BSR (BootStrap Router) for PIM is defined in RFC 5059.

Multicast Ranges

Multicast range:

- 224.0.0.0/8
- FF00::/8

Important ranges (IANA):

- Link-local range (TTL=1)
  - 224.0.0.0/24
  - FFx2::/16
- SSM range:
  - 232.0.0.0/8
  - FF3x::/32 (FF3x::8000:0000 - FF3x::FFFF:FFFF)
- GLOP range: 233.0.0.0/8
- Admin-scope range: 239.0.0.0/8

Link-local addresses are not constrained by IGMP snooping. The same also applies to x.0.0.x or x.128.0.x (due to 32:1 mcast=>eth mapping).

GLOP range

GLOP addressing is defined in RFC 3180.

Every 16bit ASN has its own GLOP 233.X.Y.0/24 range.

i.e. for AS 12345, one hex=>dec and two dec=>hex conversions need to be made.

```
12345 => 0x3039
0x30 => 48 (X)
0x39 => 57 (Y)

AS 12345 = > 233.48.57.0/24
```

Organizations with a 32-bit ASN may apply for space in AD-HOC Block III (233.252.0.0 - 233.255.255.255) also known as Extended GLOP (EGLOP), or consider using IPv6 multicast addresses.

Multicast Routing Protocols

Main functionalities:

- setup multicast forwarding state
- exchange information about the multicast forwarding state

Protocols

- DVMRP
- PIM-DM
- PIM-SM

Expect to see only PIM-SM being used in most networks.

IOS

```
ip multicast-routing
```

IOS-XR

```
multicast-routing
```

Enabling multicast-routing on an interface in IOS-XR will enable PIM and IGMP automatically. There is no need to configure the "router pim" command (unless something extra is required, like MDT or RP), since the PIM mode it's automatically determined by the group range.

IOS-XR

```
GSR#sh pim group-map
Fri Jun  9 06:11:08.463 UTC

IP PIM Group Mapping Table
(* indicates group mappings being used)
(+ indicates BSR group mappings active in MRIB)

Group Range          Proto Client   Groups RP address      Info

224.0.1.39/32*       DM    perm     0      0.0.0.0
224.0.1.40/32*       DM    perm     1      0.0.0.0
224.0.0.0/24*        NO    perm     0      0.0.0.0
232.0.0.0/8*         SSM   config   1      0.0.0.0
224.0.0.0/4*         SM    static   0      0.0.0.0         RPF: 
Null,0.0.0.0
```

Each multicast group can be one of sparse, dense, bidir, ssm. Each interface can be many modes, each one depending on the egress multicast group.

In IOS-XR, the router itself is also listed as a pim neighbor with a "*".

PIM-SM RP

For PIM-SM to work properly, all routers in a domain must know and agree on the active RP for each multicast group (Group-to-RP mappings).

Group-to-RP mappings can be created using:

- Static group-to-RP mapping
- Auto-RP
  - uses dense-mode (configure "`sparse-dense-mode`" or "`sparse-mode`" & "`ip pim auto-rp listener`")
  - uses 224.0.1.39 for RP announcements (from RP to MA)
  - uses 224.0.1.40 for MA announcements (from MA to all PIM routers)
  - the priority of each RP cannot be defined (the RP with the higher ip address wins)
  - the interval/scope of Auto-RP announcements can be defined
  - use "`ip multicast boundary ACL in/out filter-autorp`" to filter Auto-RP announcements entering/leaving your network
- PIM BSR (bootstrap router)
  - uses sparse-mode (configure "`no ip pim dm-fallback`" in dense environments)
  - uses 224.0.0.13 for BSR announcements (from BSR to all PIM routers)
  - unicast for RP announcements (from RP to BSR router)
  - the priority of each RP can be defined
  - the interval/scope of BSR announcements cannot be defined
  - use "`ip pim bsr-border`" to filter BSR announcements from entering/leaving your network

On hub-n-spoke networks, when auto-rp announcements must pass between the spokes, you cannot use nbma-mode, because this works only in sparse mode (and announcements are in dense). You have to use BSR or create a pim-enabled tunnel between the spokes.

Static RP and BSR are the most common ways to configure a Group-to-RP mapping.

Static RP

IOS

```
interface Loopback0
 ip pim sparse-mode
!
ip pim rp-address 1.1.1.1
```

IOS-XR

```
router pim
 address-family ipv4
  rp-address 1.1.1.1
  interface Loopback0
   enable
```

Auto-RP

IOS

```
interface Loopback0
 ip pim sparse-mode
!
ip pim send-rp-announce Loopback0 scope 10
ip pim send-rp-discovery Loopback0 scope 10
```

IOS-XR

```
router pim
 address-family ipv4
  auto-rp mapping-agent Loopback0 scope 10
  auto-rp candidate-rp Loopback0 scope 10
  interface Loopback0
   enable
```

In IOS-XR, Auto-RP is not supported for VRFs.

In IOS, you can use an ACL to filter the auto-rp groups.

Auto-RP requires either sparse-dense mode or sparse mode and auto-rp listener (by default enabled in IOS-XR).

BSR

IOS

```
interface Loopback0
 ip pim sparse-mode
!
ip pim bsr-candidate Loopback0
ip pim rp-candidate Loopback0
```

IOS-XR

```
router pim
 address-family ipv4
  interface Loopback0
   enable
  !
  bsr candidate-bsr 1.1.1.1
  bsr candidate-rp 1.1.1.1
```

When static RP is configured together with Auto-RP or BSR for the same PR mappings, Auto-RP/BSR created mappings take precedence, unless "`override`" is configured with the static rp-address.

PIM-SSM

PIM-SSM uses PIM-SM + IGMPv3/MLDv2.

Summary

- Receivers report interest for a particular source using IGMv3
- PIM routers do RPF lookups for the source and join upstream towards the source
- An SPT is created between the source and the receivers
- Multicast traffic starts to flow from source to the receivers on the SPT

Detailed analysis

1. Receiver
    1. sends an IGMPv3 Report (S,G) to the LAN
2. Receiver DR
    1. receives the IGMP Report (S,G) from the Receiver
    2. adds the incoming IGMP interface to OIL for (S,G)
    3. performs RPF lookup for source S
    4. sends a PIM Join (S,G) to the upstream router out of the RPF interface
3. Upstream Routers (from Receiver DR towards the Source)
    1. add the incoming PIM interface to OIL for (S,G)
    2. perform RPF lookup for source S
    3. send a PIM Join (S,G) to the next upstream router out of the RPF interface (*)
    4. ...

(*) This propagation of PIM Join messages our of the RPF interface continues until the source DR is reached or an upstream router has already multicast forwarding state for this group.

IOS

```
ip pim ssm default
!
interface X
 ip pim sparse-mode
```

IOS-XR

```
multicast-routing
 address-family ipv4
  interface X
   enable
```

SSM is enabled by default on IOS-XR for 232.0.0.0/8.

In IOS-XR, interfaces enabled under multicast-routing run PIM sparse-mode by default.

PIM-SM

Summary

- A receiver asks for multicast data and an RPT (*,G) is built from the receiver DR to the RP
- When the source transmits the multicast data, first packets are sent encapsulated into PIM Register messages by the source DR to the RP and then an SPT (S,G) is built from the RP to the source
- When the multicast data reaches the receiver, there is a new SPT (S,G) built from the receiver DR directly to the source
- Receiver DR sends a prune message to the RP to stop the initial RPT

Detailed analysis

Phase #1 (Receiver asks for multicast data)

1. Receiver
    1. sends an (*,G) IGMP Report to the LAN
2. Receiver DR Router
    1. receives the (*,G) IGMP Report from the Receiver
    2. adds the incoming IGMP interface to OIL for (*,G)
    3. performs RPF lookup for RP
    4. sends a (*,G) PIM Join to the upstream router out of the RPF interface
3. Upstream Routers (from Receiver DR towards the RP)
    1. receive the (*,G) PIM Join from the downstream router
    2. add the incoming PIM interface to OIL for (*,G)
    3. perform RPF lookup for RP
    4. send a (*,G) PIM Join to the next upstream router out of the RPF interface (*)

Phase #2 (Source sends multicast data and Receiver receives it through the RP)

1. Source S
    1. starts sending native multicast data (S,G) to the LAN
2. Source DR Router
    1. receives native multicast data from the source
    2. adds the incoming multicast data interface to IIF
    3. encapsulates multicast data in PIM Register messages
    4. sends the PIM Register messages to the RP as unicast packets
3. RP (with active RPT)
    1. receives and decapsulates the PIM Register messages
    2. sends the decapsulated native multicast data to the receiver down the RPT
4. Receiver
    1. receives the native multicast data through the RP
5. RP (with active RPT)
    1. performs RPF lookup for source S
    2. sends a (S,G) PIM Join to the upstream router (towards source) out of the RPF interface
6. Upstream Routers (from RP towards the Source)
    1. receive the (S,G) PIM Join
    2. add the incoming PIM interface to OIL for (S,G)
    3. perform RPF lookup for source S
    4. send a (S,G) PIM Join to the next upstream router (towards source) out of the RPF interface (*)
7. Source DR Router

   1. receives the (S,G) PIM Join
   2. starts sending native multicast data to the RP
8. RP (with active RPT)
   1. receives duplicate multicast data (encapsulated from Source DR and native from source/upstream)
   2. sends a PIM Register-Stop message to the Source DR
9. Source DR Router
   1. receives the PIM Register-Stop message from the RP
   2. stops sending PIM Register messages with encapsulated multicast data to the RP
10. RP (with active RPT)
    1. receives only native multicast data
    2. sends the native multicast data to the receivers down the RPT
11. Receiver
    1. receives the native multicast data through the RP

Phase #3 (Receiver receives the multicast data directly from the Source)

1. Receiver DR Router
   1. receives multicast data (S,G) from source S through the RP
   2. performs RPF lookup for source S
   3. sends a (S,G) PIM Join to the upstream router out of the RPF interface
2. Upstream Routers (from Receiver DR towards the Source DR)
   1. receive the (S,G) PIM Join from the downstream router
   2. add the incoming PIM interface to OIL for (S,G)
   3. perform RPF lookup for source S
   4. send a (S,G) PIM Join to the next upstream router out of the RPF interface (*)
3. Source DR Router
   1. receives the (S,G) PIM Join from the downstream router
   2. starts sending native multicast data to the Receiver DR Router
4. Receiver DR Router
   1. receives duplicate multicast data (from source DR through SPT and from RP through RPT)
   2. sends a PIM Prune (S,G,RPT) to the upstream router out of the RPF interface
5. Upstream Routers (from Receiver DR towards the RP)
   1. receive the PIM Prune (S,G,RPT)
   2. remove the incoming PIM interface from OIL for (*,G)
   3. if OIL is null, send a PIM Prune (S,G) to the upstream router out of the RPF interface
6. RP
   1. receives the PIM Prune (S,G)
   2. removes the incoming PIM interface from OIL for (*,G)
   3. if OIL is null, sends a PIM Prune (S,G) to the upstream router out of the RPF interface
7. Source DR
   1. receives the PIM Prune (S,G)
   2. removes the incoming PIM interface from OIL for (S,G)

(*) This propagation of PIM Join messages our of the RPF interface continues until the RP or the Source DR is reached or an upstream router has already multicast forwarding state for this group.

PIM Prune messages might not be sent, if there is another active multicast state for the same group in the router.

The Source DR periodically sends a PIM Null-Register message (without any multicast data inside) to the RP, in order to inform it that the source is still active.

IOS

```
interface X
 ip pim sparse-mode
```

IOS-XR

```
multicast-routing
 address-family ipv4
  interface X
   enable
```

OIL = Outgoing Interface List
IIF = Incoming Interface

In the shortest path tree (SPT), the root of the tree is the source.
In the shared path tree (RPT), the root of the tree is the RP.

PIM Joins are sent to 224.0.0.13.

The threshold for switching from the RPT to SPT is 0 by default, which means once the Receiver DR receives the first multicast packet and learns the source, it sends a (S,G) PIM Join towards the source. When is starts receiving multicast data from the SPT, it sends a PIM Prune for traffic received via the RPT towards the RP.

PIM Bidir

PIM Bidir is significantly simpler in operation than PIM-SM.

It eliminates the maintenance of (S,G) entries (only (*,G) are kept) and there's no data driven events hence packets never need to be signaled and preserved (much more scalable for many-to-many applications).

Bidir PIM uses the Designated Forwarder (DF) election mechanism (based on routing protocol cost to the RP) to elect a single router on each link, which becomes responsible for the following tasks:

- picking up packets from the link and forwarding them upstream towards the RP
- forwarding downstream multicast packets from the RP onto the link

When configuring both sparse and bidir groups, you need to explicitly define them, because everything excluded from bidir is dense by default.

IOS

```
ip pim bidir-enable
ip pim rp-address 1.1.1.1 bidir
```

IOS-XR

```
router pim
 address-family ipv4
  rp-address 1.1.1.1 bidir
```

In some IOS-XR releases you might need to define the mcast bidir range of group addresses.

IOS

```
R2#sh ip pim rp map
PIM Group-to-RP Mappings

Group(s): 224.0.0.0/4, Static, Bidir Mode
    RP: 1.1.1.1 (?)
```

IOS-XR

```
GSR#sh pim group-map
Tue Feb  4 13:57:55.368 UTC

IP PIM Group Mapping Table
(* indicates group mappings being used)
(+ indicates BSR group mappings active in MRIB)

Group Range         Proto Client    Groups RP address      Info

224.0.1.39/32*      DM    perm      0      0.0.0.0
224.0.1.40/32*      DM    perm      1      0.0.0.0
224.0.0.0/24*       NO    perm      0      0.0.0.0
232.0.0.0/8*        SSM   config    0      0.0.0.0
224.0.0.0/4*        BD    config    1      1.1.1.1         RPF:
Gi0/2/1/2.28,2.2.28.2
224.0.0.0/4         SM    static    0      0.0.0.0         RPF:
Null,0.0.0.0
```

You will see only (*,G) entries for the bidir groups, no (S,G) entries.

PIM Register Tunnels

In latest software releases, PIM uses tunnel interfaces for RP Register communication. You can use "sh ip pim tunnel" in order to verify the PIM connectivity, either inside or outside a vrf. If no tunnel exists but should exist, then there is probably something wrong.

IOS

```
R1#sh ip pim tunnel
Tunnel0
  Type  : PIM Encap
  RP    : 10.0.0.1*
  Source: 10.0.0.1
Tunnel1*
  Type  : PIM Decap
  RP    : 10.0.0.1*
  Source: -

R1#sh ip pim vrf VPN tunnel
Tunnel1
  Type  : PIM Encap
  RP    : 10.0.0.1
  Source: 10.1.7.1

R1#sh int tun1 | i protocol/transport
  Tunnel protocol/transport PIM/IPv4
```

IOS-XR

```
GSR#sh pim tunnel info all
Fri Jun  9 06:04:51.319 UTC

Interface           RP Address      Source Address

Encapstunnel0       10.0.0.1        10.0.0.1
Decapstunnel0       0.0.0.0          -
```

The router acting as RP should have at least two PIM tunnels: one for Encapsulation (usually src=RP) and another one for Decapsulation. All other PIM routers should have only one for Encapsulation (src=local PIM address, dst=RP).

IGMP

You can use the following as multicast receiver.

IOS

```
interface X
 ip igmp join-group 224.1.1.1
```

IOS-XR

```
router igmp
 interface X
  join-group 232.1.1.1 192.168.1.1
```

PIM (sparse) is also required on the interface where IGMP is enabled.

"`ip igmp static-group x.x.x.x`" can also be used, if we want to avoid having the multicast traffic being processed by the router.

IGMP (v3) for SSM requires to also define the source of multicast traffic.

In IOS, when using **multicast ping to test connectivity**, the multicast packets go out through all multicast enabled interfaces by default (regardless of the source ip/interface chosen in CLI). If you want to send it out through a specific interface then you need to use **extended ping** and choose a specific interface in the "Interface" option.

IOS

```
R1#ping
Protocol [ip]:
Target IP address: 232.1.1.1
Repeat count [1]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Interface [All]: FastEthernet0/0
Time to live [255]:
Source address: 1.1.1.1
```

Advanced Multicast

RPF (Reverse Path Forwarding)

In an RPF check, **the router looks in a routing table to determine its RPF interface, which is the interface closest to the root (the source or the RP)**. The RPF interface is also the incoming interface for the multicast data. RPF checks happen in the control-plane (PIM, MSDP) and in the data-plane (multicast data).

The routing table used for RPF checks can be the global unicast routing table or a separate multicast routing table. In any case, the RPF table contains only unicast routes (the multicast source or the RP).

MP-BGP and M-ISIS can be used to create separate unicast and multicast routing tables.

MP-BGP updates can include IPv4 multicast RPF routes along with IPv4 unicast routes, totally separated (different path attributes) from each other.

You can use "`sh ip mroute count`" to check for increasing RPF counts.

Fixing RPF issues

If you have IP connectivity over a TE tunnel and require PIM connectivity too, then extra care must be taked. Since you cannot activate PIM on a TE tunnel, you must somehow **fix the possible RPF issue on the head-end**. The same must be done in every other case where unicast and multicast forwarding do not agree.

Solutions

- static mroute
- multicast BGP
- multicast topology in IS-IS
- mpls traffic-eng multicast-intact (for IGP routes)

You can use the command "`sh ip rpf`" in order to verify RPF issues.

Before

IOS

```
R2#sh ip rpf 19.19.19.19
 failed, no route exists
```

IOS-XR

```
GSR#sh pim rpf 2.2.2.2
Table: IPv4-Unicast-default
* 2.2.2.2/32 [115/30]
    via Null with rpf neighbor 0.0.0.0
```

In IOS-XR, the command "sh pim rpf" provides an output only if the address provided is already used as a multicast source. Use "sh pim rpf hash" to check in advance.

After fixing the RPF issue:

IOS

```
R2#sh ip rpf 19.19.19.19
RPF information for ? (19.19.19.19)
  RPF interface: FastEthernet1/0.24
  RPF neighbor: ? (26.2.4.4)
  RPF route/mask: 19.19.19.19/32
  RPF type: unicast (isis)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

IOS-XR

```
GSR#sh pim rpf 2.2.2.2
Table: IPv4-Multicast-default
* 2.2.2.2/32 [115/30]
    via GigabitEthernet0/1/0/1 with rpf neighbor 26.3.19.3
```

static mroute

IOS

```
ip mroute 2.2.2.2 255.255.255.255 Tunnel0
```

IOS-XR

```
multicast-routing
 address-family ipv4
  static-rpf 2.2.2.2 32 tunnel-te0 3.3.3.3
```

multicast-intact

When enabled on an IGP, the IGP automatically publishes to the RIB a parallel (or alternate) set of equal-cost next-hops for all IPv4 destinations learned through LS advertisements, for use solely by PIM. These next-hops are called **mcast-intact next-hops**. The mcast-intact next-hops have the following characteristics:

- They are guaranteed not to contain any IGP shortcuts (like TE tunnels).
- They are not used for unicast routing, but are used only by PIM to look up an IPv4 next-hop to a PIM source.
- They are not published to the FIB.
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set of equal-cost mcast-intact next-hops to the RIB. This attribute applies even when the native next-hops have no IGP shortcuts.

IOS

```
router ospf 1
 mpls traffic-eng multicast-intact
router isis 1
 mpls traffic-eng multicast-intact
```

IOS-XR

```
router ospf 1
 mpls traffic-eng multicast-intact
!
router isis 1
 address-family ipv4 unicast
  mpls traffic-eng multicast-intact
```

Multicast-intact doesn't work with TE forwarding-adjacency, use multicast BGP or static mroute.

multicast-intact vs static mroute in TE tunnels

- use **multicast-intact** to accept multicast traffic coming from **outside a TE tunnel**, when the unicast route is pointing inside the TE tunnel
- use **static mroute** to accept multicast traffic coming from **inside a TE tunnel**, when the unicast route is pointing outside the TE tunnel

IS-IS Multicast Topology

Multicast topology for ISIS allows the configuration of a separate IS-IS multicast topology for IPv4 or IPv6 routing, which runs a separate SPF.

IS-IS multicast inserts routes from the IS-IS multicast topology into the multicast-unicast table in the RIB for the corresponding address family. Since PIM uses this table, PIM uses routes from the multicast topology instead of routes from the unicast topology.

Multicast BGP

The multicast BGP database can be used by a a multicast routing protocol (i.e. PIM) to perform RPF lookups for multicast-capable sources. Thus, packets can be sent and accepted based on the multicast topology and not on the unicast topology.

This is an easy way to change the multicast routing without affecting unicast too. Static mroutes can also be used.

IOS

```
router bgp 65000
 no bgp default ipv4-unicast
 neighbor 7.7.7.7 remote-as 65000
 !
 address-family ipv4 multicast
  neighbor 7.7.7.7 activate
  network 192.168.7.0
 exit-address-family
```

IOS-XR

```
router bgp 65000
 address-family ipv4 multicast
  network 192.168.7.0/24
 !
 neighbor 7.7.7.7
  address-family ipv4 multicast
```

IOS

```
R8#sh ip rpf 192.168.7.0
RPF information for ? (192.168.7.0)
  RPF interface: FastEthernet0/0
  RPF neighbor: ? (192.168.78.7)
  RPF route/mask: 192.168.7.0/24
  RPF type: mbgp
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
```

Changing of BGP distance might be needed on the remote peer, if an IGP already provides a better route.

When you enable the multicast address-family under BGP, then these prefixes take precedence over the prefixes learned in the BGP unicast address-family; only the BGP prefixes learned in the multicast address-family are used for any multicast routing. Use "`show route ipv4 multicast`" in IOS-XR to display the ipv4 multicast routes.

You might need to also advertise a specific next-hop for the RPF route, if the default next-hop doesn't satisfy the RPF needs (recursive lookup might not be supported).

MSDP & Anycast RP

MSDP provides a way to connect multiple PIM-SM domains, so that **RPs can exchange information about active multicast sources**. It uses

MSDP sessions are formed between the RPs of various PIM domains, which are called MSDP peers.

MSDP is also used between **Anycast RPs** within a single PIM domain to synchronize information about the active sources being served by each Anycast-RP peer. Anycast RPs will each have the same IP address configured on a Loopback interface (making this the Anycast address) and will MSDP peer with each other using a separate loopback interface.

With Anycast RP, RP failover depends only on IGP convergence

MSDP RPs send SA (Source-Active) messages in order to notify each other of active multicast sources.

An MSDP RP sends an SA message to its peers each time it receives a PIM Register or Null-Register message from a Source DR about a new/active source. MSDP SA messages include the same multicast data that is also encapsulated in PIM Register or Null-Register messages.

IOS

```
interface Loopback99
 ip address 99.99.99.99 255.255.255.255
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
ip msdp peer 1.1.1.1 connect-source Loopback0
ip msdp originator-id Loopback0
!
ip pim rp-address 99.99.99.99
```

IOS-XR

```
interface Loopback99
  ipv4 address 99.99.99.99 255.255.255.255
!
interface Loopback0
  ipv4 address 1.1.1.1 255.255.255.255
!
router msdp
 originator-id Loopback0
 peer 2.2.2.2
  connect-source Loopback0
!
router pim
 address-family ipv4
  rp-address 99.99.99.99
```

The address on the interface used as anycast RP must be advertised into an IGP (preferably as external route to be able to play easily with the metric).

When using anycast IPv4 addresses in Loopbacks, it's good practice to hardcode all the protocol router-ids manually in order to avoid using the anycast address as router-id for a protocol.

In some software releases originator-id might not be required, but it's always good practice to configure it (especially in anycast topologies).

In current releases, when running Inter-AS MSDP, the peer/source IPs must agree with the BGP update source on each neighbor. If you also have a BGP peering session with an MSDP peer, you should use the same IP address for MSDP (peer/connect-source) as you do for BGP (update-source). If MDT is using a non-loopback interface due to eBGP with directly connected neighbor, you might get the message "`%MDT-4-LBSRC: VRF ABC: MDT X uses source address x.x.x.x from a non-loopback interface`". Besides changing the update-source of the eBGP peering, you can also use the "`bgp next-hop loopback X`" command under the appropriate vrf.

Use "`default-peer`" in IOS-XR when the peer address isn't in BGP.

If there is a single MSDP peer, then no RPF check takes place about the originator-id. If there are multiple MSDP peers, then you can put them under a mesh-group in order to disable the RPF check on them.

Links

- IETF - RFC 3618
- IETF - RFC 4611

Multicast Filtering

Setting an interface to PIM v1 at the border of a PIM domain prevents v2 Bootstrap messages from leaking to the neighboring PIM domain.

Use "`ip pim passive`" under an interface if you want to block the forwarding of PIM control plane traffic; only IGMP traffic will pass.

BSR filtering

IOS
```
interface X
 ip pim bsr-border
```

IOS-XR
```
router pim
 address-family ipv4
  interface X
   bsr-border
```

Auto-RP filtering

IOS-XR
```
multicast-routing
 address-family ipv4
```

```
  interface TenGigE0/2/0/0
   boundary MCAST-ACL
!
ipv4 access-list MCAST-ACL
 deny host 224.0.1.39
 deny host 224.0.1.40
 permit any
```

IGMP filtering

IOS

```
interface X
 ip igmp access-group IGMP-ACL
```

! match (,G)*

```
ip access-list extended IGMP-ACL
 permit ip host 0.0.0.0 host GROUP
```

! match (S,G) and (,G)*

```
ip access-list extended IGMP-ACL
 permit ip any host GROUP
```

Multicast Admission Control

- Global or per VRF
  - limit the number of mroutes that can be added to the global table
    - `ip multicast route-limit MAX-MROUTES THRESHOLD`
  - limit the number of mroutes that can be added to a particular MVRF table
    - `ip multicast vrf MVRF route-limit MAX-MROUTES THRESHOLD`
  - limit the number of mroute states created from IGMP membership reports
    - `ip igmp limit MAX-IGMPS`
- Per interface
  - limit the number of mroute states
    - `ip multicast limit MCAST-ACL MAX-MROUTES`
  - limit the number of mroutes states created from IGMP membership reports
    - `ip igmp limit MAX-IGMPS`
- Per neighbor
  - Limits the number of SA messages allowed in the SA cache from an MSDP peer
    - `ip msdp sa-limit MSDP-PEER MAX-SA-MESSAGES`

Check "Multicast Filtering" above for the format of MCAST-ACL.

Rate-limit

The "`ip multicast rate-limit`" interface command is not supported any more.

You need to use the the MQC syntax to define multicast traffic and then police it accordingly.

Multicast Fast Convergence

- tune PIM hellos (queries)
- tune RPF check interval/backoff
- MoFRR

Multicast-only FRR

The basic idea of MoFRR is to **send a secondary PIM join from the receiver toward the source on a backup path to a different upstream interface**. The network then receives two copies of the multicast stream over two separate and redundant paths through the network, but the redundant packets are discarded at topology merge points due to RPF checks. **When the primary path fails, it can switch over to the backup path instantly without issuing a new PIM join.**

Actually MoFRR is pre-building an alternate multicast tree in order to achieve faster convergence.

- RIB-based MoFRR
    - Supported on CRS and XR12000 series routers
    - Based on routing convergence
- Flow-based MoFRR
    - Supported ASR9k
    - Based on packet count per 30ms

IOS (15.2)

```
ip multicast rpf mofrr MOFRR-SG-ACL
!
ip access-list standard MOFRR-SG-ACL
 permit 10.10.10.0 0.0.0.255
```

IOS > 15.2 is required for MoFRR.

IOS-XR

```
router pim
 address-family ipv4
  mofrr rib MOFRR-SG-ACL
!
```

```
ipv4 access-list MOFRR-SG-ACL
 10 permit ipv4 host 1.1.1.1 host 239.3.3.3
```

Links

- IETF - draft-ietf-rtgwg-mofrr

Troubleshooting

Enable "debug ip mfib fs" and "debug ip mfib ps" on the receiver, and then send some pings from a source to an igmp join group on the receiver to check the results.

IPv6 Multicast

IOS

```
ipv6 multicast-routing
```

IOS-XR

```
multicast-routing
 address-family ipv6
  interface all enable
```

In IOS, all IPv6 interfaces are PIM enabled by default after enabling ipv6 multicast-routing.

IPv6 RP definition

IOS

```
ipv6 pim bsr candidate bsr 2002:2:2::2
ipv6 pim bsr candidate rp 2002:2:2::2
!
ipv6 pim rp-address 2002:2:2::2
```

IOS-XR

```
router pim
 address-family ipv6
  rp-address 2002:2:2::2
  bsr candidate-rp 2002:2:2::2
  bsr candidate-bsr 2002:2:2::2
```

MLD

IOS

```
ipv6 multicast-routing
!interface Loopback0
 ipv6 mld join-group FF99::99
```

IOS-XR

```
multicast-routing
 address-family ipv6
  interface all enable
!
router mld
 interface Loopback0
  join-group ff99::99
```

MLDv2 is used by default.

Verification

```
R2#sh ipv6 mroute
Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,
       C - Connected, L - Local, I - Received Source Specific Host
Report,
       P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,
       J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, State

(*, FF99::99), 00:00:08/00:03:21, RP 2002:2:2::2, flags: S
  Incoming interface: Tunnel4
  RPF nbr: 2002:2:2::2
  Immediate Outgoing interface list:
    Ethernet0/2, Forward, 00:00:08/00:03:21

R2#ping FF99::99
Output Interface: Ethernet0/2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FF99::99, timeout is 2 seconds:
Packet sent with a source address of 2002:2:2:27::2

Reply to request 0 received from 2002:2:2::7, 60 ms
Reply to request 1 received from 2002:2:2::7, 0 ms
Reply to request 2 received from 2002:2:2::7, 4 ms
Reply to request 3 received from 2002:2:2::7, 4 ms
Reply to request 4 received from 2002:2:2::7, 0 ms
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/13/60 ms
5 multicast replies and 0 errors.
```

Almost everything cli-wise is similar to IPv4.

Static mroutes

IOS
```
ipv6 route 2002:2:2::2/128 Tunnel0 multicast
```

IOS-XR
```
multicast-routing
 address-family ipv6
  static-rpf 2002:2:2::2 128 tunnel-te0 2002:3::1
```

NSF

IOS-XR
```
router pim
 nsf lifetime 30
!
router igmp
 nsf lifetime 30
!
router mld
 nsf lifetime 30
```

Generally, configure the IGMP NSF and PIM NSF lifetime values to be equal or larger than the query or join query interval, but less than the holdtime

Multipath

By default, if ECMP paths are available, the RPF for multicast traffic will be based on the highest IP address (aka highest PIM neighbor).

When the '`ip multicast multipath`' command is configured, the multicast load splitting will be based on the source address of the stream. PIM Joins will be distributed over the different ECMP links based on a hash of the source address.

Multicast multipath must be enabled on the receiver side of the ECMP path.

IOS
```
ip multicast multipath
```

IOS

```
R2#sh ip rpf 19.19.19.19
RPF information for ? (19.19.19.19)
  RPF interface: FastEthernet0/0.24
  RPF neighbor: ? (20.2.4.4)
  RPF route/mask: 19.19.19.19/32
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  Multicast Multipath enabled.
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

By default only the source address is used in the calculation. Better load-splitting can be achieved by using the "s-g-hash next-hop-based" options.

When the ip multicast multipath command is enabled, the presence of PIM hello message from neighbors is not considered; that is, the chosen RPF neighbor does not depend on whether or not PIM hello messages are received from that neighbor; it only depends on the presence or absence of an equal-cost route entry.

If using BGP and multicast, then you must also enable multipath on BGP.

If using static mroutes, then you need to somehow create two (or more) different static mroutes because only one is accepted. You can use a dummy ip address and two (or more) static ip routes in order to achieve this.

IOS

```
ip route 192.168.1.1 255.255.255.255 20.2.3.3
ip route 192.168.1.1 255.255.255.255 20.2.4.4
!
ip mroute 19.19.19.19 255.255.255.255 192.168.1.1
```

In the above example, multicast load-balancing occurs between 20.2.3.3 and 20.2.4.4 for source 19.19.19.19 (192.168.1.1 is the dummy address used).

Use "sh ip multicast rpf tracked" to verify the multiple rpf paths.

Multicast VPN

Multicast VPN is defined in RFC 6513 and RFC 6514.
Cisco's Multicast VPN is defined in RFC 6037.

Two solutions:

- PIM/GRE mVPN or draft-rosen (RFC 6037)
  - PIM adjacencies between PEs to exchange mVPN routing information
  - unique multicast address per VPN
  - per-VPN PIM adjacencies between PEs and CEs
  - per-VPN MDT (GRE) tunnels between PEs
  - data MDT tunnels for optimization
- BGP/MPLS mVPN or NG mVPN
  - BGP peerings between PEs to exchange mVPN routing information
  - PIM messages are carried in BGP
  - BGP autodiscovery for inter-PE tunnels
  - MPLS P2MP inclusive tunnels between PEs
  - selective tunnels for optimization

Only the PIM/GRE mVPN model (Cisco's original implementation) is described below.

MVPN

MVPN combines multicast with MPLS VPN. PE routers establish virtual PIM neighborships with other PE routers that are connected to the same VPN.

The VPN-specific multicast routing and forwarding database is referred to as **MVRF**.

A **MDT** (multicast distribution tree) tunnel interface is an interface that MVRF uses to access the multicast domain. MDT tunnels are point-to-multipoint.

Multicast packets are sent from the CE to the ingress PE and then encapsulated and transmitted across the core (over the MDT tunnel). At the egress PE, the encapsulated packets are decapsulated and then sent to the receiving CE.

When sending customer VRF traffic, PEs encapsulate the traffic in their own (S,G) state, where the G is the MDT group address, and the S is the MDT source for the PE. By joining the (S,G) MDT of its PE neighbors, a PE router is able to receive the encapsulated multicast traffic for that VRF.

All VPN packets passing through the provider network are viewed as native multicast packets and are routed based on the routing information in the core network.

To support MVPN, PE routers only need to support native multicast routing.

RTs should be configured so that the receiver VRF has unicast reachability to prefixes in the source VRF.

Data MDT

MVPN also supports optimized VPN traffic forwarding for high-bandwidth applications that have sparsely distributed receivers.

A dedicated multicast group can be used to encapsulate packets from a specific source and an optimized MDT can be created to send traffic only to PE routers connected to interested receivers.

A unique group per vrf should be used on the PEs.

Configuration

IOS

```
ip multicast-routing
!
ip pim ssm default
!
interface Loopback0
 ip pim sparse-mode
!
interface X
 ip pim sparse-mode
!
ip multicast-routing vrf VPN
!
vrf definition VPN
 address-family ipv4
  mdt default x.x.x.x
  mdt data x.x.x.x y.y.y.y
 exit-address-family
!
router bgp 100
 address-family ipv4 mdt
  neighbor x.x.x.x activate
 exit-address-family
```

IOS-XR

```
multicast-routing
 address-family ipv4
  interface Loopback0
   enable
   !
  mdt source Loopback0
 !
 vrf VPN
  address-family ipv4
   mdt default ipv4 x.x.x.x
   mdt data y.y.y.y/24
   interface all enable
!
router bgp 100
 address-family ipv4 mdt
 !
 neighbor x.x.x.x
  address-family ipv4 mdt
```

"mdt source" is required in IOS-XR (it can be configured under the VRF if it's specific for it).

Sparse mode must be activated on all physical interfaces where multicast will be passing through (global or VRF ones) and **on the loopback interface used for the BGP VPNv4 peerings**.

The RP setup of the CEs must agree with the VRF RP setup on the PEs. In case you manually define the RP (static RP) on the CEs, then this must be done on the PEs too (inside the vrf).

Verification

- There should be (S,G) entries for each BGP neighbor, where S=BGP loopback and G=MDT default address
- There should be a bidirectional PIM adjacency across a tunnel between the PEs, but inside each PE's VRF
- If an RP is used on a CE, then each remote CE should know this RP
- Sources/Receivers from any site should be viewable on the RP
- There should be an MDT data (S,G) entry for each pair of customer (S,G) entries

Verification (using only a default mdt)

MDT default (S,G) entries

IOS

```
R5#sh ip mroute sum
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E -
```

```
Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.255.1), 00:34:36/stopped, RP 10.0.0.1, OIF count: 1, flags:
SJCFZ
  (10.0.0.6, 239.255.255.1), 00:24:11/00:02:18, OIF count: 1, flags: JTZ
  (10.0.0.5, 239.255.255.1), 00:34:35/00:02:54, OIF count: 1, flags: FT

R5#sh ip mroute 239.255.255.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E -
Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.255.1), 00:46:12/stopped, RP 10.0.0.1, flags: SJCFZ
  Incoming interface: FastEthernet0/0.15, RPF nbr 10.1.5.1
  Outgoing interface list:
    MVRF VPN, Forward/Sparse, 00:46:12/00:01:46

(10.0.0.6, 239.255.255.1), 00:35:47/00:02:28, flags: JTZ
  Incoming interface: FastEthernet0/0.57, RPF nbr 10.5.7.7
  Outgoing interface list:
    MVRF VPN, Forward/Sparse, 00:35:47/00:01:46

(10.0.0.5, 239.255.255.1), 00:46:12/00:03:19, flags: FT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0.57, Forward/Sparse, 00:35:46/00:03:11
```

```
R5#sh bgp ipv4 mdt all 10.0.0.6/32
BGP routing table entry for 100:1:10.0.0.6/32        version 2
Paths: (1 available, best #1, table IPv4-MDT-BGP-Table)
  Not advertised to any peer
  Local
    10.0.0.6 from 10.0.0.1 (10.0.0.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Originator: 10.0.0.6, Cluster list: 10.0.0.1, 10.0.0.20,
      MDT group address: 239.255.255.1
```

```
R5#sh ip pim mdt
  * implies mdt is the default MDT
  MDT Group/Num   Interface   Source                   VRF
* 239.255.255.1   Tunnel1     Loopback0                VPN
```

```
R5#sh ip pim mdt bgp
MDT (Route Distinguisher + IPv4)               Router ID         Next Hop
  MDT group 239.255.255.1
   100:1:10.0.0.6                              10.0.0.1          10.0.0.6
```

Verification (using a default and a data mdt)

MDT default (S,G) entries
MDT data (S,G) entries

IOS

```
R2#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E -
Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(2.2.2.2, 232.0.0.1), 00:08:53/00:03:27, flags: sT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0.24, Forward/Sparse, 00:08:53/00:03:27

(19.19.19.19, 232.0.0.1), 00:50:48/stopped, flags: sTIZ
  Incoming interface: FastEthernet0/0.24, RPF nbr 20.2.4.4
  Outgoing interface list:
    MVRF VPN, Forward/Sparse, 00:50:48/00:00:11

(19.19.19.19, 232.0.1.0), 00:08:23/00:00:12, flags: sTIZ
  Incoming interface: FastEthernet0/0.24, RPF nbr 20.2.4.4
  Outgoing interface list:
    MVRF VPN, Forward/Sparse, 00:02:47/00:00:12

(19.19.19.19, 232.0.1.1), 00:01:59/00:01:00, flags: sTIZ
  Incoming interface: FastEthernet0/0.24, RPF nbr 20.2.4.4
  Outgoing interface list:
    MVRF VPN, Forward/Sparse, 00:01:59/00:01:00
R2#sh ip pim mdt
  * implies mdt is the default MDT
  MDT Group/Num   Interface   Source                   VRF
* 232.0.0.1       Tunnel0     Loopback0                VPN
  232.0.1.0       Tunnel0     Loopback0                VPN
  232.0.1.1       Tunnel0     Loopback0                VPN



R2#sh ip pim mdt bgp
MDT (Route Distinguisher + IPv4)               Router ID         Next Hop
  MDT group 232.0.0.1
   100:1:19.19.19.19                           19.19.19.19
19.19.19.19
```

In both scenarios, you can also verify the mGRE tunnels by looking at the tunnel interface itself.

IOS

```
R5#sh int tun1 | i protocol/transport
  Tunnel protocol/transport multi-GRE/IP
```

When all PIM adjacencies come up, as PIM neighbors in a VRF you should see all the other MDT PEs though a tunnel and all the local connected CEs through a physical interface.

IOS

```
R5#sh ip pim vrf VPN nei
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable
Neighbor          Interface                Uptime/Expires    Ver   DR
Address
```

```
Prio/Mode
192.168.59.9      FastEthernet0/0.59      00:00:22/00:01:22 v2    1 / DR
S G
10.0.0.6          Tunnel1                 00:25:52/00:01:27 v2    1 / DR
S P G
```

PIM inside a VRF Tunnel

IOS

```
interface Tunnel1
 ip vrf forwarding VPN-A
 ip address 99.99.99.1 255.255.255.0
 ip pim sparse-mode
 tunnel source 10.0.0.1
 tunnel destination 10.0.0.2
 tunnel vrf VPN-B
!
interface Tunnel1
 ip vrf forwarding VPN-A
 ip address 99.99.99.2 255.255.255.0
 ip pim sparse-mode
 tunnel source 10.0.0.2
 tunnel destination 10.0.0.1
 tunnel vrf VPN-B
```

"`ip vrf forwarding`" defines the vrf under which the tunnel (99.99.99.0/24) operates; above it's VPN-A.

"`tunnel vrf`" defines the vrf which is used to build the tunnel (from 10.0.0.1 to 10.0.0.2); above it's VPN-B. If the tunnel source and destination are in the global routing table, then you don't need to define their vrf with the "`tunnel vrf X`" command.

Extranet

An extranet site can have either the multicast source or the receivers (otherwise multicast happens intra-as).

The Source PE has the multicast source behind a directly connected CE through the Source MVRF

The Receiver PE has one or more receivers behind a directly connected CE through the Receiver MVRF

In order to achieve multicast connectivity between the Source and Receiver PEs, you must have the same default MDT group in the source and receiver MVRF.

Two solutions:

- Configure the Receiver MVRF on the Source PE router
  - you need each receiver MVRF copied on the Source PE router
- Configure the Source MVRF on the Receiver PE routers
  - you need the Source MVRF copied on all interested Receiver PE routers

In both cases, the receiver MVRF (wherever placed) must import the source MVRF's RT.

Only PIM-SM and PIM-SSM are supported.

The multicast source and the RP must reside in the same site of the MVPN, behind the same PE router.

Receiver MVRF on the Source PE

Source PE (IOS)

```
ip vrf VPN1-S-MVRF
 rd 100:1
 route-target export 100:1
 route-target import 100:1
 mdt default 232.1.1.1
!
ip vrf VPN2-R-MVRF
 rd 100:2
 route-target export 100:2
 route-target import 100:2
 route-target import 100:1
 mdt default 232.2.2.2
!
ip multicast-routing
ip multicast-routing vrf VPN1-S-MVRF
ip multicast-routing vrf VPN2-R-MVRF
```

Receiver PE (IOS)

```
ip vrf VPN2-R-MVRF
 rd 100:2
 route-target export 100:2
 route-target import 100:2
 route-target import 100:1
 mdt default 232.2.2.2
!
ip multicast-routing
ip multicast-routing vrf VPN2-R-MVRF
```

Source MVRF on the Receiver PE

Source PE (IOS)

```
ip vrf VPN1-S-MVRF
 rd 100:1
 route-target export 100:1
 route-target import 100:1
 mdt default 232.1.1.1
!
ip multicast-routing
ip multicast-routing vrf VPN1-S-MVRF
```

Receiver PE (IOS)

```
ip vrf VPN1-S-MVRF
 rd 100:1
 route-target export 100:1
 route-target import 100:1
 mdt default 232.1.1.1
!
ip vrf VPN2-R-MVRF
 rd 100:2
 route-target export 100:2
 route-target import 100:2
 route-target import 100:1
 mdt default 232.2.2.2
!
ip multicast-routing
ip multicast-routing vrf VPN1-S-MVRF
ip multicast-routing vrf VPN2-R-MVRF
```

What matters most in both cases when doing the MVRF replication, is to have the same MDT on a MVRF on the Source PE and on a MVRF on the Receiver PE (excluding the Source MVRF).

Fixing RPF

There are two options:

static mroute between VRFs

Receiver PE (IOS)

```
ip mroute vrf VPN2-R-MVRF 192.168.1.1 255.255.255.255 fallback-lookup vrf
VPN1-S-MVRF
```

group-based VRF selection

Receiver PE (IOS)

```
ip multicast vrf VPN2-R-MVRF rpf select vrf VPN1-S-MVRF group-list 1
ip multicast vrf VPN2-R-MVRF rpf select vrf VPN3-S-MVRF group-list 3
!
access-list 1 permit 231.0.0.0 0.255.255.255
access-list 3 permit 233.0.0.0 0.255.255.255
```

Inter-AS MVPN

To establish a Multicast VPN between two ASes, a MDT-default tunnel must be setup between the involved PE routers. The appropriate MDT-default group is configured on the PE router and is unique for each VPN.

All three (A,B,C) inter-as options are supported. For option A nothing extra is required since every AS is completely isolated from the others.

In order to solve the various RPF issues imposed by the limited visibility of PEs between different ASes, each VPNv4 route carries a new transitive attribute (the **BGP connector** attribute) that defines the route's originator.

Inside a common AS, the BGP connector attribute is the same as the next hop. Between ASes the BGP connector attribute stores (in case of ipv4 mdt) the ip address of the PE router that originated the VPNv4 prefix and is preserved even after the next hop attribute is rewritten by ASBRs.

The BGP connector attribute also helps ASBRs and receiver PEs insert the RPF vector needed to build the inter-AS MDT for source PEs in remote ASes.

The **RPF proxy vector** is a PIM TLV that contains the ip address of the router that will be used as proxy for RPF checks (helping in the forwarding of PIM Joins between ASes).

A new PIM hello option has also been introduced along with the PIM RPF Vector extension to determine if the upstream router is capable of parsing the new TLV. An RPF Vector is included in PIM messages only when all PIM neighbors on an RPF interface support it.

The RPF proxy (usually the ASBR) removes the vector for the PIM Join message when it sees itself in it.

- BGP connector attribute
    - used in RPF checks inside a VRF
- RPF proxy
    - used in RPF checks in the core

Configuration Steps

- Option A
    - no MDT sessions between ASes is required
    - intra-as MDT sessions are configured as usual
- Option B
    - intra-as MDT sessions between PEs, ASBRs and RRs
    - inter-as MDT session between ASBRs
    - RPF proxy vector on all PEs for their VRFs

    - RPF proxy vector on all Ps and ASBRs
    - next-hop-self on the MDT ASBRs
- Option C
    - intra-as MDT sessions between PEs and RRs
    - inter-as MDT sessions between RRs
    - RPF proxy vector on all PEs for their VRFs
    - RPF proxy vector on all Ps and ASBRs
    - next-hop-unchanged on the MDT RRs

MSDP will be required if using an RP on both ASes. Prefer to use SSM in the core of both ASes.

Links

- IETF - RFC 5496

BFD

BFD (Bidirectional Forwarding Detection) is defined in RFC 5880.
BFD for one-hop IPv4/IPv6 is defined in RFC 5881.
BFD for multi-hop is defined in RFC 5883.
BFD for MPLS LSPs is defined in RFC 5884.

Common BFD applications

- Control plane liveliness detection
- Tunnel endpoint liveliness detection
- Trigger mechanism for IP/MPLS FRR
- MPLS date plane failure detection

BFD advantages

- failure detection in sub-sec
- generic/consistent failure detection mechanism for all protocols
- less CPU intensive if distributed to the data plane

BFD modes

- Asynchronous mode
  - continuous and periodic BFD packets
- Demand mode
  - BFD packets only after a demand

BFD echo (where a stream of echo packets is sent and received) is the most common function for both modes.

Cisco supports the asynchronous mode and the echo function by default.

BFD payload control packets are encapsulated in UDP packets

- destination port 3784
- source port 49152

Echo packets are also encapsulated in UDP packets

- destination port 3785
- source port 3785

BFD control packets are always sent as unicast packets to the BFD peer.

The encapsulation of BFD Control packets for multihop application in IPv4 and IPv6 is identical to that above, except that the UDP destination port is 4784.

Each system reports in the BFD Control packet how rapidly it would like to transmit BFD packets, as well as how rapidly it is prepared to receive them. This allows either system to determine the max packet rate (minimum interval) in both directions.

To establish a BFD neighbor in IOS-XR, BFD must either be configured under an IGP or as a static route.

BFD Configuration

BFD can be configured:

- under an interface for a specific protocol (IOS)
- under the protocol process for a specific interface (IOS-XR)
- under the protocol process for all interfaces (IOS)
- under the protocol process for all neighbors (IOS-XR)
- under the protocol process for a specific neighbor (IOS,IOS-XR)

In all cases, BFD timers (interval, min_rx, multiplier) must be defined under the relevant interfaces too. BFD and BGP in IOS-XR is the exception, as show below.

If you are using **BFD with uRPF** on a particular interface, then you need to use the "`echo disable`" command to **disable the echo mode** on that interface, otherwise echo packets are rejected. You can disable echo mode for the entire router, or for an individual interface.

BFD can be combined with "`carrier-delay 0`" for quicker protocol reaction.

In IOS-XR, "`bfd fast-detect`" is required in order to start the BFD process.

BFD might cause crashes and malfunctions when enabled on GNS3 emulated routers.

Instead of running BFD at the link-level, you can also run BFD at the LSP level (aka across the LSP), something that offers faster detection in case of path protection.

BFD & BGP

IOS

```
interface X
 bfd interval 300 min_rx 300 multiplier 3
!
router bgp 100
 neighbor 2.2.2.2 fall-over bfd
```

IOS-XR

```
router bgp 100
 neighbor 2.2.2.2
  bfd fast-detect
  bfd multiplier 3
  bfd minimum-interval 300
```

In IOS-XR, BFD parameters (multiplier, min-interval) can be configured for all BGP neighbors (under the BGP process) or for a specific BGP neighbor. BFD activation is always per neighbor.

It is generally not recommended to use BFD for iBGP, when the underlying IGP is already doing so.

BFD & ISIS

IOS

```
interface X
 bfd interval 150 min_rx 150 multiplier 3
 isis bfd
```

or

IOS

```
router isis 1
 bfd all-interfaces
!
interface X
 bfd disable
```

IOS-XR

```
router isis 1
 interface X
  bfd minimum-interval 150
  bfd multiplier 3
  bfd fast-detect ipv4
```

BFD & OSPF

IOS

```
interface X
 bfd interval 150 min_rx 150 multiplier 3
 ip ospf bfd
```

or

IOS

```
router ospf 1
 bfd all-interfaces
!
interface X
 bfd disable
```

IOS-XR

```
router ospf 1
 area 0
 interface X
  bfd minimum-interval 150
  bfd multiplier 3
  bfd fast-detect ipv4
```

BFD parameters can also be configured directly under the ospf process.

BFD establishes sessions from a neighbor to a DR or BDR, only when the neighbor state is full.

BFD does not establish sessions between DR-Other neighbors (for example, when their OSPF states are both 2-way)

BFD & RSVP-TE/FRR

IOS

```
ip rsvp signalling hello bfd
!
interface X
 bfd interval 300 min_rx 300 multiplier 3
 ip rsvp signalling hello bfd
```

IOS-XR

```
mpls traffic-eng
 interface TenGigE0/0/0/0
  bfd fast-detect
 !
 bfd minimum-interval 150
 bfd multiplier 3
```

IOS relates the BFD configuration to RSVP configuration, while IOS-XR relates it to MPLS TE configuration.

IOS

```
R3#sh bfd neighbors detail

NeighAddr                         LD/RD    RH/RS     State     Int
12.3.4.4                           1/1     Up        Up        Fa0/0.34
Session state is UP and using echo function with 300 ms interval.
OurAddr: 12.3.4.3
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 1000000, MinRxInt: 1000000, Multiplier: 3
Received MinRxInt: 1000000, Received Multiplier: 3
Holddown (hits): 0(0), Hello (hits): 1000(17)
Rx Count: 15, Rx Interval (ms) min/max/avg: 1/996/834 last: 312 ms ago
Tx Count: 18, Tx Interval (ms) min/max/avg: 1/1000/845 last: 344 ms ago
Elapsed time watermarks: 0 0 (last: 0)
Registered protocols: CEF TE/FRR
Uptime: 00:00:11
Last packet: Version: 1                  - Diagnostic: 0
             State bit: Up               - Demand bit: 0
             Poll bit: 0                 - Final bit: 0
             Multiplier: 3               - Length: 24
             My Discr.: 1                - Your Discr.: 1
             Min tx interval: 1000000    - Min rx interval: 1000000
             Min Echo interval: 300000
```

BFD & IPv6

BFD for IPv6 (BFDv6) is not supported in IOS-XR < 4.1.

BFDv6 supports both global and link-local IPv6 addresses for neighbor session creation. BFDv6 sessions select automatically source addresses to match the neighbor address types. Each type of IPv6 address on the local router must be paired with the same type on the peer router.

You can have an IPv6 static route (2001:DB8::/64) pointing to a peer router associated with a BFD neighbor (2001::1). In order to remove this IPv6 static route from the RIB if the BFD neighbor goes down, you must associate the static route with the BFD neighbor

IOS

```
ipv6 route 2001:DB8::/64 Ethernet0/0 2001::1
ipv6 route static bfd Ethernet0/0 2001::1
```

In BFD associated mode (default), an IPv6 static route is automatically associated with an IPv6 BFD neighbor, if the static route next-hop matches exactly the static BFD neighbor. If you want to avoid this, you can configure the static route as "unassociated".

QoS

- Congestion Management
  - WFQ (Weighted Fair Queuing)
    - fair-queue
  - CQ (Custom Queuing)
    - custom-queue
  - PQ (Priority Queuing)
    - priority-queue
  - CBWFQ (Class-Based WFQ)
    - MQC & bandwidth
  - LLQ (Low Latency Queuing)
    - MQC & priority
- Congestion Avoidance
  - Tail-Drop
    - default
  - WRED (Weighted Random Early Detection)
    - random-detect
  - Class-Based WRED
    - MQC & random-detect

Don't forget the set the interface "`max-reserved-bandwidth`" if more than 75% is required.

Weighted Fair Queuing

- configure congestion threshold
- configure dynamic queues
- configure reservable queues

IOS

```
interface Serial2/0
 fair-queue 512 256 128
```

IOS

```
R2#sh queueing fair
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial2/0           512        256      128       8       1
```

```
R2#sh queueing int s2/0
Interface Serial2/0 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/512/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
```

Custom Queuing

- assign packets/protocols to the 16 queues
- configure custom queue parameters
- assign custom queue to interface

Not applicable to subinterfaces.

IOS

```
queue-list 1 protocol ip 0 list 100
queue-list 1 protocol ipv6 1
queue-list 1 protocol pppoe 2
queue-list 1 default 3
!
queue-list 1 queue 0 byte-count 5000 limit 500
queue-list 1 queue 1 byte-count 2500 limit 250
queue-list 1 queue 2 byte-count 1500 limit 150
queue-list 1 queue 3 byte-count 1500 limit 50
!
interface FastEthernet0/0
 custom-queue-list 1
```

IOS

```
R4#sh queueing custom
Current custom queue configuration:

List   Queue  Args
1      3      default
1      2      protocol ip          list 100
1      1      protocol ipv6
1      2      protocol pppoe-sessi
1      0      byte-count 5000 limit 500
1      1      byte-count 2500 limit 250
1      2      limit 150
1      3      limit 50
```

```
R4#sh queueing int fa0/0
Interface FastEthernet0/0 queueing strategy: custom

Output queue utilization (queue/count)
        0/680 1/0 2/48085 3/1 4/0 5/0 6/0 7/0 8/0
        9/0 10/0 11/0 12/0 13/0 14/0 15/0 16/0
```

Priority Queuing

- assign packets/protocols to the 4 priority queues
- configure priority queue parameters
- assign priority queues to interface

Not applicable to subinterfaces.

IOS

```
priority-list 1 protocol ip high list 100
priority-list 1 protocol ipv6 medium
priority-list 1 protocol pppoe normal
priority-list 1 default low
!
priority-list 1 queue-limit 500 250 150 50
!interface FastEthernet0/0
 priority-group 1
```

IOS

```
R4#sh queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
1      low    default
1      normal protocol ip          list 100
1      medium protocol ipv6
1      normal protocol pppoe-sessi
1      high   limit 500
1      medium limit 250
1      normal limit 150
1      low    limit 50

R4#sh queueing int fa0/0
Interface FastEthernet0/0 queueing strategy: priority

Output queue utilization (queue/count)
        high/2349 medium/0 normal/48090 low/2
```

WRED

- choose between dscp and precedence WRED
- configure dscp/prec parameters

IOS

```
interface Serial2/0
 random-detect dscp-based
```

IOS

```
R2#sh queueing int s2/0
Interface Serial2/0 queueing strategy: random early detection (WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0

   dscp          Random drop      Tail drop    Minimum Maximum  Mark
                  pkts/bytes       pkts/bytes    thresh  thresh  prob
   af11      0/0              0/0           33      40  1/10
   af12      0/0              0/0           28      40  1/10
   af13      0/0              0/0           24      40  1/10
   af21      0/0              0/0           33      40  1/10
   af22      0/0              0/0           28      40  1/10
   af23      0/0              0/0           24      40  1/10
   af31      0/0              0/0           33      40  1/10
   af32      0/0              0/0           28      40  1/10
   af33      0/0              0/0           24      40  1/10
   af41      0/0              0/0           33      40  1/10
   af42      0/0              0/0           28      40  1/10
   af43      0/0              0/0           24      40  1/10
    cs1      0/0              0/0           22      40  1/10
    cs2      0/0              0/0           24      40  1/10
    cs3      0/0              0/0           26      40  1/10
    cs4      0/0              0/0           28      40  1/10
    cs5      0/0              0/0           31      40  1/10
    cs6      0/0              0/0           33      40  1/10
    cs7      0/0              0/0           35      40  1/10
     ef      0/0              0/0           37      40  1/10
   rsvp      0/0              0/0           37      40  1/10
default       0/0              0/0           20     40  1/10
```

If WRED is used for a used-defined class, then bandwidth reservation must be made too.

If WRED is used for the default class, then either bandwidth reservation of fair-queue must be used too.

Other

These topics are not considered critical because they aren't usually used as a base for something else, but they can easily give you some points if configured correctly.

Having a general idea is probably enough, as long as you know where to look in the documentation for the details.

MAC accounting

Use it to collect statistics about traffic per mac address.

IOS

```
interface FastEthernet0/0
 ip accounting mac-address input
 ip accounting mac-address output
```

IOS-XR

```
interface TenGigE0/2/0/0
 mac-accounting ingress
 mac-accounting egress
```

IP/precedence accounting

Use it to collect statistics about traffic per ip address or per ip precedence.

IOS

```
R3(config-subif)#ip accounting ?
  access-violations  Account for IP packets violating access lists on
this
                     interface
  output-packets     Account for IP packets output on this interface
  precedence         Count packets by IP precedence on this interface
  <cr>
```

IOS

```
R3(config)#ip ?
Global IP configuration subcommands:
...
  accounting-list         Select hosts for which IP accounting
information is
```

```
                            kept
  accounting-threshold      Sets the maximum number of accounting entries
  accounting-transits       Sets the maximum number of transit entries
...
```

carrier-delay

For fast convergence use low (or 0) timers, especially for the down timer.
For cpu optimization (after routing instability due to small interface flaps) use higher timers.

i.e. if there is a backup circuit available:

IOS

```
interface FastEthernet0/0
 carrier-delay msec 50
```

IOS-XR

```
interface TenGigE0/2/0/0
 carrier-delay down 0 up 3000
```

IP event dampening

It's a mechanism to suppress the effects of excessive interface flapping events on routing protocols and routing tables.

Parameters:

- half-life period
    - the penalty is reduced by half after each half-life period (assuming the interface has stopped flapping)
    - default: 5 sec
- reuse threshold
    - when the penalty drops to the reuse threshold, the route is unsuppressed
    - default: 1000 penalties
- suppress threshold
    - when the accumulated penalty reaches the suppress threshold, the interface is placed in the dampened state and the route is suppressed
    - default: 2000 penalties
- max suppress
    - the maximum amount of time an interface can remain dampened when a penalty is assigned to it
    - default: 4 x half-life sec

- restart penalty
    - initial penalty applied to an interface when it comes up after a router reload
    - default: 2000 penalties

In Cisco software, default penalty is 1000.

IOS

```
interface FastEthernet0/0
 dampening 30 2000 5000 60
```

```
R3#sh interfaces dampening
FastEthernet0/0
  Flaps Penalty    Supp ReuseTm   HalfL  ReuseV   SuppV  MaxSTm    MaxP
Restart
      0       0   FALSE       0      30    2000    5000      60    8000
      0
```

IOS-XR

```
interface TenGigE0/2/0/0
 dampening 1 2000 5000 2
```

IOS-XR half-life and max-suppress values are in mins, while in IOS they are in secs.

Use "`debug dampening interface`" to verify the dampening procedure.

preconfigure interfaces

If you don't have the actual linecards and/or interfaces, you can use preconfiguration in order to create interfaces in advance, which is a nice way of testing configurations.

IOS-XR

```
CRS(config)#int preconfigure pos ?
  R/S/I/P  Preconfig interface in Rack/Slot/Instance/Port format
```

Type-7 passwords

You can use a key-chain to recover a type 7 password.

NAT

"`sh ip nat translations`" shows translation for both global routing table and VRFs.

You can use "`ip nat inside`" on an interface even when the traffic passing through it is labeled.

IP SLA

IP SLA uses active traffic monitoring for measuring network performance.

The information collected includes data about:

- response time
- one-way latency
- jitter
- packet loss
- voice quality scoring
- network resource availability
- application performance
- server response time

Configuration Steps

- Enable the IP SLAs responder (if required)
- Configure the required IP SLAs operation type
- Configure any options available for the specified IP SLAs operation type
- Configure threshold conditions (if required)
- Schedule the operation to run
- Collect the statistics

You can use the following command to find the supported operation types to use for SLA:

IOS

```
R1#sh ip sla application
        IP Service Level Agreement Technologies
Version: Round Trip Time MIB 2.2.0, Infrastructure Engine-II

Supported Operation Types:
        802.1agEcho VLAN, EVC, Port, 802.1agJitter VLAN, EVC, Port
        dhcp, dns, echo, ftp, http, jitter, lspGroup, lspPing
        lspPingPseudowire, lspTrace, , pathEcho, pathJitter
        tcpConnect, udpEcho
Supported Features:
        IPSLAs Event Publisher
```

Common parameters:

- **frequency (sec)**
  - the rate at which a specified IP SLAs operation repeats
- **request-data-size (bytes)**
  - the protocol data size in the payload of an IP SLAs operation's request packet
- **threshold (msec)**
  - the upper threshold value for calculating network monitoring statistics
- **timeout (msec)**
  - the amount of time an IP SLAs operation waits for a response from its request packet

It's obvious that if you set the timeout < threshold, then you'll never get over-threshold statistics.

Configuration

IOS

```
ip sla 1
 icmp-echo 2.2.2.2
 timeout 40
 threshold 20
 frequency 30
ip sla schedule 1 life 600 start-time now
```

Verification

IOS

```
R1#sh ip sla statistics detail

Round Trip Time (RTT) for       Index 1
Type of operation: icmp-echo
        Latest RTT: 20 ms
Latest operation start time: *13:25:14.311 UTC Sun Jan 26 2014
Latest operation return code: OK
Over thresholds occurred: FALSE
Number of successes: 1
Number of failures: 0
Operation time to live: 588 sec
Operational state of entry: Active
Last time this entry was reset: *13:25:14.307 UTC Sun Jan 26 2014

R1#sh ip sla statistics detail

Round Trip Time (RTT) for       Index 1
Type of operation: icmp-echo
        Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: *13:25:44.311 UTC Sun Jan 26 2014
Latest operation return code: Timeout
Over thresholds occurred: FALSE
```

```
Number of successes: 1
Number of failures: 1
Operation time to live: 549 sec
Operational state of entry: Active
Last time this entry was reset: *13:25:14.307 UTC Sun Jan 26 2014

R1#sh ip sla statistics detail

Round Trip Time (RTT) for       Index 1
Type of operation: icmp-echo
        Latest RTT: 24 ms
Latest operation start time: *13:26:44.311 UTC Sun Jan 26 2014
Latest operation return code: Over threshold
Over thresholds occurred: TRUE
Number of successes: 1
Number of failures: 3
Operation time to live: 505 sec
Operational state of entry: Active
Last time this entry was reset: *13:25:14.307 UTC Sun Jan 26 2014
```

If you want to change the parameters of an already running sla operation, you have to remove its schedule first and then change it.

If you want to change the type of an already existing sla operation, you have to remove it completely and start over.

IP SLA Responder

The IP SLA Responder listens on a specific port (**UDP 1967**) for control protocol messages sent by a IP SLAs operation. Upon receipt of the control message, the responder will enable the specified UDP or TCP port for the specified duration.

It can help avoid measuring the processing delay and **provide larger accuracy**, because it allows the target device to take two time stamps both when the packet arrives on the interface at interrupt level and again just as it is leaving, eliminating the processing time.

To capture **one-way delay measurements**, **NTP must be enabled** on both the source router and target router and their **clocks need to be synchronized** to the same clock source (with the ability to configure a clock tolerance for operations with microsecond precision). One-way jitter measurements do not require clock synchronization.

Configuration

R1

IOS

```
ip sla 2
 udp-jitter 2.2.2.2 4444
 timeout 2000
 frequency 30
ip sla schedule 2 life 300 start-time now
```

R2

IOS

```
ip sla responder
```

Verification

IOS

```
R1#sh ip sla statistics detail

Round Trip Time (RTT) for       Index 2
Type of operation: jitter
        Latest RTT: 22 ms
Latest operation start time: *13:41:02.579 UTC Sun Jan 26 2014
Latest operation return code: OK
RTT Values
        Number Of RTT: 10
        RTT Min/Avg/Max: 5/22/37 ms
Latency one-way time milliseconds
        Number of Latency one-way Samples: 0
        Source to Destination Latency one way Min/Avg/Max: 0/0/0 ms
        Destination to Source Latency one way Min/Avg/Max: 0/0/0 ms
        Source to Destination Latency one way Sum/Sum2: 0/0
        Destination to Source Latency one way Sum/Sum2: 0/0
Jitter time milliseconds
        Number of SD Jitter Samples: 9
        Number of DS Jitter Samples: 9
        Source to Destination Jitter Min/Avg/Max: 1/10/24 ms
        Destination to Source Jitter Min/Avg/Max: 0/5/20 ms
        Source to destination positive jitter Min/Avg/Max: 3/10/24 ms
        Source to destination positive jitter Number/Sum/Sum2: 3/31/601
        Source to destination negative jitter Min/Avg/Max: 1/9/16 ms
        Source to destination negative jitter Number/Sum/Sum2: 6/55/699
        Destination to Source positive jitter Min/Avg/Max: 1/6/20 ms
        Destination to Source positive jitter Number/Sum/Sum2: 4/25/411
        Destination to Source negative jitter Min/Avg/Max: 1/5/13 ms
        Destination to Source negative jitter Number/Sum/Sum2: 4/20/190
        Interarrival jitterout: 0       Interarrival jitterin: 0
        Over thresholds occurred: FALSE
Packet Loss Values
        Loss Source to Destination: 0           Loss Destination to
Source: 0
        Out Of Sequence: 0      Tail Drop: 0    Packet Late Arrival: 0
        Packet Skipped: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 0
        Mean Opinion Score (MOS): 0
Number of successes: 3
Number of failures: 0
Operation time to live: 214 sec
Operational state of entry: Active
Last time this entry was reset: *13:40:02.531 UTC Sun Jan 26 2014
```

IOS

```
R2#sh ip sla responder
IP SLAs Responder is: Enabled
Number of control message received: 4 Number of errors: 0
Recent sources:
        169.254.12.1 [13:41:32.131 UTC Sun Jan 26 2014]
        169.254.12.1 [13:41:02.159 UTC Sun Jan 26 2014]
        169.254.12.1 [13:40:32.143 UTC Sun Jan 26 2014]
        169.254.12.1 [13:40:02.123 UTC Sun Jan 26 2014]
Recent error sources:
```

IP SLA for MPLS VPN

No major difference exists, you just need to define the VRF to be used for connectivity. Also it's good practice to also define the source address of the operation.

Configuration

IOS

```
ip sla 3
 path-echo 10.0.0.2 source-ip 10.0.0.1
 vrf VPN
 frequency 30
ip sla schedule 3 life 300 start-time now
```

Verification

IOS

```
R1#sh ip sla statistics detail

Round Trip Time (RTT) for       Index 3
```

```
Type of operation: path-echo
        Latest RTT: 60 ms
Latest operation start time: *14:01:35.903 UTC Sun Jan 26 2014
Latest operation return code: OK
Over thresholds occurred: FALSE
Number of successes: 10
Number of failures: 0
Operation time to live: 0
Operational state of entry: Inactive
Last time this entry was reset: *13:57:05.899 UTC Sun Jan 26 2014
```

You can always use the following command to verify your IP SLA setup and view the default values of all parameters not explicitly configured:

IOS

```
R1#sh ip sla configuration 3
IP SLAs, Infrastructure Engine-II.

Entry number: 3
Owner:
Tag:
Type of operation to perform: path-echo
Target address/Source address: 10.0.0.2/10.0.0.1
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Type Of Service parameters: 0x0
Verify data: No
Loose Source Routing: Disabled
Vrf Name: VPN
LSR Path:
Schedule:
    Operation frequency (seconds): 30
    Next Scheduled Start Time: Start Time already passed
    Group Scheduled : FALSE
    Randomly Scheduled : FALSE
    Life (seconds): 300
    Entry Ageout (seconds): never
    Recurring (Starting Everyday): FALSE
    Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000
Distribution Statistics:
    Number of statistic hours kept: 2
    Number of statistic paths kept: 5
    Number of statistic hops kept: 16
    Number of statistic distribution buckets kept: 1
    Statistic distribution interval (milliseconds): 20
History Statistics:
    Number of history Lives kept: 0
    Number of history Buckets kept: 15
    Number of history Samples kept: 16
    History Filter Type: None
```

Netflow

Netflow can help in:

- network application and user monitoring
- network analysis and planning
- security analysis, accounting and billing
- traffic engineering
- data warehousing and data mining

Netflow key fields:

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Layer 3 protocol type
- Type of service (ToS)
- Input logical interface

Netflow versions

- v1
  - initial version
- v5
  - adds support for ASN and flow sequence numbers
- v7
  - special version for old C6k releases
- v8
  - adds support for aggregation caches
- v9
  - adds support for new fields and record types using templates
  - adds support for IPv6, multicast, MPLS and BGP next hop
- v10
  - aka IPFIX

v1, v5, v9 are the most common ones.

IPFIX (an IETF standard) is based on netflow v9.

IOS

```
interface FastEthernet0/0.34
 ip flow ingress
 ip flow egress
```

IOS

```
R3#sh ip cache flow
IP packet size distribution (20 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448
480
   .550 .200 .000 .250 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
.000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 4456704 bytes
  4 active, 65532 inactive, 14 added
  219 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 533256 bytes
  0 active, 16384 inactive, 0 added, 0 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec)
Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow
/Flow
TCP-BGP              1      0.0         2    49      0.0      11.7
15.0
UDP-other            8      0.0         1    28      0.0       0.0
15.3
ICMP                 1      0.0         5   100      0.0       8.0
15.8

SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP
Pkts
Total:              10      0.0         1    54      0.0       1.9
15.4

SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP
Pkts
Fa0/0.37      34.3.7.7        Fa0/0.34*     46.0.0.8        11 C013
829F     1
Fa0/0.37      34.3.7.7        Fa0/0.34*     46.0.0.8        11 C012
829E     1
Fa0/0.37      34.3.7.7        Fa0/0.34*     46.0.0.8        11 C011
829D     1
Fa0/0.34      169.254.34.4    Local         169.254.34.3    06 5415
00B3     2
```

When using netflow v9, you can include the BGP next-hop with either the peer-as or the origin-as.

IOS

```
ip flow-export version 9 origin-as bgp-nexthop
ip flow-export destination 34.0.0.7 3333
```

IOS

```
R3#sh ip flow export
Flow export v9 is enabled for main cache
  Export source and destination details :
  VRF ID : Default
    Destination(1)  34.0.0.7 (3333)
  Version 9 flow records, peer-as bgp-nexthop
  5 flows exported in 2 udp datagrams
  0 flows failed due to lack of export packet
  0 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues
  0 export packets were dropped due to fragmentation failures
  0 export packets were dropped due to encapsulation fixup failures
```

You can also define the duration of active/inactive flows before they are exported.

You need to find the right balance between short and large timeouts, taking into account the cache size and the cpu load.

IOS

```
ip flow-cache timeout inactive 30
ip flow-cache timeout active 10
```

Various options are also available for aggregation caches:

IOS

```
R3(config)#ip flow-aggregation cache ?
  as                        AS aggregation
  as-tos                    AS-TOS aggregation
  bgp-nexthop-tos           BGP nexthop TOS aggregation
  destination-prefix        Destination Prefix aggregation
  destination-prefix-tos    Destination Prefix TOS aggregation
  prefix                    Prefix aggregation
  prefix-port               Prefix-port aggregation
  prefix-tos                Prefix-TOS aggregation
  protocol-port             Protocol and port aggregation
  protocol-port-tos         Protocol, port and TOS aggregation
  source-prefix             Source Prefix aggregation
  source-prefix-tos         Source Prefix TOS aggregation
```

You can use **two export protocols**:

- UDP (default)
    - unreliable
    - not congestion aware
- SCTP
    - uses reliable, partly-reliable or no reliable transmission
    - implements congestion control mechanism

IOS

```
ip flow-export destination 2.2.2.2 2222
ip flow-export destination 3.3.3.3 3333 sctp
  backup destination 4.4.4.4 4444
```

SCTP is supported in IOS > 12.4(4)T.

MPLS egress netflow

It allows you to capture IP flow information for packets that arrive on a router as MPLS packets and that are transmitted as IP packets (i.e. PE=>CE direction).

IOS

```
interface FastEthernet0/0.17
 ip vrf forwarding ONE
 mpls netflow egress
```

IOS

```
R7#sh mpls forwarding-table vrf ONE 10.1.7.0 24 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
18     Aggregate   10.1.7.0/24[V]    0
        MAC/Encaps=0/0, MRU=0, Tag Stack{}
        VPN route: ONE
        Feature Quick flag set
    Per-packet load-sharing

R7#sh ip cache flow
IP packet size distribution (5 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448
480
   .000 .000 .000 1.00 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
.000
```

```
    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
  1 active, 4095 inactive, 1 added
  4 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 25800 bytes
  1 active, 1023 inactive, 1 added, 1 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec)
Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow
/Flow

SrcIf          SrcIPaddress    DstIf          DstIPaddress    Pr SrcP DstP
Pkts
Fa0/0.37       10.0.0.3        Fa0/0.17*      10.0.0.1        01 0000
0800     5
```

Netflow for Layer 2

You can also capture some extra fields that include L2 information, like below:

IOS

```
R3(config)#ip flow-capture ?
  fragment-offset  Capture the fragment offset
  icmp             Capture the ICMP type and code
  ip-id            Capture the IP id
  mac-addresses    Capture src and dst MAC addresses
  packet-length    Capture the max and min packet length
  ttl              Capture the TTL
  vlan-id          Capture the VLAN id
```

IOS

```
R3#sh ip cache verbose flow
IP packet size distribution (229 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448
480
   .427 .528 .000 .043 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
.000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
```

```
IP Flow Switching Cache, 4456704 bytes
  2 active, 65534 inactive, 158 added
  3870 ager polls, 0 flow alloc failures
  Active flows timeout in 10 minutes
  Inactive flows timeout in 30 seconds
IP Sub Flow Cache, 533256 bytes
  6 active, 16378 inactive, 47 added, 43 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec)
Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow
/Flow
TCP-BGP             57      0.0         2    49      0.0      13.4
25.2
UDP-other           98      0.0         1    28      0.0       0.0
15.7
ICMP                 1      0.0         5   100      0.0       8.0
15.8
Total:             156      0.0         1    41      0.0       4.9
19.1

SrcIf          SrcIPaddress    DstIf          DstIPaddress    Pr TOS
Flgs  Pkts
Port Msk AS                    Port Msk AS    NextHop              B/Pk
Active
BGP: BGP NextHop

SrcIf          SrcIPaddress    DstIf          DstIPaddress    Pr TOS
Flgs  Pkts
Port Msk AS                    Port Msk AS    NextHop              B/Pk
Active
BGP: BGP NextHop
Fa0/0.37       34.3.7.7        Fa0/0.34*      46.0.0.4        01 00
10       5
0000 /24 0                     0800 /32 248   169.254.34.4
100     7.9
BGP: 169.254.34.4
FFlags: 01
MAC: (VLAN id) c208.0618.0000  (037)          ca06.13bc.0000  (034)
ICMP type:       8                            ICMP code:        0

Fa0/0.34       169.254.34.4    Local          169.254.34.3    06 C0
18       2
5415 /32 0                     00B3 /32 0     0.0.0.0
49    17.9
BGP: 0.0.0.0
MAC: (VLAN id) ca06.13bc.0000  (034)          0000.0000.0000  (000)
```

Netflow for IPv6

You can also export IPv6 flows, like the IPv4 ones.

IOS

```
interface X
 ipv6 flow ingress
 ipv6 flow egress
!
ipv6 flow-export destination 5.5.5.5 5555
```

The same flow parameters apply to IPv6 as well.

An extra option is the ability to specify a minimum mask for prefixes, in order to define the detail of addresses.

Netflow for IPv6 is supported in IOS > 12.3.(7)T.

Netflow for multicast

- ingress
  - information about the source and how many times the traffic was replicated
  - packets that fail RPF check
- egress
  - information about the destination of the traffic flow

IOS

```
ip multicast netflow output-counters
ip multicast netflow rpf-failure
```

You also need to enable normal netflow under the relevant interfaces.

Links

- IETF - RFC 3954
- IETF - RFC 7011
- IETF - RFC 7012

PBB (802.1ah) or MAC-in-MAC

Ingress UNI & Tunnel configuration

IOS

```
interface X
 service instance 10 ethernet
  encapsulation dot1q 10
  bridge-domain 100 c-mac
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 200 c-mac
!
interface Y
 service instance 10 ethernet
  encapsulation dot1q 10
  bridge-domain 100 c-mac
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 200 c-mac
!
ethernet mac-tunnel virtual 1
 bridge-domain 1111
 mac tunnel address destination default 9999.9999.9999
 service instance 1 ethernet
  encapsulation dot1ah isid 1000
  bridge-domain 100 c-mac
 service instance 2 ethernet
  encapsulation dot1ah isid 2000
  bridge-domain 200 c-mac
```

Egress forwarding can be accomplished using one of the following methods:

L2 bridging with switchport

```
interface Z
 switchport
 switchport mode trunk
 switchport trunk allowed vlan 1111
```

L2 bridging with EVC

```
interface Z
 service instance 1 ethernet
  encapsulation dot1q 1111
  bridge-domain 1111
```

EoMPLS

```
interface vlan 1111
 xconnect 10.10.10.10 11 encapsulation mpls
```

VPLS

```
l2 vfi PBB-VFI manual
 vpn id 1111
 neighbor 20.20.20.20 22 encapsulation mpls
 neighbor 30.30.30.30 33 encapsulation mpls
!
interface vlan 1111
 xconnect vfi PBB-VFI
```

PBB is supported on 7600.