Cisco IOS XR MPLS Configuration Guide

Cisco IOS XR Software Release 3.7

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Customer Order Number: OL-15850-02

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco IOS XR MPLS Configuration Guide
© 2008 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface MPC-xiii

Changes to This Document MPC-xiii

Obtaining Documentation and Submitting a Service Request MPC-xiv

Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software MPC-1

Contents MPC-2

Prerequisites for Implementing Cisco MPLS LDP MPC-2

Information About Implementing Cisco MPLS LDP MPC-2

Overview of Label Distribution Protocol MPC-3

LDP Graceful Restart MPC-6

Label Advertisement Control (Outbound Filtering) MPC-10

Label Acceptance Control (Inbound Filtering) MPC-10

Local Label Allocation Control MPC-10

Session Protection MPC-11

IGP Synchronization MPC-11

IGP Auto-configuration MPC-12

LDP Nonstop Routing MPC-12

How to Implement LDP on Cisco IOS XR Software MPC-13

Configuring LDP Discovery Parameters MPC-13

Configuring LDP Discovery Over a Link MPC-15

Configuring LDP Discovery for Active Targeted Hellos MPC-17

Configuring LDP Discovery for Passive Targeted Hellos MPC-19

Configuring Label Advertisement Control (Outbound Filtering) MPC-21

Setting Up LDP Neighbors MPC-23

Setting Up LDP Forwarding MPC-25

Setting Up LDP NSF Using Graceful Restart MPC-27

Configuring Label Acceptance control (Inbound Filtering) MPC-30

Configuring Local Label Allocation Control MPC-31

Configuring Session Protection MPC-33

Configuring LDP IGP Synchronization: OSPF MPC-34

Configuring LDP IGP Synchronization: ISIS MPC-36

Configuring LDP IGP Sync Delay Interval MPC-38

Enabling LDP Auto-configuration for a Specified OSPF Instance MPC-39

Enabling LDP Auto-configuration in an Area for a Specified OSPF Instance MPC-41

Disabling LDP Auto-configuration **MPC-42**
Configuring LDP NSR **MPC-44**
Configuration Examples for Implementing LDP **MPC-45**
Configuring LDP with Graceful Restart: Example **MPC-46**
Configuring LDP Discovery: Example **MPC-46**
Configuring LDP Link: Example **MPC-46**
Configuring LDP Discovery for Targeted Hellos: Example **MPC-46**
Configuring Label Advertisement (Outbound Filtering): Example **MPC-47**
Configuring LDP Neighbors: Example **MPC-47**
Configuring LDP Forwarding: Example **MPC-47**
Configuring LDP Nonstop Forwarding with Graceful Restart: Example **MPC-48**
Configuring Label Acceptance (Inbound Filtering): Example **MPC-48**
Configuring Local Label Allocation Control: Example **MPC-48**
Configuring LDP Session Protection: Example **MPC-48**
Configuring LDP IGP Synchronization - OSPF: Example **MPC-48**
Configuring LDP IGP Synchronization - ISIS: Example **MPC-49**
Configuring LDP Auto-configuration: Example **MPC-49**
Additional References **MPC-49**
Related Documents **MPC-49**
Standards **MPC-50**
MIBs **MPC-50**
RFCs **MPC-50**
Technical Assistance **MPC-50**

Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software **MPC-51**
Contents **MPC-52**
Prerequisites for Implementing RSVP for MPLS-TE and MPLS O-UNI **MPC-52**
Information About Implementing RSVP for MPLS-TE and MPLS O-UNI **MPC-52**
Overview of RSVP for MPLS-TE and MPLS O-UNI **MPC-52**
LSP Setup **MPC-53**
High Availability **MPC-54**
Graceful Restart **MPC-54**
ACL-based Prefix Filtering **MPC-57**
Information About Implementing RSVP Authentication **MPC-57**
RSVP Authentication Functions **MPC-58**
RSVP Authentication Design **MPC-58**
Global, Interface, and Neighbor Authentication Modes **MPC-58**
Security Association **MPC-59**
Key-source Key-chain **MPC-60**

Guidelines for Window-Size and Out-of-Sequence Messages **MPC-61**
Caveats for Out-of-Sequence **MPC-61**
How to Implement RSVP **MPC-61**
Configuring Traffic Engineering Tunnel Bandwidth **MPC-61**
Confirming DiffServ-TE Bandwidth **MPC-62**
Configuring MPLS O-UNI Bandwidth **MPC-63**
Enabling Graceful Restart **MPC-63**
Configuring ACL-based Prefix Filtering **MPC-65**
Verifying RSVP Configuration **MPC-68**
How to Implement RSVP Authentication **MPC-71**
Configuring Global Configuration Mode RSVP Authentication **MPC-72**
Configuring an Interface for RSVP Authentication **MPC-76**
Configuring RSVP Neighbor Authentication **MPC-82**
Verifying the Details of the RSVP Authentication **MPC-88**
Eliminating Security Associations for RSVP Authentication **MPC-88**
Configuration Examples for RSVP **MPC-89**
Bandwidth Configuration (Prestandard): Example **MPC-89**
Bandwidth Configuration (MAM): Example **MPC-89**
Bandwidth Configuration (RDM): Example **MPC-89**
Refresh Reduction and Reliable Messaging Configuration: Example **MPC-89**
Configuring Graceful Restart: Example **MPC-90**
Configuring ACL-based Prefix Filtering: Example **MPC-91**
Setting DSCP for RSVP Packets: Example **MPC-91**
Configuration Examples for RSVP Authentication **MPC-92**
RSVP Authentication Global Configuration Mode: Example **MPC-92**
RSVP Authentication for an Interface: Example **MPC-92**
RSVP Neighbor Authentication: Example **MPC-92**
RSVP Authentication by Using All the Modes: Example **MPC-93**
Additional References **MPC-93**
Related Documents **MPC-93**
Standards **MPC-94**
MIBs **MPC-94**
RFCs **MPC-94**
Technical Assistance **MPC-94**

Implementing MPLS Forwarding on Cisco IOS XR Software **MPC-95**
MFI Control-Plane Services **MPC-95**
MFI Data-Plane Services **MPC-95**

Implementing MPLS Traffic Engineering on Cisco IOS XR Software **MPC-97**

Contents **MPC-98**

Prerequisites for Implementing Cisco MPLS Traffic Engineering **MPC-98**

Information About Implementing MPLS Traffic Engineering **MPC-98**

Overview of MPLS Traffic Engineering **MPC-99**

Protocol-Based CLI **MPC-100**

Differentiated Services Traffic Engineering **MPC-100**

Flooding **MPC-102**

Fast Reroute **MPC-103**

MPLS-TE and Fast Reroute over Link Bundles **MPC-104**

Ignore Intermediate System-to-Intermediate System Overload Bit Setting in MPLS-TE **MPC-104**

Generalized MPLS **MPC-105**

Flexible Name-based Tunnel Constraints **MPC-107**

MPLS Traffic Engineering Interarea Tunneling **MPC-108**

MPLS-TE Forwarding Adjacency **MPC-110**

Unequal Load Balancing **MPC-111**

Path Computation Element **MPC-112**

Policy-based Tunnel Selection **MPC-113**

How to Implement Traffic Engineering on Cisco IOS XR Software **MPC-115**

Building MPLS-TE Topology **MPC-115**

Creating an MPLS-TE Tunnel **MPC-119**

Configuring Forwarding over the MPLS-TE Tunnel **MPC-121**

Protecting MPLS Tunnels with Fast Reroute **MPC-123**

Configuring a Prestandard Diff-Serv TE Tunnel **MPC-127**

Configuring an IETF Diff-Serv TE Tunnel Using RDM **MPC-129**

Configuring an IETF Diff-Serv TE Tunnel Using MAM **MPC-131**

Configuring the Ignore Integrated Intermediate System-to-Intermediate System Overload Bit Setting in MPLS-TE **MPC-134**

Configuring GMPLS on Cisco IOS XR Software **MPC-135**

Configuring Flexible Name-based Tunnel Constraints **MPC-165**

Configuring IS-IS to Flood MPLS-TE Link Information **MPC-170**

Configuring an OSPF Area of MPLS-TE **MPC-172**

Configuring Explicit Paths with ABRs Configured as Loose Addresses **MPC-174**

Configuring MPLS-TE Forwarding Adjacency **MPC-175**

Configuring Unequal Load Balancing **MPC-177**

Configuring a Path Computation Client and Element **MPC-180**

Configuring Policy-based Tunnel Selection **MPC-185**

Configuration Examples for Cisco MPLS-TE **MPC-187**

Configuring Fast Reroute and SONET APS: Example **MPC-187**

Building MPLS-TE Topology and Tunnels: Example **MPC-188**
Configuring IETF Diff-Serv TE Tunnels: Example **MPC-189**
Configuring the Ignore IS-IS Overload Bit Setting in MPLS-TE: Example **MPC-189**
Configuring GMPLS: Example **MPC-189**
Configuring Flexible Name-based Tunnel Constraints: Example **MPC-191**
Configuring an Interarea Tunnel: Example **MPC-193**
Configuring Forwarding Adjacency: Example **MPC-193**
Configuring Unequal Load Balancing: Example **MPC-193**
Configuring PCE: Example **MPC-194**
Configure Policy-based Tunnel Selection: Example **MPC-195**
Additional References **MPC-196**
Related Documents **MPC-196**
Standards **MPC-196**
MIBs **MPC-196**
RFCs **MPC-197**
Technical Assistance **MPC-197**

Implementing MPLS Optical User Network Interface Protocol on Cisco IOS XR Software **MPC-199**
Contents **MPC-199**
Prerequisites for Implementing MPLS O-UNI **MPC-200**
Information About Implementing MPLS O-UNI **MPC-200**
How to Implement MPLS O-UNI on Cisco IOS XR Software **MPC-202**
Setting Up an MPLS O-UNI Connection **MPC-202**
Tearing Down an MPLS O-UNI Connection **MPC-205**
Verifying MPLS O-UNI Configuration **MPC-207**
Configuration Examples for MPLS O-UNI **MPC-210**
MPLS O-UNI Neighbor and Data Link Configuration: Examples **MPC-210**
O-UNI Connection Establishment: Example **MPC-211**
O-UNI Connection Tear-Down: Example **MPC-211**
Additional References **MPC-212**
Related Documents **MPC-212**
Standards **MPC-212**
MIBs **MPC-212**
RFCs **MPC-213**
Technical Assistance **MPC-213**

Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software **MPC-215**
Contents **MPC-216**

Prerequisites for Implementing MPLS L2VPN on Cisco IOS XR Software MPC-216
Information About Implementing L2VPN MPC-216
L2VPN Overview MPC-217
ATMoMPLS with L2VPN Capability MPC-217
Virtual Circuit Connection Verification on L2VPN MPC-218
Ethernet over MPLS MPC-218
Quality of Service MPC-222
High Availability MPC-223
Preferred Tunnel Path MPC-223
How to Implement L2VPN MPC-224
Configuring an Interface or Connection for L2VPN MPC-224
Configuring Static Point-to-Point Cross-Connects MPC-226
Configuring Dynamic Point-to-Point Cross-Connects MPC-228
Configuring Inter-AS MPC-230
Configuring L2VPN Quality of Service MPC-230
Configuring Preferred Tunnel Path MPC-234
Configuration Examples for L2VPN MPC-235
L2VPN Interface Configuration: Example MPC-236
Point-to-Point Cross-connect Configuration: Examples MPC-236
Inter-AS: Example MPC-236
L2VPN Quality of Service: Example MPC-238
Preferred Path: Example MPC-238
Additional References MPC-238
Related Documents MPC-238
Standards MPC-239
MIBs MPC-239
RFCs MPC-239
Technical Assistance MPC-240

Implementing Virtual Private LAN Services on Cisco IOS XR Software MPC-241

Contents MPC-241
Prerequisites for Implementing Virtual Private LAN Services on Cisco IOS XR Software MPC-242
Restrictions for Implementing Virtual Private LAN Services on Cisco IOS XR Software MPC-242
Information About Implementing Virtual Private LAN Services MPC-242
Virtual Private LAN Services Overview MPC-243
VPLS for an MPLS-based Provider Core MPC-243
Signaling MPC-244
Bridge Domain MPC-244

MAC Address-related Parameters **MPC-244**
LSP Ping over VPWS and VPLS **MPC-247**
How to Implement Virtual Private LAN Services **MPC-247**
Configuring a Bridge Domain **MPC-247**
Configuring a Layer 2 Virtual Forwarding Instance **MPC-257**
Configuring the MAC Address-related Parameters **MPC-269**
Configuration Examples for Virtual Private LAN Services **MPC-278**
Virtual Private LAN Services Configuration for Provider Edge-to-Provider Edge: Example **MPC-279**
Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example **MPC-280**
Additional References **MPC-280**
Related Documents **MPC-280**
Standards **MPC-281**
MIBs **MPC-281**
RFCs **MPC-281**
Technical Assistance **MPC-281**

Implementing IPv6 VPN Provider Edge Transport over MPLS on Cisco IOS XR Software **MPC-283**
Contents **MPC-283**
Prerequisites for Implementing 6PE **MPC-284**
Information About 6PE **MPC-284**
Overview of 6PE **MPC-284**
Benefits of 6PE **MPC-284**
Deploying IPv6 over MPLS Backbones **MPC-285**
IPv6 on the Provider Edge and Customer Edge Routers **MPC-285**
IPv6 Provider Edge Multipath **MPC-286**
How to Implement 6PE **MPC-286**
Configuring 6PE **MPC-286**
Configuration Examples for 6PE **MPC-288**
Configuring 6PE on a PE Router: Example **MPC-288**
Additional References **MPC-289**
Related Document **MPC-289**
Standards **MPC-289**
MIBs **MPC-290**
RFCs **MPC-290**
Technical Assistance **MPC-290**

Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software **MPC-291**
Contents **MPC-291**
Prerequisites for Layer 2 Tunnel Protocol Version 3 **MPC-291**

Information About Layer 2 Tunnel Protocol Version 3 **MPC-292**
L2TPv3 Operation **MPC-292**
L2TPv3 Benefits **MPC-293**
L2TPv3 Features **MPC-293**
How to Implement Layer 2 Tunnel Protocol Version 3 **MPC-296**
Configuring a Pseudowire Class **MPC-297**
Configuring L2TP Control-Channel Parameters **MPC-298**
Configuring L2TPv3 Pseudowires **MPC-309**
Configuring the Cross-connect Attachment Circuit **MPC-315**
Configuration Examples for Layer 2 Tunnel Protocol Version 3 **MPC-318**
Configuring an L2TP Class for L2TPv3-based L2VPN PE Routers: Example **MPC-318**
Configuring a Pseudowire Class: Example **MPC-319**
Configuring L2TPv3 Control Channel Parameters: Example **MPC-319**
Configuring the Cross-Connect Group: Example **MPC-319**
Configuring an Interface for Layer 2 Transport Mode: Example **MPC-319**
Additional References **MPC-320**
Related Documents **MPC-320**
Standards **MPC-320**
MIBs **MPC-320**
RFCs **MPC-320**
Technical Assistance **MPC-321**

Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software **MPC-323**

Contents **MPC-323**
Prerequisites for Configuring MPLS VPNs over IP Tunnels **MPC-324**
Restrictions for Configuring MPLS VPNs over IP Tunnels **MPC-324**
Information About MPLS VPNs over IP Tunnels **MPC-324**
Overview: MPLS VPNs over IP Tunnels **MPC-324**
Advertising Tunnel Type and Tunnel Capabilities Between PE Routers—BGP **MPC-325**
PE Routers and Address Space **MPC-325**
Packet Validation Mechanism **MPC-326**
Quality of Service Using the Modular QoS CLI **MPC-326**
BGP Multipath Load Sharing for MPLS VPNs over IP Tunnels **MPC-326**
Inter-AS and CSC Support over IP Tunnels **MPC-327**
How to Configure MPLS VPNs over IP Tunnels **MPC-327**
Configuring the Global VRF Definition **MPC-327**
Configuring a Route-Policy Definition **MPC-329**
Configuring a Static Route **MPC-330**

Configuring an IPv4 Loopback Interface MPC-331
Configuring a CFI VRF Interface MPC-333
Configuring the Core Network MPC-334
Configuring Inter-AS and CSC Support over IP Tunnels MPC-335
Verifying MPLS VPN over IP MPC-342
Configuration Examples for MPLS VPNs over IP Tunnels MPC-342
Configuring an L2TPv3 Tunnel: Example MPC-342
Configuring the Global VRF Definition: Example MPC-342
Configuring a Route-Policy Definition: Example MPC-343
Configuring a Static Route: Example MPC-343
Configuring an IPv4 Loopback Interface: Example MPC-343
Configuring a CFI VRF Interface: Example MPC-343
Additional References MPC-344
Related Documents MPC-344
Standards MPC-344
MIBs MPC-344
RFCs MPC-345
Technical Assistance MPC-345

Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software MPC-347

Contents MPC-348
MPLS L3VPN Prerequisites MPC-348
Information About MPLS Layer 3 VPNs on Cisco IOS XR Software MPC-349
MPLS L3VPN Overview MPC-349
MPLS L3VPN Benefits MPC-350
MPLS L3VPN Restrictions MPC-350
How MPLS L3VPN Works MPC-351
MPLS L3VPN Major Components MPC-353
Inter-AS Support for L3VPN MPC-354
Inter-AS Restrictions MPC-354
Inter-AS Support: Overview MPC-354
Inter-AS and ASBRs MPC-355
Transmitting Information Between Autonomous Systems MPC-356
Exchanging VPN Routing Information MPC-357
Packet Forwarding MPC-359
Confederations MPC-362
MPLS VPN Inter-AS BGP Label Distribution MPC-364
Exchanging IPv4 Routes with MPLS labels MPC-364
Carrier Supporting Carrier Support for L3VPN MPC-366

CSC Prerequisites **MPC-367**
CSC Benefits **MPC-367**
Configuration Options for the Backbone and Customer Carriers **MPC-367**
IPv6 VPN Provider Edge (6VPE) Support **MPC-369**
6PVE Benefits **MPC-369**
6VPE Network Architecture **MPC-369**
Dual Stack **MPC-370**
6VPE Operation **MPC-370**
How to Implement MPLS Layer 3 VPNs on Cisco IOS XR Software **MPC-371**
Configuring the Core Network **MPC-371**
Connecting MPLS VPN Customers **MPC-374**
Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels **MPC-394**
Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses **MPC-403**
Configuring Carrier Supporting Carrier **MPC-412**
Verifying the MPLS Layer 3 VPN Configuration **MPC-421**
Configuring 6VPE Support **MPC-424**
Configuring an IPv6 Address Family Under VRF **MPC-425**
Configuring BGP Route Distinguisher and Core-facing Sessions **MPC-426**
Configuring a PE-CE Protocol **MPC-428**
Configuration Examples for Implementing MPLS Layer 3 VPNs **MPC-430**
Configuring an MPLS VPN Using BGP: Example **MPC-431**
Configuring the Routing Information Protocol on the PE Router: Example **MPC-432**
Configuring the PE Router Using EIGRP: Example **MPC-432**
Configuration Examples for MPLS VPN CSC **MPC-432**
Configuration Examples for 6VPE **MPC-434**
Additional References **MPC-440**
Related Documents **MPC-440**
Standards **MPC-440**
MIBs **MPC-441**
RFCs **MPC-441**
Technical Assistance **MPC-441**

Index

Preface

The *Cisco IOS XR MPLS Configuration Guide* preface contains the following sections:

- Changes to This Document, page MPC-xiii
- Obtaining Documentation and Submitting a Service Request, page MPC-xiv

Changes to This Document

Table 1 lists the technical changes made to this document since it was first printed.

Table 1 Changes to This Document

Revision	Date	Change Summary
OL-15850-02	December 2008	The Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software module was modified as follows: • Added the **interface** command to the static point-to-point cross-connects procedure. See "Configuring Static Point-to-Point Cross-Connects" section on page 226. • Modified the static and dynamic point-to-point cross-connects configuration examples. See "Point-to-Point Cross-connect Configuration: Examples" section on page 236.
		The Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software was modified as follows: • Modified the listed L2TPv3 features. See "L2TPv3 Features" section on page 293. • Added a new Local Switching figure. See "Local Switching" section on page 294. • Modified the pseudowire class configuration example. See "Configuring a Pseudowire Class: Example" section on page 319.

Table 1 Changes to This Document (continued)

Revision	Date	Change Summary
OL-15850-02 (continued)	December 2008	• Added the following L2TPv3 configuration examples: – L2TPv3 control channel parameters. See "Configuring L2TPv3 Control Channel Parameters: Example" section on page 319. – Cross-Connect group. "Configuring the Cross-Connect Group: Example" section on page 319. – Interface configuration for Layer 2 transport mode. Configuring an Interface for Layer 2 Transport Mode: Example, page MPC-319.
OL-15850-01	June 2008	Initial release of this document.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software

Multiprotocol Label Switching (MPLS) is a standards-based solution driven by the Internet Engineering Task Force (IETF) that was devised to convert the Internet and IP backbones from best-effort networks into business-class transport mediums.

MPLS, with its label switching capabilities, eliminates the need for an IP route look-up and creates a virtual circuit (VC) switching function, allowing enterprises the same performance on their IP-based network services as with those delivered over traditional networks such as Frame Relay or ATM.

Label Distribution Protocol (LDP) performs label distribution in MPLS environments. LDP provides the following capabilities:

- LDP performs hop-by-hop or dynamic path setup; it does not provide end-to-end switching services.
- LDP assigns labels to routes using the underlying Interior Gateway Protocols (IGP) routing protocols.
- LDP provides constraint-based routing using LDP extensions for traffic engineering.

Finally, LDP is deployed in the core of the network and is one of the key protocols used in MPLS-based Layer 2 and Layer 3 Virtual Private Networks (VPNs).

Feature History for Implementing MPLS LDP on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router. Support was added for conceptual and configuration information about LDP Label Advertisement Control (Outbound label filtering).
Release 3.3.0	Support was added for • Inbound Label Filtering • Local Label Allocation Control • Session Protection • LDP-IGP Synchronization
Release 3.4.0	No modification.
Release 3.4.1	No modification.

Release 3.5.0	Support was added for LDP Auto-configuration.
Release 3.6.0	Support was added for LDP nonstop routing (NSR).
Release 3.7.0	No modification.

Contents

- Prerequisites for Implementing Cisco MPLS LDP, page MPC-2
- Information About Implementing Cisco MPLS LDP, page MPC-2
- How to Implement LDP on Cisco IOS XR Software, page MPC-13
- Configuration Examples for Implementing LDP, page MPC-45
- Additional References, page MPC-49

Prerequisites for Implementing Cisco MPLS LDP

The following prerequisites are required to implement MPLS LDP:

- You must be in a user group associated with a task group that includes the proper task IDs for MPLS LDP commands.
- You must be running Cisco IOS XR software.
- You must install a composite mini-image and the MPLS package.
- You must activate IGP.

Information About Implementing Cisco MPLS LDP

To implement MPLS LDP, you should understand the following concepts:

- Overview of Label Distribution Protocol, page MPC-3
- LDP Graceful Restart, page MPC-6
- Label Advertisement Control (Outbound Filtering), page MPC-10
- Label Acceptance Control (Inbound Filtering), page MPC-10
- Local Label Allocation Control, page MPC-10
- Session Protection, page MPC-11
- IGP Synchronization, page MPC-11
- IGP Auto-configuration, page MPC-12
- LDP Nonstop Routing, page MPC-12

Overview of Label Distribution Protocol

LDP performs label distribution in MPLS environments. LDP uses hop-by-hop or dynamic path setup, but does not provide end-to-end switching services. Labels are assigned to routes that are chosen by the underlying IGP routing protocols. The Label Switched Paths (LSPs) that result from the routes, forward labeled traffic across the MPLS backbone to adjacent nodes.

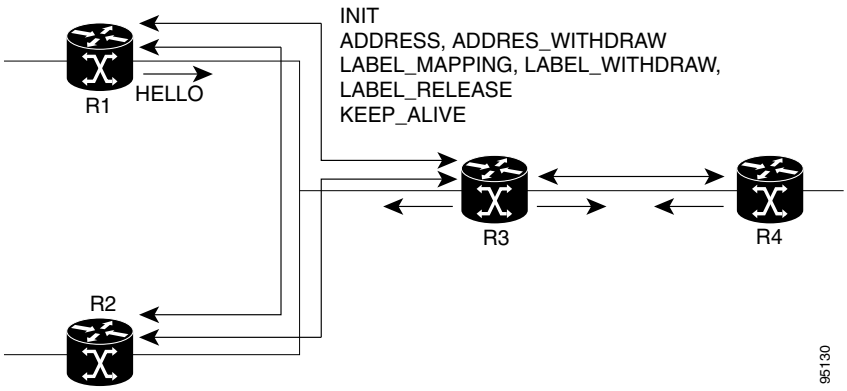
Label Switched Paths

LSPs are created in the network through MPLS. They can be created statically, by RSVP traffic engineering (TE) or by LDP. LSPs created by LDP perform hop-by-hop path setup instead of an end-to-end path.

LDP Control Plane

The control plane enables label switched routers (LSRs) to discover their potential peer routers and to establish LDP sessions with those peers to exchange label binding information. Figure 1 shows the control messages exchanged between LDP peers.

Figure 1 LDP Control Protocol



LDP uses the hello discovery mechanism to discover its neighbor or peer on the network. When LDP is enabled on an interface, it sends hello messages to a link-local multicast address, and joins a specific multicast group to receive hellos from other LSRs present on the given link. When LSRs on a given link receive hellos, their neighbors are discovered and the LDP session (using TCP) is established.

Note Hellos are not only used to discover and trigger LDP sessions; they are also required to maintain LDP sessions. If a certain number of hellos from a given peer are missed in sequence, LDP sessions are brought down, until the peer is discovered again.

LDP also supports non-link neighbors that could be multiple hops away on the network, using the targeted hello mechanism. In these cases, hellos are sent on a directed, unicast address.

The first message in the session establishment phase is the initialization message, which is used to negotiate session parameters. After session establishment, LDP sends a list of all its interface addresses to its peers in an address message. Whenever a new address becomes available or unavailable, the peers are notified regarding such changes via ADDRESS or ADDRESS_WITHDRAW messages respectively.

When MPLS LDP learns an IGP prefix it allocates a label locally as the inbound label. The local binding between the prefix label is conveyed to its peers via LABEL_MAPPING message. If the binding breaks and becomes unavailable, a LABEL_WITHDRAW message is sent to all its peers, which respond with LABEL_RELEASE messages.

The local label binding and remote label binding received from its peer(s) is used to setup forwarding entries. Using routing information from the IGP protocol and the forwarding information base (FIB), the next active hop is selected. Label binding is learned from the next hop peer, and is used as the outbound label while setting up the forwarding plane.

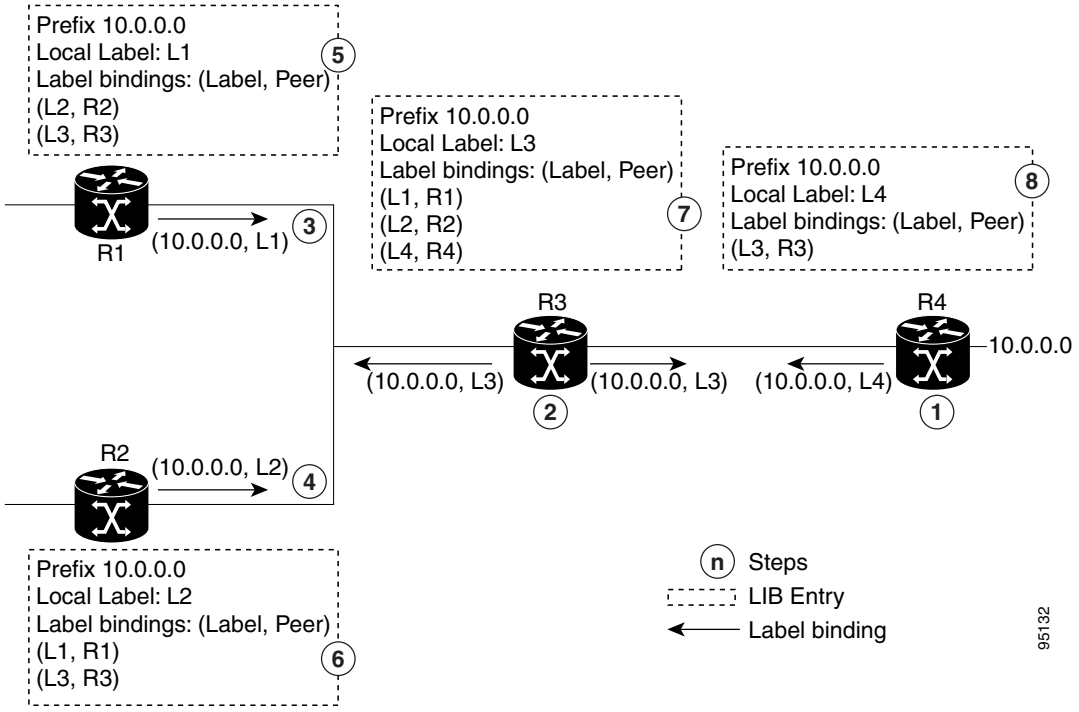
The LDP session is also kept alive using the LDP keepalive mechanism, where an LSR sends a keepalive message periodically to its peers. If no messages are received and a certain number of keepalive messages are missed from a peer, the session is declared dead, and brought down immediately.

Exchanging Label Bindings

LDP creates LSPs to perform the hop-by-hop path setup so that MPLS packets can be transferred between the nodes on the MPLS network.

Figure 2 illustrates the process of label binding exchange for setting up LSPs.

Figure 2 Setting Up Label Switched Paths



For a given network (10.0.0.0), hop-by-hop LSPs are set up between each of the adjacent routers (or, nodes) and each node allocates a local label and passes it to its neighbor as a binding:

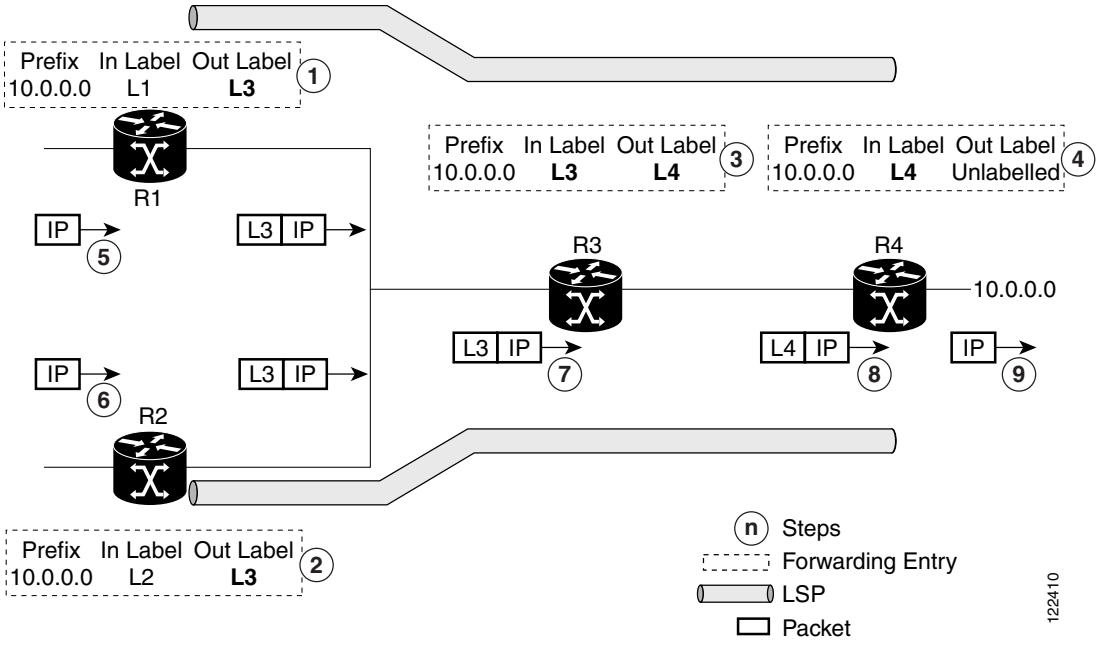
1. R4 allocates local label L4 for prefix 10.0.0.0 and advertises it to its neighbors (R3).
2. R3 allocates local label L3 for prefix 10.0.0.0 and advertises it to its neighbors (R1, R2, R4).
3. R1 allocates local label L1 for prefix 10.0.0.0 and advertises it to its neighbors (R2, R3).
4. R2 allocates local label L2 for prefix 10.0.0.0 and advertises it to its neighbors (R1, R3).

5. R1's Label Information Base (LIB) keeps local and remote labels bindings from its neighbors.
6. R2's LIB keeps local and remote labels bindings from its neighbors.
7. R3's LIB keeps local and remote labels bindings from its neighbors.
8. R4's LIB keeps local and remote labels bindings from its neighbors.

Setting Up LDP Forwarding

Once label bindings are learned, the LDP control plane is ready to setup the MPLS forwarding plane as shown in Figure 3.

Figure 3 Forwarding Setup



1. Because R3 is next hop for 10.0.0.0 as notified by the forwarding information base (FIB), R1 selects label binding from R3 and installs forwarding entry (L1, L3).
2. Because R3 is next hop for 10.0.0.0 (as notified by FIB), R2 selects label binding from R3 and installs forwarding entry (L2, L3).
3. Because R4 is next hop for 10.0.0.0 (as notified by FIB), R3 selects label binding from R4 and installs forwarding entry (L3, L4).
4. Because next hop for 10.0.0.0 (as notified by FIB) is beyond R4, R4 uses NO-LABEL as the outbound and installs the forwarding entry (L4); the outbound packet is forwarded IP-only.
5. Incoming IP traffic on ingress LSR R1 gets label-imposed and is forwarded as an MPLS packet with label L3.
6. Incoming IP traffic on ingress LSR R2 gets label-imposed and is forwarded as an MPLS packet with label L3.
7. R3 receives an MPLS packet with label L3, looks up in the MPLS label forwarding table and switches this packet as an MPLS packet with label L4.

8. R4 receives an MPLS packet with label L4, looks up in the MPLS label forwarding table and finds that it should be Unlabeled, pops the top label, and passes it to the IP forwarding plane.
9. IP forwarding takes over and forwards the packet onward.

LDP Graceful Restart

LDP graceful restart, provides a control plane mechanism to ensure high availability, allows detection and recovery from failure conditions while preserving Nonstop Forwarding (NSF) services. Graceful restart is a way to recover from signaling and control plane failures without impacting forwarding.

Without LDP graceful restart, when an established session fails, the corresponding forwarding states are cleaned immediately from the restarting and peer nodes. In this case LDP forwarding will have to restart from the beginning, causing a potential loss of data and connectivity.

The LDP graceful restart capability is negotiated between two peers during session initialization time, in FT SESSION TLV. In this typed length value (TLV), each peer advertises the following information to its peers:

- Reconnect time: the maximum time that other peer will wait for this LSR to reconnect after control channel failure.
- Recovery time: Max time that other peer has on its side to reinstate or refresh its states with this LSR. This time is used only during session reestablishment after earlier session failure.
- FT flag: This flag indicates whether a restart could restore the preserved (local) node state.

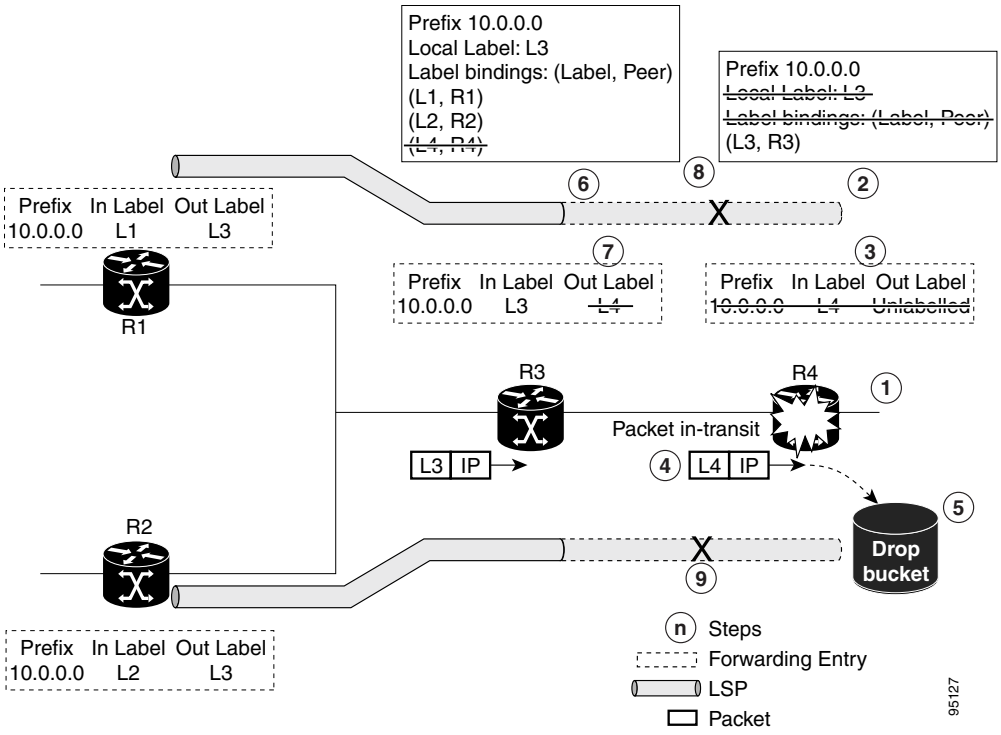
Once the graceful restart session parameters are conveyed and the session is up and running, graceful restart procedures are activated.

Control Plane Failure

When a control plane failure occurs, connectivity can be affected. The forwarding states installed by the router control planes are lost, and the in-transit packets could be dropped, thus breaking NSF.

Figure 4 illustrates a control plane failure and shows the process and results of a control plane failure leading to loss of connectivity.

Figure 4 Control Plane Failure



1. The R4 LSR control plane restarts.
2. LIB is lost when the control plane restarts.
3. The forwarding states installed by the R4 LDP control plane are immediately deleted.
4. Any in-transit packets flowing from R3 to R4 (still labelled with L4) arrive at R4.
5. The MPLS forwarding plane at R4 performs a lookup on local label L4 which fails. Because of this failure, the packet is dropped and NSF is not met.
6. The R3 LDP peer detects the failure of the control plane channel and deletes its label bindings from R4.
7. The R3 control plane stops using outgoing labels from R4 and deletes the corresponding forwarding state (rewrites), which in turn causes forwarding disruption.
8. The established LSPs connected to R4 are terminated at R3, resulting in broken end-to-end LSPs from R1 to R4.
9. The established LSPs connected to R4 are terminated at R3, resulting in broken LSPs end-to-end from R2 to R4.

Phases in Graceful Restart

The graceful restart mechanism can be divided into different phases as follows:

- Control communication failure detection
- Forwarding state maintenance during failure
- Control state recovery

Control Communication Failure Detection

Control communication failure is detected when the system detects either:

- Missed LDP hello discovery messages
- Missed LDP keepalive protocol messages
- Detection of Transmission Control Protocol (TCP) disconnection a with a peer

Forwarding State Maintenance During Failure

Persistent forwarding states at each LSR are achieved through persistent storage (checkpoint) by the LDP control plane. While the control plane is in the process of recovering, the forwarding plane keeps the forwarding states, but marks them as stale. Similarly, the peer control plane also keeps (and marks as stale) the installed forwarding rewrites associated with the node that is restarting. The combination of local node forwarding and remote node forwarding plane states ensures NSF and no disruption in the traffic.

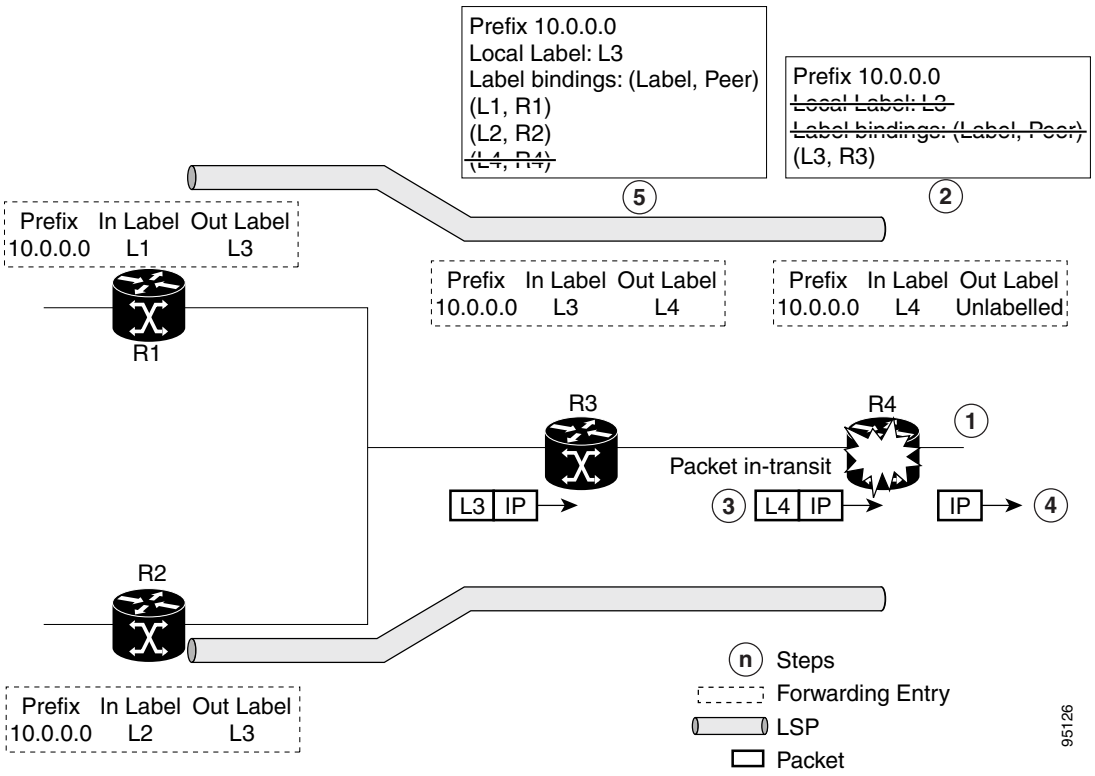
Control State Recovery

Recovery occurs when the session is reestablished and label bindings are exchanged again. This process allows the peer nodes to synchronize and to refresh stale forwarding states.

Recovery with Graceful-Restart

Figure 5 illustrates the process of failure recovery using graceful restart.

Figure 5 Recovering with Graceful Restart



1. The router R4 LSR control plane restarts.
2. With the control plane restart, LIB is gone but forwarding states installed by R4's LDP control plane are not immediately deleted but are marked as stale.
3. Any in-transit packets from R3 to R4 (still labelled with L4) arrive at R4.
4. The MPLS forwarding plane at R4 performs a successful lookup for the local label L4 as forwarding is still intact. The packet is forwarded accordingly.
5. The router R3 LDP peer detects the failure of the control plane and channel and deletes the label bindings from R4. The peer, however, does not delete the corresponding forwarding states but marks them as stale.
6. At this point there are no forwarding disruptions.
7. The peer also starts the neighbor reconnect timer using the reconnect time value.
8. The established LSPs going toward the router R4 are still intact, and there are no broken LSPs.

When the LDP control plane recovers, the restarting LSR starts its forwarding state hold timer and restores its forwarding state from the checkpointed data. This action reinstates the forwarding state and entries and marks them as old.

The restarting LSR reconnects to its peer, indicating in the FT Session TLV, that it either was or was not able to restore its state successfully. If it was able to restore the state, the bindings are resynchronized.

The peer LSR stops the neighbor reconnect timer (started by the restarting LSR), when the restarting peer connects and starts the neighbor recovery timer. The peer LSR checks the FT Session TLV if the restarting peer was able to restore its state successfully. It reinstates the corresponding forwarding state entries and receives binding from the restarting peer. When the recovery timer expires, any forwarding state that is still marked as stale is deleted.

If the restarting LSR fails to recover (restart), the restarting LSR forwarding state and entries will eventually timeout and is deleted, while neighbor-related forwarding states or entries are removed by the Peer LSR on expiration of the reconnect or recovery timers.

Label Advertisement Control (Outbound Filtering)

By default, LDP advertises labels for all the prefixes to all its neighbors. When this is not desirable (for scalability and security reasons), you can configure LDP to perform outbound filtering for local label advertisement for one or more prefixes to one more peers. This feature is known as *LDP outbound label filtering*, or *local label advertisement control.*

Label Acceptance Control (Inbound Filtering)

By default, LDP accepts labels (as remote bindings) for all prefixes from all peers. LDP operates in liberal label retention mode, which instructs LDP to keep remote bindings from all peers for a given prefix. For security reasons, or to conserve memory, you can override this behavior by configuring label binding acceptance for set of prefixes from a given peer.

The ability to filter remote bindings for a defined set of prefixes is also referred to as *LDP inbound label filtering*.

Note Inbound filtering can also be implemented using an outbound filtering policy; however, you may not be able to implement this system if an LDP peer resides under a different administration domain. When both inbound and outbound filtering options are available, we recommend that you use outbound label filtering.

Local Label Allocation Control

By default, LDP allocates local labels for all prefixes that are not Border Gateway Protocol (BGP) prefixes[1]. This is acceptable when LDP is used for applications other than Layer 3 virtual private networks (L3VPN) core transport. When LDP is used to set up transport LSPs for L3VPN traffic in the core, it is not efficient or even necessary to allocate and advertise local labels for, potentially, thousands of IGP prefixes. In such a case, LDP is typically required to allocate and advertise local label for loopback /32 addresses for PE routers. This is accomplished using LDP local label allocation control, where an access list can be used to limit allocation of local labels to a set of prefixes. Limiting local label allocation provides several benefits, including reduced memory usage requirements, fewer local forwarding updates, and fewer network and peer updates.

Tip You can configure label allocation using an IP access list to specify a set of prefixes that local labels will allocate and advertise.

1. For L3VPN Inter-AS option C, LDP may also be required to assign local labels for some BGP prefixes.

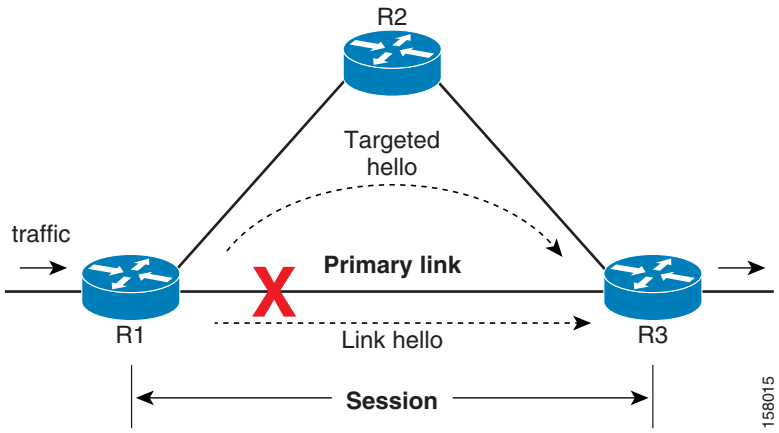
Session Protection

When a link comes up, IP converges earlier and much faster than MPLS LDP and may result in MPLS traffic loss until MPLS convergence. If a link flaps, the LDP session will also flap due to loss of link discovery. LDP session protection minimizes traffic loss and provides faster convergence and protects existing LDP (link) sessions by means of "parallel" source of targeted discovery/hello. An LDP session is kept alive and neighbor label bindings are maintained when links are down. Upon reestablishment of primary link adjacencies, MPLS convergence is expedited as LDP need not relearn the neighbor label bindings.

LDP session protection lets you configure LDP to automatically protect sessions with all or a given set of peers (as specified by peer-acl). When configured, LDP initiates backup targeted hellos automatically for neighbors for which primary link adjacencies already exist. These backup targeted hellos maintain LDP sessions when primary link adjacencies go down.

Figure 6 illustrates LDP session protection between neighbors R1 and R3. The primary link adjacency between R1 and R3 is directly connected link and the backup; targeted adjacency is maintained between R1 and R3. If the direct link fails, LDP link adjacency is destroyed, but the session is kept up and running using targeted hello adjacency (through R2). When the direct link comes back up, there is no change in the LDP session state and LDP can converge quickly and begin forwarding MPLS traffic.

Figure 6 Session Protection



Note When LDP session protection is activated (upon link failure), protection is maintained for an unlimited period time.

IGP Synchronization

Lack of synchronization between LDP and IGP can cause MPLS traffic loss. Upon link up, for example, IGP can advertise and use a link before LDP convergence has occurred; or, a link may continue to be used in IGP after an LDP session goes down.

LDP IGP synchronization synchronizes LDP and IGP so that IGP advertises links with regular metrics only when MPLS LDP is converged on that link. LDP considers a link converged when at least one LDP session is up and running on the link for which LDP has sent its applicable label bindings and received at least one label binding from the peer. LDP communicates this information to IGP upon link up or session down events and IGP acts accordingly, depending on sync state.

In the event of an LDP graceful restart session disconnect, a session is treated as converged as long as the graceful restart neighbor is timed out. Additionally, upon local LDP restart, a checkpointed recovered LDP graceful restart session is used and treated as converged and is given an opportunity to connect and re-synchronize.

Under certain circumstances, it might be required to delay declaration of re-synchronization to a configurable interval. LDP provides a configuration option to delay declaring synchronization up for up to 60 seconds. LDP communicates this information to IGP upon linkup or session down events.

Note The configuration for LDP IGP synchronization resides in respective IGPs (OSPF and IS-IS) and there is no LDP-specific configuration for enabling of this feature. However, there is a specific LDP configuration for IGP sync delay timer.

IGP Auto-configuration

To enable LDP on a large number of interfaces, IGP auto-configuration lets you automatically configure LDP on all interfaces associated with a specified IGP interface; for example, when LDP is used for transport in the core network. However, there needs to be one IGP set up to enable LDP auto-configuration.

Typically, LDP assigns and advertises labels for IGP routes and must often be enabled on all active interfaces by an IGP. Without IGP auto-configuration, you must define the set of interfaces under LDP, a procedure that is time-intensive and error-prone.

Note LDP auto-configuration is supported for IPv4 unicast family in the default VRF. The IGP is responsible for verifying and applying the configuration.

You can also disable auto-configuration on a per-interface basis. This permits LDP to enable all IGP interfaces except those that are explicitly disabled and prevents LDP from enabling an interface when LDP auto-configuration is configured under IGP.

LDP Nonstop Routing

LDP nonstop routing (NSR) functionality makes failures, such as route processor (RP) or distributed route processor (DRP) failover, invisible to routing peers with minimal to no disruption of convergence performance. By default, NSR is globally enabled on all LDP sessions except AToM.

A disruption in service may include any of the following events:

- Route processor (RP) or distributed route processor (DRP) failover
- LDP process restart
- In-service system upgrade (ISSU)
- Minimum disruption restart (MDR)

Note Unlike graceful restart functionality, LDP NSR does not require protocol extensions and does not force software upgrades on other routers in the network, nor does LDP NSR require peer routers to support NSR.

Process failures of active TCP or LDP results in session loss and, as a result, NSR cannot be provided unless RP switchover is configured as a recovery action. For more information about how to configure switchover as a recovery action for NSR, see "Configuring Transports on Cisco IOS XR Software" in *Cisco IOS XR IP Addresses and Services Configuration Guide*.

How to Implement LDP on Cisco IOS XR Software

A typical MPLS LDP deployment requires coordination among several global neighbor routers. Various configuration tasks are required to implement MPLS LDP on Cisco IOS XR software, as follows:

- Configuring LDP Discovery Parameters, page MPC-13 (optional)
- Configuring LDP Discovery Over a Link, page MPC-15 (required)
- Configuring LDP Discovery for Active Targeted Hellos, page MPC-17 (required)
- Configuring LDP Discovery for Passive Targeted Hellos, page MPC-19 (required)
- Configuring Label Advertisement Control (Outbound Filtering), page MPC-21 (optional)
- Setting Up LDP Neighbors, page MPC-23 (optional)
- Setting Up LDP Forwarding, page MPC-25 (optional)
- Setting Up LDP NSF Using Graceful Restart, page MPC-27 (optional)
- Configuring Label Acceptance control (Inbound Filtering), page MPC-30 (optional)
- Configuring Local Label Allocation Control, page MPC-31 (optional)
- Configuring Session Protection, page MPC-33 (optional)
- Configuring LDP IGP Synchronization: OSPF, page MPC-34 (optional)
- Configuring LDP IGP Synchronization: ISIS, page MPC-36 (optional)
- Configuring LDP IGP Sync Delay Interval, page MPC-38 (optional)
- Enabling LDP Auto-configuration for a Specified OSPF Instance, page MPC-39 (optional)
- Enabling LDP Auto-configuration in an Area for a Specified OSPF Instance, page MPC-41
- Disabling LDP Auto-configuration, page MPC-42 (optional)
- Configuring LDP NSR, page MPC-44

Configuring LDP Discovery Parameters

Perform this task to configure LDP discovery parameters (which may be crucial for LDP operations).

Note The LDP discovery mechanism is used to discover or locate neighbor nodes.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** {*type number* | *ip-address*}
4. **discovery** {**hello** | **targeted-hello**} **holdtime** *seconds*
5. **discovery** {**hello** | **targeted-hello**} **interval** *seconds*
6. **end**
   or
   commit
7. **show mpls ldp parameters**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration submode.
Step 3	**router-id** {*type number*	*ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-ldp)# router-id loopback 1`	Specifies the router ID of the local node. • In Cisco IOS XR software, the router ID is specified as an interface name or IP address. By default, LDP uses the global router ID (configured by the global router ID process).

Step 4	**discovery** {**hello**	**targeted-hello**} **holdtime** *seconds*
		
		Example:
		`RP/0/RP0/CPU0:router(config-ldp)# discovery hello holdtime 30`
		`RP/0/RP0/CPU0:router(config-ldp)# discovery targeted-hello holdtime 180`
			Specifies the time that a discovered neighbor is kept without receipt of any subsequent hello messages.
			• The default value for the seconds argument is 15 seconds for link hello and 90 seconds for targeted hello messages.

Step 5	**discovery** {**hello**	**targeted-hello**} **interval** *seconds*
		
		Example:
		`RP/0/RP0/CPU0:router(config-ldp)# discovery hello interval 15`
		`RP/0/RP0/CPU0:router(config-ldp)# discovery targeted-hello interval 20`
			Selects the period of time between the transmission of consecutive hello messages.
			• The default value for the seconds argument is 5 seconds for link hello messages and 10 seconds for targeted hello messages.

	Command or Action	Purpose

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 7	**show mpls ldp parameters** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp parameters`	(Optional) Displays all the current MPLS LDP parameters.

Configuring LDP Discovery Over a Link

Perform this task to configure LDP discovery over a link.

Note There is no need to enable LDP globally.

Prerequisites

A stable router ID is required at either end of the link to ensure the link discovery (and session setup) is successful. If you do not assign a router ID to the routers, the system will default to the global router ID. Default router IDs are subject to change and may cause an unstable discovery.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** {*type number* | *ip-address*}
4. **interface** *type number*

5. **end**
   or
   commit
6. **show mpls ldp discovery**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration mode.
Step 3	**router-id** {*type number* | *ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-ldp)# router-id loopback 1`	(Optional) Specifies the router ID of the local node. • In Cisco IOS XR, the router ID is specified as an interface name or IP address. By default, LDP uses the global router ID (configured by the global router ID process).
Step 4	**interface** *type number* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# interface tunnel-te 12001`	Enters interface configuration mode for the LDP protocol. Interface type must be Tunnel-TE.

	Command or Action	Purpose

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**show mpls ldp discovery** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp discovery`	(Optional) Displays the status of the LDP discovery process. • This command, without an interface filter, generates a list of interfaces over which the LDP discovery process is running. The output information contains the state of the link (xmt/rcv hellos), local LDP identifier, the discovered peer's LDP identifier, and holdtime values.

Configuring LDP Discovery for Active Targeted Hellos

Perform this task to configure LDP discovery for active targeted hellos.

Note The active side for targeted hellos initiates the unicast hello toward a specific destination.

Prerequisites

The following prerequisites are required to configure LDP discovery for active targeted hellos:

- A stable router ID is required at either end of the targeted session. If you do not assign a router ID to the routers, the system will default to the global router ID. Please note that default router IDs are subject to change and may cause an unstable discovery.
- One or more MPLS Traffic Engineering tunnels are established between non-directly connected LSRs.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** {*type number* | *ip-address*}
4. **interface** *type number*
5. **end**
   or
   commit
6. **show mpls ldp discovery**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration submode.
Step 3	**router-id** [*type number* | *ip-address*] **Example:** `RP/0/RP0/CPU0:router(config-ldp)# router-id loopback 1`	Specifies the router ID of the local node. In Cisco IOS XR, the router ID is specified as an interface name or IP address. By default, LDP uses the global router ID (configured by global router ID process).
Step 4	**interface** *type number* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# interface tunnel-te 12001`	Enters interface configuration mode for the LDP protocol.

	Command or Action	Purpose

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**show mpls ldp discovery** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp discovery`	(Optional) Displays the status of the LDP discovery process. • This command, without an interface filter, generates a list of interfaces over which the LDP discovery process is running. The output information contains the state of the link (xmt/rcv hellos), local LDP identifier, the discovered peer's LDP identifier, and holdtime values.

Configuring LDP Discovery for Passive Targeted Hellos

Perform this task to configure LDP discovery for passive targeted hellos.

A passive side for targeted hello is the destination router (tunnel tail), which passively waits for an incoming hello message. Because targeted hellos are unicast, the passive side waits for an incoming hello message to respond with hello toward its discovered neighbor.

Prerequisites

A stable router ID is required at either end of the link to ensure that the link discovery (and session setup) is successful. If you do not assign a router ID to the routers, the system will default to the global router ID. Default router IDs are subject to change and may cause an unstable discovery.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** [*type number* | *ip-address*]
4. **discovery targeted-hello accept**

5. **end**
   or
   commit
6. **show mpls ldp discovery**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration mode.
Step 3	**router-id** [*type number* | *ip-address*] **Example:** `RP/0/RP0/CPU0:router(config-ldp)# router-id loopback 1`	(Optional) Specifies the router ID of the local node. • In Cisco IOS XR, the router ID is specified as an interface name or IP address. By default, LDP uses the global router ID (configured by global router ID process).
Step 4	**discovery targeted-hello accept** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# discovery targeted-hello accept`	Directs the system to accept targeted hello messages from any source and activates passive mode on the LSR for targeted hello acceptance. • This command is executed on the tail-end node (with respect to a given MPLS TE tunnel). • You can control the targeted-hello acceptance using the **discovery targeted-hello accept** command.

	Command or Action	Purpose

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**show mpls ldp discovery** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp discovery`	(Optional) Displays the status of the LDP discovery process. • This command, without an interface filter, generates a list of interfaces over which the LDP discovery process is running. The output information contains the state of the link (xmt/rcv hellos), local LDP identifier, the discovered peer's LDP identifier, and holdtime values.

Configuring Label Advertisement Control (Outbound Filtering)

Perform this task to configure label advertisement (outbound filtering).

By default, a label switched router (LSR) advertises all incoming label prefixes to each neighboring router. You can control the exchange of label binding information using the **mpls ldp label advertise** command. Using the optional keywords, you can *advertise selective prefixes to all neighbors,* advertise selective prefixes to defined neighbors, or **disable label advertisement to all peers for all prefixes.**

Note Prefixes and peers advertised selectively are defined in the access list.

Prerequisites

Before configuring label advertisement, enable LDP and configure an access list.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**

3. **label advertise** {**disable** | **for** *prefix-acl* [**to** *peer-acl*] | **interface** *interface-id*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration mode.
Step 3	**label advertise** {**disable** | **for** *prefix-acl* [**to** *peer-acl*] | **interface** *interface-id*} **Example:** `RP/0/RP0/CPU0:router(config-ldp)# label advertise interface POS 0/1/0/0` **`RP/0/RP0/CPU0:router(config-ldp)# for pfx_acl1 to peer_acl1`**	Configures label advertisement as specified by one of the following arguments: • **disable**—Disables label advertisement to all peers for all prefixes (if there are no other conflicting rules). • **interface**—Specifies an interface for label advertisement of an interface address. • **for** *aclist* **to** *peer-acl*—Specifies neighbors that advertise and receive label advertisements.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Setting Up LDP Neighbors

Perform this task to set up LDP neighbors.

Prerequisites

A stable router ID is required at either end of the link to ensure the link discovery (and session setup) is successful. If you do not assign a router ID to the routers, the system will default to the global router ID. Default router IDs are subject to change and may cause an unstable discovery.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **interface** *type number*
4. **discovery transport-address** [*ip-address* | **interface**]
5. **end**
   or
   commit
6. **holdtime** *seconds*
7. **neighbor** *ip-address* **password** [*encryption*] *password*
8. **backoff** *initial maximum*
9. **end**
   or
   commit
10. **show mpls ldp neighbor**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration submode.
Step 3	**interface** *type number* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# interface POS 0/1/0/0`	Enters interface configuration mode for the LDP protocol.

	Command or Action	Purpose

Step 4	**discovery transport-address** [*ip-address* | **interface**]
	
	Example:
	`RP/0/RP0/CPU0:router(onfig-ldp-if)# discovery transport-address 192.168.1.42`
	or
	`RP/0/RP0/CPU0:router(onfig-ldp)# discovery transport-address interface`
		Provides an alternative transport address for a TCP connection.
		• The default transport address advertised by an LSR (for TCP connections) to its peer is the router ID.
		• The transport address configuration is applied for a given LDP-enabled interface.
		• If the interface version of the command is used, the configured IP address of the interface is passed to its neighbors as the transport address.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**holdtime** *seconds* **Example:** `RP/0/RP0/CPU0:router(onfig-ldp)# holdtime 30`	Changes the time for which an LDP session is maintained in the absence of LDP messages from the peer. • The outgoing keepalive interval is adjusted accordingly (to make 3 keepalives in given holdtime) with a change in session holdtime value. • The session holdtime is also exchanged when the session is established. • In this example holdtime is set to 30 seconds, which causes the peer session to timeout in 30 seconds, as well as transmitting outgoing keepalive messages toward the peer every 10 seconds.
Step 7	**neighbor** *ip-address* **password** [*encryption*] *password* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# neighbor 192.168.2.44 password secretpasswd`	Configures password authentication (using the TCP MD5 option) for a given neighbor.

	Command or Action	Purpose
Step 8	**backoff** *initial maximum* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# backoff 10 20`	Configures the parameters for the LDP backoff mechanism. • The LDP backoff mechanism prevents two incompatibly configured LSRs from engaging in an unthrottled sequence of session setup failures. If a session setup attempt fails due to such incompatibility, each LSR delays its next attempt (backs off), increasing the delay exponentially with each successive failure until the maximum backoff delay is reached.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show mpls ldp neighbor** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp neighbor`	(Optional) Displays the status of the LDP session with its neighbors. • This command can be run with various filters as well as with the brief option.

Setting Up LDP Forwarding

Perform this task to set up LDP forwarding.

By default, the LDP control plane implements the penultimate hop popping (PHOP) mechanism. The PHOP mechanism requires that label switched routers use the implicit-null label as a local label for the given Forwarding Equivalence Class (FEC) for which LSR is the penultimate hop. Although PHOP has certain advantages, it may be required to extend LSP up to the ultimate hop under certain circumstances (for example, to propagate MPL QoS). This is done using a special local label (explicit-null) advertised to the peers after which the peers use this label when forwarding traffic toward the ultimate hop (egress LSR).

Prerequisites

A stable router ID is required at either end of the link to ensure the link discovery (and session setup) is successful. If you do not assign a router ID to the routers, the system will default to the global router ID. Default router IDs are subject to change and may cause an unstable discovery.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **explicit-null**
4. **end**
   or
   commit
5. **show mpls ldp forwarding**
6. **show mpls forwarding**
7. **ping** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration submode.
Step 3	**explicit-null** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# explicit-null`	Causes a router to advertise an explicit null label in situations where it normally advertises an implicit null label (for example, to enable an ultimate-hop disposition instead of PHOP).

	Command or Action	Purpose

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show mpls ldp forwarding** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp forwarding`	(Optional) Displays the MPLS LDP view of installed forwarding states (rewrites).
Step 6	**show mpls forwarding** **Example:** `RP/0/RP0/CPU0:router# show mpls forwarding`	(Optional) Displays a global view of all MPLS installed forwarding states (rewrites) by various applications (LDP, TE, and static).
Step 7	**ping** *ip-address* **Example:** `RP/0/RP0/CPU0:router# ping 192.168.2.55`	(Optional) Checks for connectivity to a particular IP address (going through MPLS LSP as shown in the **show mpls forwarding** command).

Setting Up LDP NSF Using Graceful Restart

Perform this task to set up NSF using LDP graceful restart.

LDP graceful restart is a way to enable NSF for LDP. The correct way to set up NSF using LDP graceful restart is to bring up LDP neighbors (link or targeted) with additional configuration related to graceful restart.

Prerequisites

A stable router ID is required at either end of the link to ensure the link discovery (and session setup) is successful. If you do not assign a router ID to the routers, the system will default to the global router ID. Default router IDs are subject to change and may cause an unstable discovery.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **interface** {*type number*}
4. **end**
   or
   commit
5. **graceful-restart**
6. **graceful-restart forwarding-state-holdtime** *seconds*
7. **graceful-restart reconnect-timeout** *seconds*
8. **end**
   or
   commit
9. **show mpls ldp parameters**
10. **show mpls ldp neighbor**
11. **show mpls ldp graceful-restart**

Note Repeat these steps on neighboring routers.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters MPLS LDP configuration mode.
Step 3	**interface** *type number* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# interface POS 0/1/0/0`	Enters interface configuration mode for the LDP protocol.

	Command or Action	Purpose

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**graceful-restart** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# graceful-restart`	Enables the LDP graceful restart feature.
Step 6	**graceful-restart forwarding-state-holdtime** *seconds* **Example:** `RP/0/RP0/CPU0:router(onfig-ldp)# graceful-restart forwarding state-holdtime 180`	(Optional) Specifies the length of time that forwarding can keep LDP-installed forwarding states and rewrites, and specifies when the LDP control plane restarts. • After restart of the control plane, when the forwarding state holdtime expires, any previously installed LDP forwarding state or rewrite that is not yet refreshed is deleted from the forwarding. • The recovery time sent after restart is computed as the current remaining value of the forwarding state hold timer.
Step 7	**graceful-restart reconnect-timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ldp)# graceful-restart reconnect timeout 169`	(Optional) The length of time a neighbor waits before restarting the node to reconnect before declaring an earlier graceful restart session as down. • This command is used to start a timer on the peer (upon a neighbor restart). This timer is referred to as Neighbor Liveness timer.

	Command or Action	Purpose

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**show mpls ldp parameters** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp parameters`	(Optional) Displays all the current MPLS LDP parameters.
Step 10	**show mpls ldp neighbor** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp neighbor`	(Optional) Displays the status of the LDP session with its neighbors. • This command can be run with various filters as well as with the brief option.
Step 11	**show mpls ldp graceful-restart** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp graceful-restart`	(Optional) Displays the status of the LDP graceful restart feature. • The output of this command not only shows states of different graceful restart timers, but also a list of graceful restart neighbors, their state, and reconnect count.

Configuring Label Acceptance control (Inbound Filtering)

Perform this task to configure LDP inbound label filtering.

Note By default, there is no inbound label filtering performed by LDP and thus an LSR accepts (and retains) all remote label bindings from all peers.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**

3. **label accept for** *prefix-acl* **from** *ip-address*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters the MPLS LDP configuration mode.

Step 3	**label accept for** ***prefix-acl*** **from** ***ip-address***
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# label accept for pfx_acl_1 from 192.168.1.1`
	`RP/0/RP0/CPU0:router(config-ldp-lbl-acpt)# label accept for pfx_acl_2 from 192.168.2.2`
		Configures inbound label acceptance for prefixes specified by prefix-acl from neighbor (as specified by its IP address).

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Local Label Allocation Control

Perform this task to configure label allocation control.

Note By default, local label allocation control is disabled and all non-BGP prefixes are assigned local labels.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **label allocate for** *prefix-acl*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters the MPLS LDP configuration mode.

	Command or Action	Purpose
Step 3	**label allocate for** ***prefix-acl*** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# label allocate for pfx_acl_1`	Configures label allocation control for prefixes as specified by prefix-acl.

Step 4	**end**
	 or
	 commit
	
	 Example:
	 `RP/0/RP0/CPU0:router(config-ldp)# end`
	 or
	 `RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		 • When you issue the **end** command, the system prompts you to commit changes:
		 `Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		 – Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		 – Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		 – Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		 • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Session Protection

Perform this task to configure LDP session protection.

By default, there is no protection is done for link sessions by means of targeted hellos.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **session protection** [**for** *peer-acl*] [**duration** *seconds*]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters the MPLS LDP configuration mode.
Step 3	**session protection for *peer-acl* duration *seconds*** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# session protection for peer_acl_1 duration 60`	Configures LDP session protection for peers specified by peer-acl with a maximum duration in seconds.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Configuring LDP IGP Synchronization: OSPF

Perform this task to configure LDP IGP Synchronization under OSPF.

Note By default, there is no synchronization between LDP and IGPs.

SUMMARY STEPS

1. **configure**
2. **router ospf** *interface-id*
3. **mpls ldp sync**
   or
   area *area-id* **mpls ldp sync**
   or
   area *area-id* **interface** *name* **mpls ldp sync**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 100`	Enters OSPF configuration mode.

	Command or Action	Purpose
Step 3	**mpls ldp sync** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls ldp sync`	Enables LDP IGP synchronization on an interface.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Configuring LDP IGP Synchronization: ISIS

Perform this task to configure LDP IGP Synchronization under ISIS.

Note By default, there is no synchronization between LDP and ISIS.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id* **interface** *name* **address-family ipv4 unicast**
3. **mpls ldp sync** [**level 1- 2**]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router isis** *instance-id* **interface** *name* **address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config)# router isis 100 interface POS 0/2/0/0 address-family ipv4 unicast`	Enters ISIS interface configuration mode for the IPv4 unicast address family.
Step 3	**mpls ldp sync** **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# mpls ldp sync`	Enables LDP IGP synchronization.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-if-af)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Configuring LDP IGP Sync Delay Interval

Perform this task to configure the LDP IGP synchronization delay interval.

By default, LDP does not delay declaring sync up as soon as convergence conditions are met.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **igp sync delay** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters Global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters the MPLS LDP configuration mode.

	Command or Action	Purpose
Step 3	**igp sync delay** ***seconds*** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# igp sync delay 30`	Configures LDP IGP sync delay in seconds.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Enabling LDP Auto-configuration for a Specified OSPF Instance

Perform this task to enable IGP auto-configuration globally for a specified OSPF process name.

You can disable auto-configuration on a per-interface basis. This lets LDP enable all IGP interfaces except those that are explicitly disabled (see Disabling LDP Auto-configuration, page MPC-42).

Note This feature is supported for IPv4 unicast family in default VRF only.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **mpls ldp auto-config**
4. **area** *area-id*
5. **interface** *type interface-id*
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf`	Enters a uniquely identifiable OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.
Step 3	**mpls ldp auto-config** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls ldp auto-config`	Enables LDP auto-configuration.
Step 4	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 8`	Configures an OSPF area and identifier. The *area-id* argument is specified as either a decimal value or an IP address.
Step 5	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface pos 0/6/0/0`	Enables LDP auto-configuration on the specified interface. **Note** LDP configurable limit for maximum number of interfaces does not apply to IGP auto-configuration interfaces.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling LDP Auto-configuration in an Area for a Specified OSPF Instance

Perform this task to enable IGP auto-configuration in a defined area with a specified OSPF process name.

You can disable auto-configuration on a per-interface basis. This lets LDP enable all IGP interfaces except those that are explicitly disabled (see, Disabling LDP Auto-configuration, page MPC-42).

Note This feature is supported for IPv4 unicast family in default VRF only.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process name*
3. **area** *area-id*
4. **mpls ldp auto-config**
5. **interface** *type interface-id*
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf`	Enters a uniquely identifiable OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.
Step 3	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# router ospf`	Configures an OSPF area and identifier. The *area-id* argument is specified as either a decimal value or an IP address.
Step 4	**mpls ldp auto-config** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls ldp auto-config`	Enables LDP auto-configuration.

	Command or Action	Purpose
Step 5	**interface** *type interface-id* **Example:Example:** `RP/0/RP0/CPU0:router(config-ospf)# interface pos 0/6/0/0`	Enables LDP auto-configuration on the specified interface. **Note** LDP configurable limit for maximum number of interfaces does not apply to IGP auto-config interfaces.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling LDP Auto-configuration

Perform this task to disable IGP auto-configuration.

You can disable auto-configuration on a per-interface basis. This lets LDP enable all IGP interfaces except those that are explicitly disabled.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **interface** *type interface-id*
4. **igp auto-config disable**
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters the MPLS LDP configuration mode.
Step 3	**interface** *type interface-id* **Example:Example:** `RP/0/RP0/CPU0:router(config-ldp)# interface pos 0/6/0/0`	Enters interface configuration mode and configures an interface.
Step 4	**igp auto-config disable** **Example:** `RP/0/RP0/CPU0:router(config-ldp-if)# igp auto-config disable`	Disables auto-configuration on the specified interface.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring LDP NSR

Perform this task to configure LDP NSR (see LDP Nonstop Routing, page MPC-12).

Note By default, NSR is globally-enabled on all LDP sessions except AToM.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **nsr**
4. **end**
   or
   commit
5. **show mpls ldp nsr statistics**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RP0/CPU0:router(config)# mpls ldp`	Enters the MPLS LDP configuration mode.
Step 3	**nsr** **Example:Example:** `RP/0/RP0/CPU0:router(config-ldp)# nsr`	Enables LDP nonstop routing.

	Command or Action	Purpose

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ldp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ldp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show mpls ldp nsr statistics** **Example:** `RP/0/RP0/CPU0:router(config-ldp)# igp auto-config disable`	Shows MPLS LDP NSR statistics.

Configuration Examples for Implementing LDP

This section provides the following configuration examples:

- Configuring LDP with Graceful Restart: Example, page MPC-46
- Configuring LDP Discovery: Example, page MPC-46
- Configuring LDP Link: Example, page MPC-46
- Configuring LDP Discovery for Targeted Hellos: Example, page MPC-46
- Configuring Label Advertisement (Outbound Filtering): Example, page MPC-47
- Configuring LDP Neighbors: Example, page MPC-47
- Configuring LDP Forwarding: Example, page MPC-47
- Configuring LDP Nonstop Forwarding with Graceful Restart: Example, page MPC-48
- Configuring Label Acceptance (Inbound Filtering): Example, page MPC-48
- Configuring Local Label Allocation Control: Example, page MPC-48
- Configuring LDP Session Protection: Example, page MPC-48
- Configuring LDP IGP Synchronization - OSPF: Example, page MPC-48
- Configuring LDP IGP Synchronization - ISIS: Example, page MPC-49
- Configuring LDP Auto-configuration: Example, page MPC-49

Configuring LDP with Graceful Restart: Example

The following example shows how to enable LDP with graceful restart on the POS interface 0/2/0/0:

```
mpls ldp
 graceful-restart
 interface pos0/2/0/0
!
```

Configuring LDP Discovery: Example

The following example shows how to configure LDP discovery parameters:

```
mpls ldp
 router-id loopback0
 discovery hello holdtime 15
 discovery hello interval 5
!

show mpls ldp parameters
show mpls ldp discovery
```

Configuring LDP Link: Example

The following example shows how to configure LDP link parameters:

```
mpls ldp
 interface pos 0/1/0/0
 !
!

show mpls ldp discovery
```

Configuring LDP Discovery for Targeted Hellos: Example

The following example shows how to configure LDP Discovery to accept targeted hello messages:

Active (tunnel head)

```
mpls ldp
 router-id loopback0
 interface tunnel-te 12001
 !
!
```

Passive (tunnel tail)

```
mpls ldp
 router-id loopback0
 discovery targeted-hello accept
!
```

Configuring Label Advertisement (Outbound Filtering): Example

The following example shows how to configure LDP label advertisement control:

```
mpls ldp
    label
        advertise
            disable
            for pfx_acl_1 to peer_acl_1
            for pfx_acl_2 to peer_acl_2
            for pfx_acl_3
            interface POS 0/1/0/0
            interface POS 0/2/0/0
        !
    !
!
ipv4 access-list pfx_acl_1
    10 permit ip host 1.0.0.0 any
!
ipv4 access-list pfx_acl_2
    10 permit ip host 2.0.0.0 any
!
ipv4 access-list peer_acl_1
    10 permit ip host 1.1.1.1 any
    20 permit ip host 1.1.1.2 any
!
ipv4 access-list peer_acl_2
    10 permit ip host 2.2.2.2 any
!

show mpls ldp binding
```

Configuring LDP Neighbors: Example

The following example shows how to disable label advertisement:

```
mpls ldp
    router-id Loopback0
    neighbor 1.1.1.1 password encrypted 110A1016141E
    neighbor 2.2.2.2 implicit-withdraw
!
```

Configuring LDP Forwarding: Example

The following example shows how to configure LDP forwarding:

```
mpls ldp
 explicit-null
!

show mpls ldp forwarding
show mpls forwarding
```

Configuring LDP Nonstop Forwarding with Graceful Restart: Example

The following example shows how to configure LDP nonstop forwarding with graceful restart:

```
mpls ldp
log
graceful-restart
!
 graceful-restart
 graceful-restart forwarding state-holdtime 180
 graceful-restart reconnect-timeout 15
 interface pos0/1/0/0
!

show mpls ldp graceful-restart
show mpls ldp neighbor gr
show mpls ldp forwarding
show mpls forwarding
```

Configuring Label Acceptance (Inbound Filtering): Example

The following example shows how to configure inbound label filtering:

```
mpls ldp
 label
accept
 for pfx_acl_2 from 192.168.2.2
!
 !
!
```

Configuring Local Label Allocation Control: Example

The following example shows how to configure local label allocation control:

```
mpls ldp
 label
allocate for pfx_acl_1
 !
!
```

Configuring LDP Session Protection: Example

The following example shows how to configure session protection:

```
mpls ldp
 session protection for peer_acl_1 duration 60
!
```

Configuring LDP IGP Synchronization - OSPF: Example

The following example shows how to configure LDP IGP synchronization:

```
router ospf 100
mpls ldp sync
!
```

```
mpls ldp
 igp sync delay 30
!
```

Configuring LDP IGP Synchronization - ISIS: Example

The following example shows how to configure LDP IGP synchronization:

```
router isis 100
 interface POS 0/2/0/0
address-family ipv4 unicast
mpls ldp sync
!
 !
!
mpls ldp
 igp sync delay 30
!
```

Configuring LDP Auto-configuration: Example

The following example shows how to configure the IGP auto-configuration feature globally for a specific OSPF interface ID:

```
router ospf 100
 mpls ldp auto-config
 area 0
  interface pos 1/1/1/1
```

The following example shows how to configure the IGP auto-configuration feature on a given area for a given OSPF interface ID:

```
router ospf 100
 area 0
  mpls ldp auto-config
  interface interface pos 1/1/1/1
```

Additional References

For additional information related to Implementing MPLS Label Distribution Protocol, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR LDP commands	*MPLS Label Distribution Protocol Commands on Cisco IOS XR Software*
Cisco CRS-1 router getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs[1]	Title
RFC 3031	*Multiprotocol Label Switching Architecture*
RFC 3036	*LDP Specification*
RFC 3037	LDP Applicability
RFC 3478	Graceful Restart Mechanism for Label Distribution Protocol
RFC3815	Definitions of Managed Objects for MPLS LDP

1. Not all supported RFCs are listed.

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software

Multiprotocol Label Switching (MPLS) is a standards-based solution, driven by the Internet Engineering Task Force (IETF), devised to convert the Internet and IP backbones from best-effort networks into business-class transport media.

Resource Reservation Protocol (RSVP) is a signaling protocol that enables systems to request resource reservations from the network. RSVP processes protocol messages from other systems, processes resource requests from local clients, and generates protocol messages. As a result, resources are reserved for data flows on behalf of local and remote clients. RSVP creates, maintains, and deletes these resource reservations.

RSVP provides a secure method to control quality-of-service (QoS) access to a network.

MPLS Traffic Engineering (MPLS-TE) and MPLS Optical User Network Interface (MPLS O-UNI) use RSVP to signal label switched paths (LSPs).

Feature History for Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router. Support was added for ACL-based prefix filtering.
Release 3.3.0	No modification.
Release 3.4.0	No modification.
Release 3.4.1	Support was added for RSVP authentication.
Release 3.5.0	No modification.
Release 3.6.0	No modification.
Release 3.7.0	No modification.

Contents

- Prerequisites for Implementing RSVP for MPLS-TE and MPLS O-UNI, page MPC-52
- Information About Implementing RSVP for MPLS-TE and MPLS O-UNI, page MPC-52
- Information About Implementing RSVP Authentication, page MPC-57
- How to Implement RSVP, page MPC-61
- How to Implement RSVP Authentication, page MPC-71
- Configuration Examples for RSVP, page MPC-89
- Configuration Examples for RSVP Authentication, page MPC-92
- Additional References, page MPC-93

Prerequisites for Implementing RSVP for MPLS-TE and MPLS O-UNI

The following are prerequisites are required to implement RSVP for MPLS-TE and MPLS O-UNI:

- You must be in a user group associated with a task group that includes the proper task IDs for MPLS RSVP commands.
- Either a composite mini-image plus an MPLS package, or a full image, must be installed.

Information About Implementing RSVP for MPLS-TE and MPLS O-UNI

To implement MPLS RSVP, you must understand the following concepts, which are described in the sections that follow:

- Overview of RSVP for MPLS-TE and MPLS O-UNI, page MPC-52
- LSP Setup, page MPC-53
- High Availability, page MPC-54
- Graceful Restart, page MPC-54
- ACL-based Prefix Filtering, page MPC-57

For information on how to implement RSVP authentication, see How to Implement RSVP Authentication, page MPC-71.

Overview of RSVP for MPLS-TE and MPLS O-UNI

RSVP is a network control protocol that enables Internet applications to signal LSPs for MPLS-TE, and LSPs for O-UNI. The RSVP implementation is compliant with the IETF RFC 2205, RFC 3209, and OIF2000.125.7.

When configuring an O-UNI LSP, the RSVP session is bidirectional. The exchange of data between a pair of machines actually constitutes a single RSVP session. The RSVP session is established using an Out-Of-Band (OOB) IP Control Channel (IPCC) with the neighbor. The RSVP messages are transported over an interface other than the data interface.

RSVP supports extensions according to OIF2000.125.7 requirements, including:

- Generalized Label Request
- Generalized UNI Attribute
- UNI Session
- New Error Spec sub-codes

RSVP is automatically enabled on interfaces on which MPLS-TE is configured. For MPLS-TE LSPs with non-zero bandwidth, the RSVP bandwidth has to be configured on the interfaces. There is no need to configure RSVP, if all MPLS-TE LSPs have zero bandwidth. For O-UNI, there is no need for any RSVP configuration.

RSVP Refresh Reduction, defined in RFC2961, includes support for reliable messages and summary refresh messages. Reliable messages are retransmitted rapidly if the message is lost. Because each summary refresh message contains information to refresh multiple states, this greatly reduces the amount of messaging needed to refresh states. For refresh reduction to be used between two routers, it must be enabled on both routers. Refresh Reduction is enabled by default.

Message rate limiting for RSVP allows you to set a maximum threshold on the rate at which RSVP messages are sent on an interface. Message rate limiting is disabled by default.

The process that implements RSVP is restartable. A software upgrade, process placement or process failure of RSVP or any of its collaborators, has been designed to ensure Nonstop Forwarding (NSF) of the data plane.

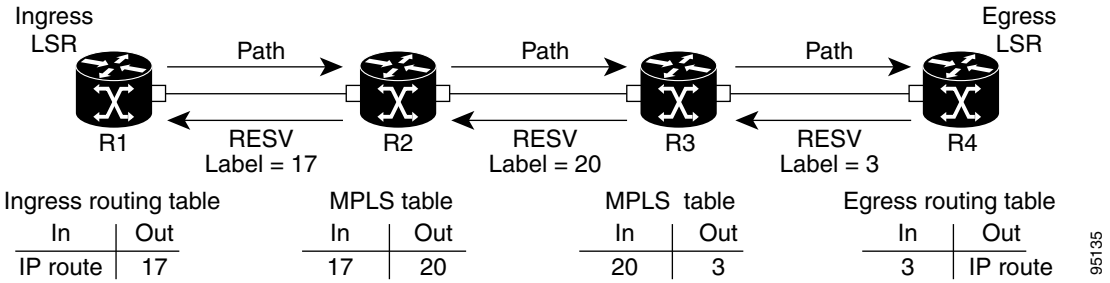
RSVP supports graceful restart, which is compliant with RFC 3473. It follows the procedures that apply when the node reestablishes communication with the neighbor's control plane within a configured restart time.

It is important to note that RSVP is not a routing protocol. RSVP works in conjunction with routing protocols and installs the equivalent of dynamic access lists along the routes that routing protocols calculate. Because of this, implementing RSVP in an existing network does not require migration to a new routing protocol.

LSP Setup

LSP setup is initiated when the LSP head node sends path messages to the tail node (see Figure 7).

Figure 7 RSVP Operation



The Path messages reserve resources along the path to each node, creating Path soft states on each node. When the tail node receives a path message, it sends a reservation (RESV) message with a label back to the previous node. When the reservation message arrives at the previous node, it causes the reserved resources to be locked and forwarding entries are programmed with the MPLS label sent from the tail-end node. A new MPLS label is allocated and sent to the next node upstream.

When the reservation message reaches the head node, the label is programmed and the MPLS data starts to flow along the path.

Figure 7 illustrates an LSP setup for non-O-UNI applications. In the case of an O-UNI application, the RSVP signaling messages are exchanged on a control channel, and the corresponding data channel to be used is acquired from the LMP Manager module based on the control channel. Also the O-UNI LSP's are by default bidirectional while the MPLS-TE LSP's are uni-directional.

High Availability

RSVP has been designed to ensure nonstop forwarding under the following constraints:

- Ability to tolerate the failure of one or more MPLS/O-UNI processes.
- Ability to tolerate the failure of one RP of a 1:1 redundant pair.
- Hitless software upgrade.

The RSVP high availability (HA) design follows the constraints of the underlying architecture where processes can fail without affecting the operation of other processes. A process failure of RSVP or any of its collaborators does not cause any traffic loss or cause established LSPs to go down. When RSVP restarts, it recovers its signaling states from its neighbors. No special configuration or manual intervention are required. You may configure RSVP graceful restart, which offers a standard mechanism to recover RSVP state information from neighbors after a failure.

Graceful Restart

RSVP graceful restart provides a control plane mechanism to ensure high availability, which allows detection and recovery from failure conditions while preserving nonstop forwarding services on the systems running Cisco IOS XR software.

RSVP graceful restart provides a mechanism that minimizes the negative effects on MPLS traffic caused by the following types of faults:

- Disruption of communication channels between two nodes when the communication channels are separate from the data channels. This is called control channel failure.
- The control plane of a node fails but the node preserves its data forwarding states. This is called node failure.

The procedure for RSVP graceful restart is described in the "Fault Handling" section of RFC 3473: *Generalized MPLS Signaling, RSVP-TE Extensions*. One of the main advantages of using RSVP graceful restart is recovery of the control plane while preserving nonstop forwarding and existing labels.

Graceful Restart: Standard and Interface-Based

When you configure RSVP graceful restart, Cisco IOS XR software sends and expects node-id address based Hello messages (that is, Hello Request and Hello Ack messages). The RSVP graceful restart Hello session is not established if the neighbor router does not respond with a node-id based Hello Ack message.

You can also configure graceful restart to respond (send Hello Ack messages) to interface-address based Hello messages sent from a neighbor router in order to establish a graceful restart Hello session on the neighbor router. If the neighbor router does not respond with node-id based Hello Ack message, however, the RSVP graceful restart Hello session is not established.

Cisco IOS XR software provides two commands to configure graceful restart:

- **signalling hello graceful-restart**
- **signalling hello graceful-restart interface-based**

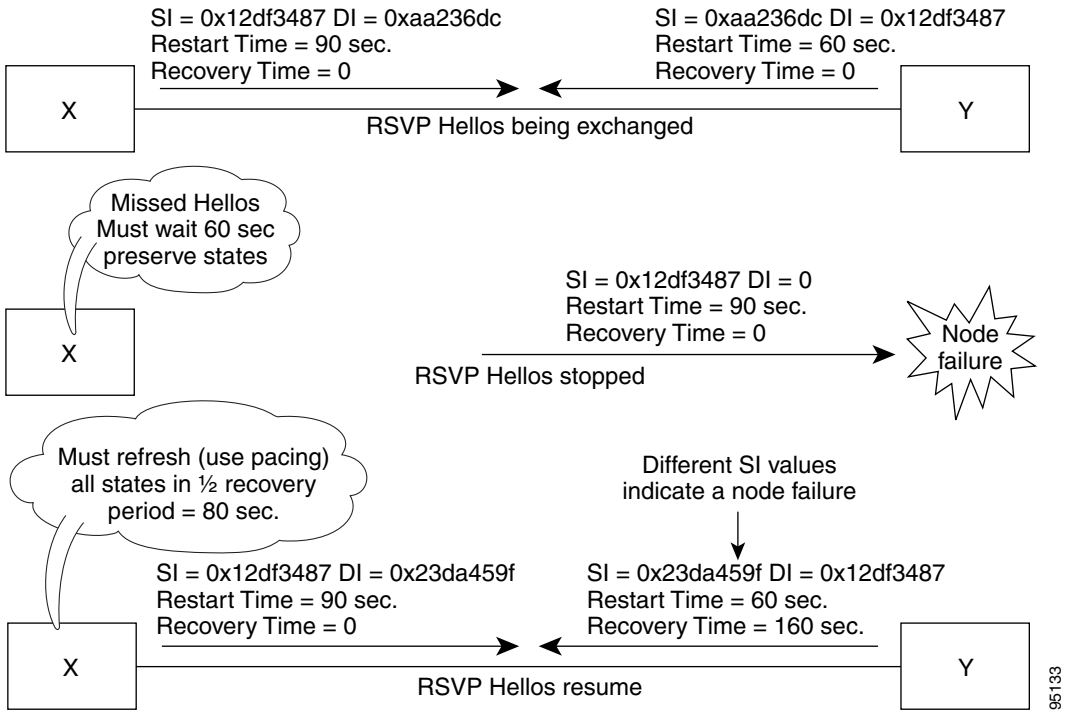
Note By default, graceful restart is disabled. To enable interface-based graceful restart, you must first enable standard graceful restart. You cannot enable interface-based graceful restart independently.

For detailed configuration steps, refer to Enabling Graceful Restart, page MPC-63.

Graceful Restart: Figure

Figure 8 illustrates how RSVP graceful restart handles a node failure condition.

Figure 8 Node Failure with RSVP



RSVP graceful restart requires the use of RSVP hello messages. Hello messages are used between RSVP neighbors. Each neighbor can autonomously issue a hello message containing a hello request object. A receiver that supports the hello extension replies with a hello message containing a hello acknowledgement (ACK) object. This means that a hello message contains either a hello Request or a hello ACK object. These two objects have the same format.

The restart cap object indicates a node's restart capabilities. It is carried in hello messages if the sending node supports state recovery. The restart cap object has the following two fields:

- Restart Time: Time after a loss in Hello messages within which RSVP hello session can be reestablished. It is possible for a user to manually configure the Restart Time.
- Recovery Time: Time that the sender waits for the recipient to re-synchronize states after the re-establishment of hello messages. This value is computed and advertised based on number of states that existed before the fault occurred.

For graceful restart, the hello messages are sent with an IP Time to Live (TTL) of 64. This is because the destination of the hello messages can be multiple hops away. If graceful restart is enabled, hello messages (containing the restart cap object) are send to an RSVP neighbor when RSVP states are shared with that neighbor.

Restart cap objects are sent to an RSVP neighbor when RSVP states are shared with that neighbor. If the neighbor replies with hello messages containing the restart cap object, the neighbor is considered to be graceful restart capable. If the neighbor does not reply with hello messages or replies with hello messages that do not contain the restart cap object, RSVP backs off sending hellos to that neighbor. If graceful restart is disabled, no hello messages (Requests or ACKs) are sent. If a hello Request message is received from an unknown neighbor, no hello ACK is sent back.

ACL-based Prefix Filtering

RSVP provides for the configuration of extended access lists (ACLs) to forward, drop, or perform normal processing on RSVP Router-Alert (RA) packets. Prefix filtering is designed for use at core access routers in order that RA packets (identified by a source/destination address) can be seamlessly forwarded across the core from one access point to another (or, conversely to be dropped at this node). RSVP applies prefix filtering rules only to RA packets because RA packets contain source and destination addresses of the RSVP flow.

Note RA packets forwarded due to prefix filtering must not be sent as RSVP bundle messages, because bundle messages are hop-by-hop and do not contain RA. Forwarding a Bundle message does not work, because the node receiving the messages is expected to apply prefix filtering rules only to RA packets.

For each incoming RSVP RA packet, RSVP inspects the IP header and attempts to match the source/destination IP addresses with a prefix configured in an extended ACL. The results are as follows:

- If an ACL does not exist, the packet is processed like a normal RSVP packet.
- If the ACL match yields an explicit permit (and if the packet is not locally destined), the packet is forwarded. The IP TTL is decremented on all forwarded packets.
- If the ACL match yields an explicit deny, the packet is dropped.

If there is no explicit permit or explicit deny, the ACL infrastructure returns an implicit (default) deny. RSVP may be configured to drop the packet. By default, RSVP processes the packet if the ACL match yields an implicit (default) deny.

Information About Implementing RSVP Authentication

Before implementing RSVP authentication, you must configure a keychain first. The name of the keychain must be the same as the one used in the keychain configuration. For more information about configuring keychains, see *Cisco IOS XR System Security Configuration Guide*.

Note RSVP authentication supports only keyed-hash message authentication code (HMAC) type algorithms.

To implement RSVP authentication on Cisco IOS XR software, you must understand the following concepts:

- RSVP Authentication Functions, page MPC-58
- RSVP Authentication Design, page MPC-58
- Global, Interface, and Neighbor Authentication Modes, page MPC-58
- Security Association, page MPC-59
- Key-source Key-chain, page MPC-60
- Guidelines for Window-Size and Out-of-Sequence Messages, page MPC-61
- Caveats for Out-of-Sequence, page MPC-61

RSVP Authentication Functions

You can carry out the following tasks with RSVP authentication:

- Set up a secure relationship with a neighbor by using secret keys that are known only to you and the neighbor.
- Configure RSVP authentication in global, interface, or neighbor configuration modes.
- Authenticate incoming messages by checking if there is a valid security relationship that is associated based on key identifier, incoming interface, sender address, and destination address.
- Add an integrity object with message digest to the outgoing message.
- Use sequence numbers in an integrity object to detect replay attacks.

RSVP Authentication Design

Network administrators need the ability to establish a security domain to control the set of systems that initiates RSVP requests.

The RSVP authentication feature permits neighbors in an RSVP network to use a secure hash to sign all RSVP signaling messages digitally, thus allowing the receiver of an RSVP message to verify the sender of the message without relying solely on the sender's IP address.

The signature is accomplished on a per-RSVP-hop basis with an RSVP integrity object in the RSVP message as defined in RFC 2747. This method provides protection against forgery or message modification. However, the receiver must know the security key used by the sender to validate the digital signature in the received RSVP message.

Network administrators manually configure a common key for each RSVP neighbor on the shared network.

The following reasons explain how to choose between global, interface, or neighbor configuration modes:

- Global configuration mode is optimal when a router belongs to a single security domain (for example, part of a set of provider core routers). A single common key set is expected to be used to authenticate all RSVP messages.
- Interface, or neighbor configuration mode, is optimal when a router belongs to more than one security domain. For example, a provider router is adjacent to the provider edge (PE), or a PE is adjacent to an edge device. Different keys can be used but not shared.

Global configuration mode configures the defaults for interface and neighbor interface modes. These modes, unless explicitly configured, inherit the parameters from global configuration mode, as follows:

- Window-size is set to 1.
- Lifetime is set to 1800.
- The **key-source key-chain** command is set to none or disabled.

Global, Interface, and Neighbor Authentication Modes

You can configure global defaults for all authentication parameters including key, window size, and lifetime. These defaults are inherited when you configure authentication for each neighbor or interface. However, you can also configure these parameters individually on a neighbor or interface basis, in which case the global values (configured or default) are no longer inherited.

Note RSVP uses the following rules when choosing which authentication parameter to use when that parameter is configured at multiple levels (interface, neighbor, or global). RSVP goes from the most specific to least specific; that is, neighbor, interface, and global.

Global keys simplify the configuration and eliminate the chances of a key mismatch when receiving messages from multiple neighbors and multiple interfaces. However, global keys do not provide the best security.

Interface keys are used to secure specific interfaces between two RSVP neighbors. Because many of the RSVP messages are IP routed, there are many scenarios in which using interface keys are not recommended. If all keys on the interfaces are not the same, there is a risk of a key mismatch for the following reasons:

- When the RSVP graceful restart is enabled, RSVP hello messages are sent with a source IP address of the local router ID and a destination IP address of the neighbor router ID. Because multiple routes can exist between the two neighbors, the RSVP hello message can traverse to different interfaces.
- When the RSVP Fast Reroute (FRR) is active, the RSVP Path and Resv messages can traverse multiple interfaces.
- When Generalized Multiprotocol Label Switching (GMPLS) optical tunnels are configured, RSVP messages are exchanged with router IDs as the source and destination IP addresses. Since multiple control channels can exist between the two neighbors, the RSVP messages can traverse different interfaces.

Neighbor-based keys are particularly useful in a network in which some neighbors support RSVP authentication procedures and others do not. When the neighbor-based keys are configured for a particular neighbor, you are advised to configure all the neighbor's addresses and router IDs for RSVP authentication.

Security Association

A security association (SA) is defined as a collection of information that is required to maintain secure communications with a peer to counter replay attacks, spoofing, and packet corruption.

Table 2 lists the main parameters that define a security association.

Table 2 Security Association Main Parameters

Parameter	Description
src	IP address of the sender.
dst	IP address of the final destination.
interface	Interface of the SA.
direction	Send or receive type of the SA.
Lifetime	Expiration timer value that is used to collect unused security association data.
Sequence Number	Last sequence number that was either sent or accepted (dependent of the direction type).
key-source	Source of keys for the configurable parameter.
keyID	Key number (returned form the key-source) that was last used.

Table 2 Security Association Main Parameters (continued)

Parameter	Description
digest	Algorithm last used (returned from the key-source).
Window Size	Specifies the tolerance for the configurable parameter. The parameter is applicable when the direction parameter is the receive type.
Window	Specifies the last *window size* value sequence number that is received or accepted. The parameter is applicable when the direction parameter is the receive type.

An SA is created dynamically when sending and receiving messages that require authentication. The neighbor, source, and destination addresses are obtained either from the IP header or from an RSVP object, such as a HOP object, and whether the message is incoming or outgoing.

When the SA is created, an expiration timer is created. When the SA authenticates a message, it is marked as recently used. The lifetime timer periodically checks if the SA is being used. If so, the flag is cleared and is cleaned up for the next period unless it is marked again.

Table 3 shows how to locate the source and destination address keys for an SA that is based on the message type.

Table 3 Source and Destination Address Locations for Different Message Types

Message Type	Source Address Location	Destination Address Location
Path	HOP object	SESSION object
PathTear	HOP object	SESSION object
PathError	HOP object	IP header
Resv	HOP object	IP header
ResvTear	HOP object	IP header
ResvError	HOP object	IP header
ResvConfirm	IP header	CONFIRM object
Ack	IP header	IP header
Srefresh	IP header	IP header
Hello	IP header	IP header
Bundle	—	—

Key-source Key-chain

The key-source key-chain is used to specify which keys to use.

You configure a list of keys with specific IDs and have different lifetimes so that keys are changed at predetermined intervals automatically, without any disruption of service. Rollover enhances network security by minimizing the problems that could result if an untrusted source obtained, deduced, or guessed the current key.

RSVP handles rollover by using the following key ID types:

- On TX, use the youngest eligible key ID.
- On RX, use the key ID that is received in an integrity object.

For more information about implementing keychain management on Cisco IOS XR Software, see *Cisco IOS XR System Security Configuration Guide*.

Guidelines for Window-Size and Out-of-Sequence Messages

The following guidelines are required for window-size and out-of-sequence messages:

- The default window-size is set to 1. If a single message is received out-of-sequence, RSVP rejects it and displays a message.
- When RSVP messages are sent in burst mode (for example, tunnel optimization), some messages can become out-of-sequence for a short amount of time.
- The window size can be increased by using the **window-size** command. When the window size is increased, replay attacks can be detected with duplicate sequence numbers.

Caveats for Out-of-Sequence

The following caveats are listed for out-of-sequence:

- When RSVP messages traverse multiple interface types with different maximum transmission unit (MTU) values, some messages can become out-of-sequence if they are fragmented.
- Packets with some IP options may be reordered.
- A change in QoS configurations may lead to a transient reorder of packets.
- QoS policies can cause a reorder of packets in a steady state.

Because all out-of-sequence messages are dropped, the sender must retransmit them. Because RSVP state timeouts are generally long, out-of-sequence messages during a transient state do not lead to a state timeout.

How to Implement RSVP

RSVP requires coordination among several routers, establishing exchange of RSVP messages to set up LSPs. Depending on the client application, RSVP requires some basic configuration, as described in the following sections:

- Configuring Traffic Engineering Tunnel Bandwidth, page MPC-61
- Confirming DiffServ-TE Bandwidth, page MPC-62
- Configuring MPLS O-UNI Bandwidth, page MPC-63
- Enabling Graceful Restart, page MPC-63
- Configuring ACL-based Prefix Filtering, page MPC-65
- Verifying RSVP Configuration, page MPC-68

Configuring Traffic Engineering Tunnel Bandwidth

To configure traffic engineering tunnel bandwidth, you must first set up TE tunnels and configure the reserved bandwidth per interface (there is no need to configure bandwidth for the data channel or the control channel).

Cisco IOS XR software supports two DS-TE modes: Prestandard and IETF.

The configuration steps for each option are described in the following sections in *Implementing MPLS Traffic Engineering on Cisco IOS XR Software*:

- Configuring a Prestandard Diff-Serv TE Tunnel, page MPC-127
- Configuring an IETF Diff-Serv TE Tunnel Using RDM, page MPC-129
- Configuring an IETF Diff-Serv TE Tunnel Using MAM, page MPC-131

Note For prestandard DS-TE you do not need to configure bandwidth for the data channel or the control channel. There is no other specific RSVP configuration required for this application.

Note When no RSVP bandwidth is specified for a particular interface, you can specify zero bandwidth in the LSP setup if it is configured under RSVP interface configuration mode or MPLS-TE configuration mode.

Confirming DiffServ-TE Bandwidth

Perform this task to confirm DiffServ-TE bandwidth.

In RSVP global and subpools, reservable bandwidths are configured per interface to accommodate TE tunnels on the node. Available bandwidth from all configured bandwidth pools is advertised using IGP. RSVP is used to signal the TE tunnel with appropriate bandwidth pool requirements.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **interface** *interface-id*
4. **bandwidth** *total-bandwidth max-flow* **sub-pool** *sub-pool-bw*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters RSVP configuration mode.

	Command or Action	Purpose
Step 3	**interface** *interface-id* **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# interface pos 0/2/0/0`	Enters interface configuration mode for the RSVP protocol.
Step 4	**bandwidth** *total-bandwidth max-flow* **sub-pool** *sub-pool-bw* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 1000 100 sub-pool 150`	Sets the reservable bandwidth, the maximum RSVP bandwidth available for a flow and the sub-pool bandwidth on this interface.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring MPLS O-UNI Bandwidth

For this application you do not need to configure bandwidth for the data channel or the control channel. There is no other specific RSVP configuration needed for this application.

Enabling Graceful Restart

Perform this task to enable graceful restart for implementations using both node-id- and interface-based hellos.

RSVP graceful restart provides a control plane mechanism to ensure high availability, which allows detection and recovery from failure conditions while preserving nonstop forwarding services.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **signalling graceful-restart**
4. **signalling graceful-restart interface-based**
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure terminal`	Enters global configuration mode
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters the RSVP configuration submode.
Step 3	**signalling graceful-restart** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling graceful-restart`	Enables the graceful restart process on the node.

	Command or Action	Purpose
Step 4	**signalling graceful-restart interface-based** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling graceful-restart interface-based`	Enables interface-based graceful restart process on the node.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring ACL-based Prefix Filtering

This section includes two procedures associated with RSVP Prefix Filtering:

- Configuring ACLs for Prefix Filtering, page MPC-65
- Configuring RSVP Packet Dropping, page MPC-66

Configuring ACLs for Prefix Filtering

Perform this task to configure an extended access list ACL that identifies the source and destination prefixes used for packet filtering.

Note The extended ACL needs to be configured separately using extended ACL configuration commands.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **signalling prefix-filtering access-list**

4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters the RSVP configuration submode.
Step 3	**signalling prefix-filtering access-list** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling prefix-filtering access-list banks`	Enter an extended access list name as a string.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring RSVP Packet Dropping

Perform this task to configure RSVP to drop RA packets when the ACL match returns an implicit (default) deny.

Note The default behavior will perform normal RSVP processing on RA packets when the ACL match returns an implicit (default) deny.

SUMMARY STEPS

1. **configure**
2. **rsvp**
3. **signalling prefix-filtering default-deny-action drop**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode
Step 2	**rsvp** **Example:** `RP/0/RP0/CPU0:router(config)# rsvp`	Enters the RSVP configuration submode.

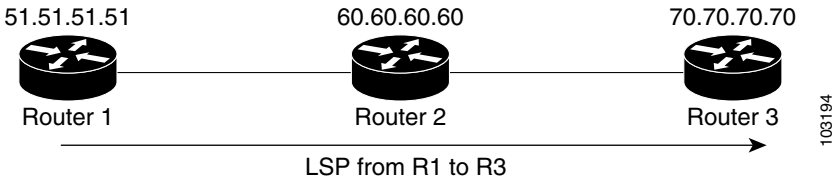
	Command or Action	Purpose
Step 3	**signalling prefix-filtering default-deny-action** **Example:** `RP/0/RP0/CPU0:router(config-rsvp)# signalling prefix-filtering default-deny-action`	Drops RA messages.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying RSVP Configuration

Figure 9 illustrates the topology that forms the basis for this section.

Figure 9 Sample Topology



To verify RSVP configuration, perform the following steps.

SUMMARY STEPS

1. **show rsvp session**
2. **show rsvp counters messages summary**
3. **show rsvp counters events**
4. **show rsvp interface** *type interface-id* [**detail**]

5. **show rsvp graceful-restart**
6. **show rsvp graceful-restart** [**neighbors** *ip-address* | **detail**]
7. **show rsvp interface**
8. **show rsvp neighbor**

DETAILED STEPS

Step 1 show rsvp session

Use this command to verify that all routers on the path of the LSP are configured with at least one Path State Block (PSB) and one Reservation State Block (RSB) per session. For example:

```
RP/0/RP0/CPU0:router# show rsvp session

Type Destination Add DPort  Proto/ExtTunID  PSBs  RSBs  Reqs
---- --------------- ----- --------------- ----- ----- -----
LSP4     172.16.70.70     6     10.51.51.51     1     1     0
```

In the example above, the output represents an LSP from ingress (head) router 10.51.51.51 to egress (tail) router 172.16.70.70. The tunnel ID (a.k.a destination port) is 6.

- If no states can be found for a session that should be up, verify the application (for example, MPLS-TE and O-UNI) to see if everything is in order.
- If a session has one PSB but no RSB, this indicates that either the Path message is not making it to the egress (tail) router or the reservation message is not making it back to the router R1 in question.

Go to the downstream router R2 and display the session information:

- If R2 has no PSB, either the path message is not making it to the router or the path message is being rejected (for example, due to lack of resources).
- If R2 has a PSB but no RSB, go to the next downstream router R3 to investigate.
- If R2 has a PSB and an RSB, this means the reservation is not making it from R2 to R1 or is being rejected.

Step 2 show rsvp counters messages summary

Use this command to verify whether RSVP message are being transmitted and received. For example:

```
RP/0/RP0/CPU0:router# show rsvp counters messages summary

 All RSVP Interfaces    Recv     Xmit                        Recv     Xmit
    Path                   0       25    Resv                   30        0
    PathError              0        0    ResvError               0        1
    PathTear               0       30    ResvTear               12        0
    ResvConfirm            0        0    Ack                    24       37
    Bundle                 0             Hello                   0     5099
    SRefresh            8974     9012    OutOfOrder              0
    Retransmit                     20    Rate Limited                     0
```

Step 3 show rsvp counters events

Use this command to see how many RSVP states have expired. Since RSVP uses a soft-state mechanism, some failures will lead to RSVP states to expire due to lack of refresh from the neighbor. For example:

```
RP/0/RP0/CPU0:router# show rsvp counters events

mgmtEthernet0/0/0/0                        tunnel6
 Expired Path states           0          Expired Path states           0
```

```
 Expired Resv states           0          Expired Resv states           0
 NACKs received                0          NACKs received                0
POS0/3/0/0                              POS0/3/0/1
 Expired Path states           0          Expired Path states           0
 Expired Resv states           0          Expired Resv states           0
 NACKs received                0          NACKs received                0
POS0/3/0/2                              POS0/3/0/3
 Expired Path states           0          Expired Path states           0
 Expired Resv states           0          Expired Resv states           1
 NACKs received                0          NACKs received                1
```

Step 4 **show rsvp interface** *type interface-id* [**detail**]

Use this command to verify that refresh reduction is working on a particular interface. For example:

```
RP/0/RP0/CPU0:router# show rsvp interface pos0/3/0/3 detail

INTERFACE: POS0/3/0/3 (ifh=0x4000D00).
 BW (bits/sec): Max=1000M. MaxFlow=1000M. Allocated=1K (0%). MaxSub=0.
 Signalling: No DSCP marking. No rate limiting.
 States in: 1. Max missed msgs: 4.
 Expiry timer: Running (every 30s). Refresh interval: 45s.
 Normal Refresh timer: Not running. Summary refresh timer: Running.
 Refresh reduction local: Enabled. Summary Refresh: Enabled (4096 bytes max).
 Reliable summary refresh: Disabled.
 Ack hold: 400 ms, Ack max size: 4096 bytes. Retransmit: 900ms.
 Neighbor information:
    Neighbor-IP    Nbor-MsgIds States-out  Refresh-Reduction Expiry(min::sec)
 -------------- -------------- ---------- ------------------ ----------------
    64.64.64.65              1          1            Enabled 14::45
```

Step 5 **show rsvp graceful-restart**

Use this command to verify that graceful restart is enabled locally. For example:

```
RP/0/RP0/CPU0:router# show rsvp graceful-restart

Graceful restart: enabled  Number of global neighbors: 1
Local MPLS router id: 10.51.51.51
Restart time: 60 seconds  Recovery time: 0 seconds
Recovery timer: Not running
Hello interval: 5000 milliseconds  Maximum Hello miss-count: 3
```

Step 6 **show rsvp graceful-restart** [**neighbors** *ip-address* | **detail**]

Use this command to verify that graceful restart is enabled on the neighbor(s). In the following examples, the neighbor 192.168.60.60 is not responding to hello messages:

```
RP/0/RP0/CPU0:router# show rsvp graceful-restart neighbors

Neighbor          App  State Recovery       Reason                Since  LostCnt
--------------- ----- ------ -------- ------------ -------------------- --------
192.168.60.60     MPLS   INIT     DONE          N/A  12/06/2003 19:01:49        0

RP/0/RP0/CPU0:router# show rsvp graceful-restart neighbors detail

Neighbor: 192.168.60.60 Source: 10.51.51.51 (MPLS)
  Hello instance for application MPLS
    Hello State: INIT        (for 3d23h)
    Number of times communications with neighbor lost: 0
    Reason: N/A
  Recovery State: DONE
  Number of Interface neighbors: 1
    address: 10.64.64.65
  Restart time: 0 seconds  Recovery time: 0 seconds
```

```
  Restart timer: Not running
  Recovery timer: Not running
  Hello interval: 5000 milliseconds  Maximum allowed missed Hello messages: 3
```

Step 7 **show rsvp interface**

Use this command to verify available RSVP bandwidth. For example:

```
RP/0/RP0/CPU0:router# show rsvp interface

Interface   MaxBW    MaxFlow  Allocated       MaxSub
----------- -------- -------- --------------- --------
Et0/0/0/0          0        0        0 (  0%)        0
PO0/3/0/0      1000M    1000M        0 (  0%)        0
PO0/3/0/1      1000M    1000M        0 (  0%)        0
PO0/3/0/2      1000M    1000M        0 (  0%)        0
PO0/3/0/3      1000M    1000M       1K (  0%)        0
```

Step 8 **show rsvp neighbor**

Use this command to verify RSVP neighbors. For example:

```
RP/0/RP0/CPU0:router# show rsvp neighbor detail

Global Neighbor: 40.40.40.40
Interface Neighbor: 1.1.1.1
Interface: POS0/0/0/0
Refresh Reduction: "Enabled" or "Disabled".
Remote epoch: 0xXXXXXXXX
Out of order messages: 0
Retransmitted messages: 0
Interface Neighbor: 2.2.2.2
Interface: POS0/1/0/0
Refresh Reduction: "Enabled" or "Disabled".
Remote epoch: 0xXXXXXXXX
Out of order messages: 0
Retransmitted messages: 0
```

How to Implement RSVP Authentication

There are three types of RSVP authentication modes—global, interface, and neighbor. The sections that follow describe how to implement RSVP authentication for each mode:

- Configuring Global Configuration Mode RSVP Authentication, page MPC-72
- Configuring an Interface for RSVP Authentication, page MPC-76
- Configuring RSVP Neighbor Authentication, page MPC-82
- Verifying the Details of the RSVP Authentication, page MPC-88
- Eliminating Security Associations for RSVP Authentication, page MPC-88

Configuring Global Configuration Mode RSVP Authentication

This section includes the following procedures for RSVP authentication in global configuration mode, as follows:

- Enabling RSVP Authentication Using the Keychain in Global Configuration Mode, page MPC-72
- Configuring a Lifetime for RSVP Authentication in Global Configuration Mode, page MPC-73
- Configuring the Window Size for RSVP Authentication in Global Configuration Mode, page MPC-74

Enabling RSVP Authentication Using the Keychain in Global Configuration Mode

Perform this task to enable RSVP authentication for cryptographic authentication by specifying the keychain in global configuration mode.

Note You must configure a keychain before completing this task (see *Cisco IOS XR System Security Configuration Guide).*

SUMMARY STEPS

1. **configure**
2. **rsvp authentication**
3. **key-source key-chain** *key-chain-name*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode

Step 2	`rsvp authentication`
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 3	**key-source key-chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-auth)# key-source key-chain mpls-keys`	Specifies the source of the key information to authenticate RSVP signaling messages. The *key-chain-name* argument is used to specify the name of the keychain. The maximum number of characters is 32.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Lifetime for RSVP Authentication in Global Configuration Mode

Perform this task to configure a lifetime value for RSVP authentication in global configuration mode.

SUMMARY STEPS

1. **configure**
2. **rsvp authentication**
3. **life-time** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-auth)#`
		Enters RSVP authentication configuration mode.

Step 3	**life-time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-auth)# life-time 2000`	Controls how long Resource Reservation Protocol (RSVP) maintains security associations with other trusted RSVP neighbors. • Use the *seconds* argument to specify the length of time (in seconds) that RSVP maintains idle security associations with other trusted RSVP neighbors. Range is from 30 to 86400. The default value is 1800.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Window Size for RSVP Authentication in Global Configuration Mode

Perform this task to configure the window size for RSVP authentication in global configuration mode.

SUMMARY STEPS

1. **configure**
2. **rsvp authentication**
3. **window-size** {*N*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 3	**window-size** {*N*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-auth)# window-size 33`	Specifies the maximum number of Resource Reservation Protocol (RSVP) authenticated messages that can be received out-of-sequence. • Use the *N* argument to specify the Size of the window to restrict out-of-sequence messages. The range is from 1 to 64. The default value is 1, in which case all out-of-sequence messages are dropped.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an Interface for RSVP Authentication

This section contains the following procedures for configuring an interface for RSVP authentication:

- Specifying the RSVP Authentication Keychain in Interface Mode, page MPC-76
- Configuring a Lifetime for an Interface for RSVP Authentication, page MPC-78
- Configuring the Window Size for an Interface for RSVP Authentication, page MPC-80

Specifying the RSVP Authentication Keychain in Interface Mode

Perform this task to specify RSVP authentication keychain in interface mode.

You must configure a keychain first (see *Cisco IOS XR System Security Configuration Guide*).

SUMMARY STEPS

1. **configure**
2. **rsvp interface** {*type interface-id*}
3. **authentication**
4. **key-source key-chain** *key-chain-name*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp interface** {*type interface-id*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp interface POS 0/2/1/0`
	`RP/0/RP0/CPU0:router(config-rsvp-if)#`
		Enters RSVP interface configuration mode.

Step 3	**authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 4	**key-source key-chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-auth)# key-source key-chain mpls-keys`	Specifies the source of the key information to authenticate RSVP signaling messages. The *key-chain-name* argument is used to specify the name of the keychain. The maximum number of characters is 32.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Lifetime for an Interface for RSVP Authentication

Perform this task to configure a lifetime for the security association for an interface.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** {*type interface-id*}
3. **authentication**
4. **life-time** *seconds*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp interface** {*type interface-id*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp interface POS 0/2/1/0`
	`RP/0/RP0/CPU0:router(config-rsvp-if)#`
		Enters RSVP interface configuration mode.

Step 3	**authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)#`
		Enters RSVP authentication configuration mode.

	Command or Action	Purpose
Step 4	**life-time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-auth)# life-time 2000`	Controls how long Resource Reservation Protocol (RSVP) maintains security associations with other trusted RSVP neighbors. • Use the *seconds* argument to specify the length of time (in seconds) that RSVP maintains idle security associations with other trusted RSVP neighbors. Range is from 30 to 86400. The default value is 1800.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Window Size for an Interface for RSVP Authentication

Perform this task to configure the window size for an interface for RSVP authentication to check the validity of the sequence number received.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** {*type interface-id*}
3. **authentication**
4. **window-size** {*N*}
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp interface** {*type interface-id*}
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp interface POS 0/2/1/0`
	`RP/0/RP0/CPU0:router(config-rsvp-if)#`
		Enters RSVP interface configuration mode.

Step 3	**authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)#`
		Enters RSVP interface authentication configuration mode.

	Command or Action	Purpose
Step 4	**window-size** {*N*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-auth)# window-size 33`	Specifies the maximum number of Resource Reservation Protocol (RSVP) authenticated messages that can be received out-of-sequence. • Use the *N* argument to specify the size of the window to restrict out-of-sequence messages. The range is from 1 to 64. The default value is 1, in which case all out-of-sequence messages are dropped.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring RSVP Neighbor Authentication

This section contains the following procedures for RSVP neighbor authentication:

- Specifying the Keychain for RSVP Neighbor Authentication, page MPC-82
- Configuring a Lifetime for RSVP Neighbor Authentication, page MPC-84
- Configuring the Window Size for RSVP Neighbor Authentication, page MPC-86

Specifying the Keychain for RSVP Neighbor Authentication

Perform this task to specify the keychain RSVP neighbor authentication.

You must configure a keychain first (see *Cisco IOS XR System Security Configuration Guide).*

SUMMARY STEPS

1. **configure**
2. **rsvp neighbor** *IP address* **authentication**
3. **key-source key-chain** *key-chain-name*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp neighbor** *IP address* **authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp neighbor 1.1.1.1 authentication`
	`P/0/RP0/CPU0:router(config-rsvp-nbor-auth)#`
		Enters neighbor authentication configuration mode. Use the **rsvp neighbor** command to activate Resource Reservation Protocol (RSVP) cryptographic authentication for a neighbor.
		• Use the *IP address* argument to specify the IP address of the neighbor. A single IP address for a specific neighbor; usually one of the neighbor's physical or logical (loopback) interfaces.
		• Use the **authentication** keyword to configure the RSVP authentication parameters.

	Command or Action	Purpose
Step 3	**key-source key-chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# key-source key-chain mpls-keys`	Specifies the source of the key information to authenticate RSVP signaling messages. The *key-chain-name* argument is used to specify the name of the keychain. The maximum number of characters is 32.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Lifetime for RSVP Neighbor Authentication

Perform this task to configure a lifetime for security association for RSVP neighbor authentication mode.

SUMMARY STEPS

1. **configure**
2. **rsvp neighbor** *IP address* **authentication**
3. **life-time** *seconds*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp neighbor** *IP address* **authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp neighbor 1.1.1.1 authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)#`
		Enters RSVP neighbor authentication configuration mode. Use the **rsvp neighbor** command to specify a neighbor under RSVP.
		• Use the *IP address* argument to specify the IP address of the neighbor. A single IP address for a specific neighbor; usually one of the neighbor's physical or logical (loopback) interfaces.
		• Use the **authentication** keyword to configure the RSVP authentication parameters.

	Command or Action	Purpose
Step 3	**life-time** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# life-time 2000`	Controls how long Resource Reservation Protocol (RSVP) maintains security associations with other trusted RSVP neighbors. • Use the *seconds* argument to specify the length of time (in seconds) that RSVP maintains idle security associations with other trusted RSVP neighbors. Range is from 30 to 86400. The default value is 1800.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Window Size for RSVP Neighbor Authentication

Perform this task to configure the RSVP neighbor authentication window size to check the validity of the sequence number received.

SUMMARY STEPS

1. **configure**
2. **rsvp neighbor** *IP address* **authentication**
3. **window-size** {*N*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode

Step 2	**rsvp neighbor** *IP address* **authentication**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# rsvp neighbor 1.1.1.1 authentication`
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)#`
		Enters RSVP neighbor authentication configuration mode. Use the **rsvp neighbor** command to specify a neighbor under RSVP.
		• Use the *IP address* argument to specify the IP address of the neighbor. A single IP address for a specific neighbor; usually one of the neighbor's physical or logical (loopback) interfaces.
		• Use the **authentication** keyword to configure the RSVP authentication parameters.

	Command or Action	Purpose
Step 3	**window-size** {*N*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# window-size 33`	Specifies the maximum number of Resource Reservation Protocol (RSVP) authenticated messages that can be received out-of-sequence. • Use the *N* argument to specify the Size of the window to restrict out-of-sequence messages. The range is from 1 to 64. The default value is 1, in which case all out-of-sequence messages are dropped.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-nbor-auth)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying the Details of the RSVP Authentication

To display the security associations that RSVP has established with other RSVP neighbors, use the **show rsvp authentication** command.

Eliminating Security Associations for RSVP Authentication

To eliminate RSVP authentication SA's, use the **clear rsvp authentication** command. To eliminate RSVP counters for each SA, use the **clear rsvp counters authentication** command.

Configuration Examples for RSVP

The following section gives sample RSVP configurations for some of the supported RSVP features. More details on the commands can be found in the *Resource Reservation Protocol Infrastructure Commands* guide. Examples are provided for the following features:

- Bandwidth Configuration (Prestandard): Example, page MPC-89
- Bandwidth Configuration (MAM): Example, page MPC-89
- Bandwidth Configuration (RDM): Example, page MPC-89
- Refresh Reduction and Reliable Messaging Configuration: Example, page MPC-89
- Configuring Graceful Restart: Example, page MPC-90
- Configuring ACL-based Prefix Filtering: Example, page MPC-91
- Setting DSCP for RSVP Packets: Example, page MPC-91

Bandwidth Configuration (Prestandard): Example

The following example shows the configuration of bandwidth on an interface using prestandard DS-TE mode. The example configures an interface for a reservable bandwidth of 7500, specifies the maximum bandwidth for one flow to be 1000 and adds a sub-pool bandwidth of 2000:

```
rsvp interface pos 0/3/0/0
 bandwidth 7500 1000 sub-pool 2000
```

Bandwidth Configuration (MAM): Example

The following example shows the configuration of bandwidth on an interface using MAM. The example shows how to limit the total of all RSVP reservations on POS interface 0/3/0/0 to 7500 kbps, and allows each single flow to reserve no more than 1000 kbps:

```
rsvp interface pos 0/3/0/0
 bandwidth mam 7500 1000
```

Bandwidth Configuration (RDM): Example

The following example shows the configuration of bandwidth on an interface using RDM. The example shows how to limit the total of all RSVP reservations on PoS interface 0/3/0/0 to 7500 kbps, and allows each single flow to reserve no more than 1000 kbps:

```
rsvp interface pos 0/3/0/0
 bandwidth rdm 7500 1000
```

Refresh Reduction and Reliable Messaging Configuration: Example

Refresh reduction feature as defined by RFC 2961 is supported and enabled by default. The following examples illustrate the configuration for the refresh reduction feature. Refresh reduction is used with a neighbor only if the neighbor supports it also.

Changing the Refresh Interval and the Number of Refresh Messages

The following example shows how to configure the refresh interval to 30 seconds on POS 0/3/0/0 and how to change the number of refresh messages the node can miss before cleaning up the state from the default value of 4 to 6:

```
rsvp interface pos 0/3/0/0
 signalling refresh interval 30
 signalling refresh missed 6
```

Configuring Retransmit Time Used in Reliable Messaging

The following example shows how to set the retransmit timer to 2 seconds. To prevent unnecessary retransmits, the retransmit time value configured on the interface must be greater than the ACK hold time on its peer.

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction reliable retransmit-time 2000
```

Configuring Acknowledgement Times

The following example shows how to change the acknowledge hold time from the default value of 400 ms, to delay or speed up sending of ACKs, and the maximum acknowledgment message size from default size of 4096 bytes.

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction reliable ack-hold-time 1000
rsvp interface pos 0/4/0/1
 signalling refresh reduction reliable ack-max-size 1000
```

Note Make sure retransmit time on the peers' interface is at least twice the amount of the ACK hold time to prevent unnecessary retransmissions.

Changing the Summary Refresh Message Size

The following example shows how to set the summary refresh message maximum size to 1500 bytes:

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction summary max-size 1500
```

Disabling Refresh Reduction

If the peer node does not support refresh reduction or for any other reason you want to disable refresh reduction on an interface, use the following commands to disable refresh reduction on that interface:

```
rsvp interface pos 0/4/0/1
 signalling refresh reduction disable
```

Configuring Graceful Restart: Example

RSVP graceful restart is configured globally or per interface (as are refresh-related parameters). The following examples show how to enable graceful restart, set the restart time, and change the hello message interval.

Enabling Graceful Restart

RSVP graceful restart is enabled by default. If disabled, enable it with the following command:

```
rsvp signalling graceful-restart
```

Enabling Interface-Based Graceful Restart

Configure the RSVP graceful restart feature on an interface using the following command:

```
signalling hello graceful-restart interface-based
```

Changing the Restart-Time

Configure the restart time that is advertised in hello messages sent to neighbor nodes:

```
rsvp signalling graceful-restart restart-time 200
```

Changing the Hello Interval

Configure the interval at which RSVP graceful restart hello messages are sent per neighbor, and change the number of hellos missed before the neighbor is declared down:

```
rsvp signalling hello graceful-restart refresh interval 4000
rsvp signalling hello graceful-restart refresh misses 4
```

Configuring ACL-based Prefix Filtering: Example

In the following example, when RSVP receives a Router Alert (RA) packet from source address 1.1.1.1 and 1.1.1.1 is not a local address, the packet is forwarded with IP TTL decremented. Packets destined to 2.2.2.2 are dropped. All other RA packets are processed as normal RSVP packets.

```
show run ipv4 access-list
  ipv4 access-list rsvpacl
  10 permit ip host 1.1.1.1 any
  20 deny ip any host 2.2.2.2
  !
show run rsvp
  rsvp
  signalling prefix-filtering access-list rsvpacl
  !
```

Setting DSCP for RSVP Packets: Example

The following configuration can be used to set the Differentiated Services Code Point (DSCP) field in the IP header of RSVP packets:

```
rsvp interface pos0/2/0/1
 signalling dscp 20
```

Configuration Examples for RSVP Authentication

This section provides the following configuration examples:

- RSVP Authentication Global Configuration Mode: Example, page MPC-92
- RSVP Authentication for an Interface: Example, page MPC-92
- RSVP Neighbor Authentication: Example, page MPC-92
- RSVP Authentication by Using All the Modes: Example, page MPC-93

RSVP Authentication Global Configuration Mode: Example

The following configuration is used to enable authentication of all RSVP messages and to increase the default lifetime of the SAs:

```
rsvp
 authentication
  key-source key-chain default_keys
  life-time 3600
 !
!
```

Note The specified keychain (default_keys) must exist and contain valid keys, or signaling will fail.

RSVP Authentication for an Interface: Example

The following configuration is used to enable authentication of all RSVP messages that are being sent or received on one interface only, and sets the window-size of the SA's:

```
rsvp
 interface GigabitEthernet0/6/0/0
  authentication
   window-size 64
  !
 !
```

Note Because the key-source keychain configuration is not specified, the global authentication mode keychain is used and inherited. The global keychain must exist and contain valid keys or signaling fails.

RSVP Neighbor Authentication: Example

The following configuration is used to enable authentication of all RSVP messages that being sent to and received from only a particular IP address:

```
rsvp
 neighbor 10.0.0.1
  authentication
   key-source key-chain nbr_keys
  !
 !
!
```

RSVP Authentication by Using All the Modes: Example

The following configuration shows how to perform the following functions:

- Authenticates all RSVP messages.
- Authenticates the RSVP messages to or from 10.0.0.1 by setting the keychain for the **key-source key-chain** command to nbr_keys, SA lifetime is set to 3600, and the default window-size is set to 1.
- Authenticates the RSVP messages not to or from 10.0.0.1 by setting the keychain for the **key-source key-chain** command to default_keys, SA lifetime is set to 3600, and the window-size is set 64 when using GigabitEthernet0/6/0/0; otherwise, the default value of 1 is used.

```
rsvp
 interface GigabitEthernet0/6/0/0
  authentication
   window-size 64
  !
 !
 neighbor 10.0.0.1
  authentication
   key-source key-chain nbr_keys
  !
 !
 authentication
  key-source key-chain default_keys
  life-time 3600
 !
!
```

Note If a keychain does not exist or contain valid keys, this is considered a configuration error because signaling fails. However, this can be intended to prevent signaling. For example, when using the above configuration, if the nbr_keys does not contain valid keys, all signaling with 10.0.0.1 fails.

Additional References

The following section provides references related to implementing MPLS RSVP:

Related Documents

Related Topic	Document Title
Cisco IOS XR MPLS RSVP commands	*RSVP Infrastructure Commands on Cisco IOS XR Software* module in the *Cisco IOS XR MPLS Command Reference*
Cisco CRS-1 getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
OIF2000.125.7	*User Network Interface (UNI) 1.0 Signaling Specification*

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs[1]	Title
RFC 2205	*Resource Reservation Protocol Version 1 Functional Specification*
RFC 2747	*RSVP Cryptographic Authentication*
RFC 3209	*RSVP-TE: Extensions to RSVP for LSP Tunnels*
RFC 2961	*RSVP Refresh Overhead Reduction Extensions*
RFC 3473	*Generalized MPLS Signaling, RSVP-TE Extensions*
RFC 4090	*Fast Reroute Extensions to RSVP-TE for LSP Tunnels*

1. Not all supported RFCs are listed.

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS Forwarding on Cisco IOS XR Software

All MPLS features require a core set of MPLS label management and forwarding services; the MPLS Forwarding Infrastructure (MFI) supplies these services.

MPLS combines the performance and capabilities of Layer 2 (data link layer) switching with the proven scalability of Layer 3 (network layer) routing. MPLS enables service providers to meet the challenges of growth in network utilization while providing the opportunity to differentiate services without sacrificing the existing network infrastructure. The MPLS architecture is flexible and can be employed in any combination of Layer 2 technologies. MPLS support is offered for all Layer 3 protocols, and scaling is possible well beyond that typically offered in today's networks.

MFI Control-Plane Services

The MFI control-plane provides services to MPLS applications, such as Label Distribution Protocol (LDP) and Traffic Engineering (TE), that include enabling and disabling MPLS on an interface, local label allocation, MPLS rewrite setup (including backup links), management of MPLS label tables, and the interaction with other forwarding paths (IPv4 for example) to set up imposition and disposition.

MFI Data-Plane Services

The MFI data-plane provides a software implementation of MPLS forwarding in all of the following forms:

- Imposition
- Disposition
- Label swapping

Implementing MPLS Traffic Engineering on Cisco IOS XR Software

Multiprotocol Label Switching (MPLS) is a standards-based solution driven by the Internet Engineering Task Force (IETF) that was devised to convert the Internet and IP backbones from best-effort networks into business-class transport mediums.

MPLS, with its label switching capabilities, eliminates the need for an IP route look-up and creates a virtual circuit (VC) switching function, allowing enterprises the same performance on their IP-based network services as with those delivered over traditional networks such as Frame Relay or Asynchronous Transfer Mode (ATM).

MPLS traffic engineering (MPLS-TE) software enables an MPLS backbone to replicate and expand upon the TE capabilities of Layer 2 ATM and Frame Relay networks. MPLS is an integration of Layer 2 and Layer 3 technologies. By making traditional Layer 2 features available to Layer 3, MPLS enables traffic engineering. Thus, you can offer in a one-tier network what now can be achieved only by overlaying a Layer 3 network on a Layer 2 network.

Feature History for Implementing MPLS-TE on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.3.0	Support was added for Generalized MPLS.
Release 3.4.0	Support was added for Flexible Name-based Tunnel Constraints, Interarea MPLS-TE, MPLS-TE Forwarding Adjacency, and GMPLS Protection and Restoration, and GMPLS Path Protection.
Release 3.4.1	Support was added for MPLS-TE and fast reroute link bundling on the Cisco CRS-1.
Release 3.5.0	Support was added for Unequal Load Balancing, IS-IS IP Fast Reroute Loop-free Alternative routing functionality, and Path Computation Element (PCE).

Release 3.6.0	No modification.
Release 3.7.0	Support was added for the following features: • PBTS for L2VPN and IPv6 traffic on the Cisco XR 12000 Series Router. • Ignore Intermediate System-to-Intermediate System (IS-IS) overload bit setting in MPLS-TE. • MPLS-TE/Fast Reroute (FRR) over Virtual Local Area Network (VLAN) interfaces on the Cisco CRS-1.

Contents

- Prerequisites for Implementing Cisco MPLS Traffic Engineering, page MPC-98
- Information About Implementing MPLS Traffic Engineering, page MPC-98
- How to Implement Traffic Engineering on Cisco IOS XR Software, page MPC-115
- Configuration Examples for Cisco MPLS-TE, page MPC-187
- Additional References, page MPC-196

Prerequisites for Implementing Cisco MPLS Traffic Engineering

The following prerequisites are required to implement MPLS TE:

- You must be in a user group associated with a task group that includes the proper task IDs for MPLS-TE commands.
- A router that runs Cisco IOS XR software.
- An installed composite mini-image and the MPLS package, or a full composite image.
- IGP activated.

Information About Implementing MPLS Traffic Engineering

To implement MPLS-TE, you should understand the concepts that are described in the following sections:

- Overview of MPLS Traffic Engineering, page MPC-99
- Protocol-Based CLI, page MPC-100
- Differentiated Services Traffic Engineering, page MPC-100
- Flooding, page MPC-102
- Fast Reroute, page MPC-103
- MPLS-TE and Fast Reroute over Link Bundles, page MPC-104
- Ignore Intermediate System-to-Intermediate System Overload Bit Setting in MPLS-TE, page MPC-104
- Generalized MPLS, page MPC-105
- Flexible Name-based Tunnel Constraints, page MPC-107

- MPLS Traffic Engineering Interarea Tunneling, page MPC-108
- MPLS-TE Forwarding Adjacency, page MPC-110
- Unequal Load Balancing, page MPC-111
- Path Computation Element, page MPC-112
- Policy-based Tunnel Selection Overview, page MPC-113

Overview of MPLS Traffic Engineering

MPLS-TE software enables an MPLS backbone to replicate and expand upon the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks. MPLS is an integration of Layer 2 and Layer 3 technologies. By making traditional Layer 2 features available to Layer 3, MPLS enables traffic engineering. Thus, you can offer in a one-tier network what now can be achieved only by overlaying a Layer 3 network on a Layer 2 network.

MPLS-TE is essential for service provider and Internet service provider (ISP) backbones. Such backbones must support a high use of transmission capacity, and the networks must be very resilient so that they can withstand link or node failures. MPLS-TE provides an integrated approach to traffic engineering. With MPLS, traffic engineering capabilities are integrated into Layer 3, which optimizes the routing of IP traffic, given the constraints imposed by backbone capacity and topology.

Benefits of MPLS Traffic Engineering

MPLS-TE enables ISPs to route network traffic to offer the best service to their users in terms of throughput and delay. By making the service provider more efficient, traffic engineering reduces the cost of the network.

Currently, some ISPs base their services on an overlay model. In the overlay model, transmission facilities are managed by Layer 2 switching. The routers see only a fully meshed virtual topology, making most destinations appear one hop away. If you use the explicit Layer 2 transit layer, you can precisely control how traffic uses available bandwidth. However, the overlay model has numerous disadvantages. MPLS-TE achieves the TE benefits of the overlay model without running a separate network and without a non-scalable, full mesh of router interconnects.

How MPLS-TE Works

MPLS-TE automatically establishes and maintains label switched paths (LSPs) across the backbone by using resource reservation protocol (RSVP). The path that an LSP uses is determined by the LSP resource requirements and network resources, such as bandwidth. Available resources are flooded by means of extensions to a link-state-based Interior Gateway Protocol (IGP).

MPLS-TE tunnels are calculated at the LSP headend router, based on a fit between the required and available resources (constraint-based routing). The IGP automatically routes the traffic to these LSPs.

Typically, a packet crossing the MPLS-TE backbone travels on a single LSP that connects the ingress point to the egress point. MPLS-TE is built on the following mechanisms:

- Tunnel interfaces—From a Layer 2 standpoint, an MPLS tunnel interface represents the headend of an LSP. It is configured with a set of resource requirements, such as bandwidth and media requirements, and priority. From a Layer 3 standpoint, an LSP tunnel interface is the headend of a unidirectional virtual link to the tunnel destination.

- MPLS-TE path calculation module—This calculation module operates at the LSP headend. The module determines a path to use for an LSP. The path calculation uses a link-state database containing flooded topology and resource information.
- RSVP with TE extensions—RSVP operates at each LSP hop and is used to signal and maintain LSPs based on the calculated path.
- MPLS-TE link management module—This module operates at each LSP hop, performs link call admission on the RSVP signaling messages, and performs bookkeeping on topology and resource information to be flooded.
- Link-state IGP (Intermediate System-to-Intermediate System [IS-IS] or Open Shortest Path First [OSPF]—each with traffic engineering extensions)—These IGPs are used to globally flood topology and resource information from the link management module.
- Enhancements to the shortest path first (SPF) calculation used by the link-state IGP (IS-IS or OSPF)—The IGP automatically routes traffic to the appropriate LSP tunnel, based on tunnel destination. Static routes can also be used to direct traffic to LSP tunnels.
- Label switching forwarding—This forwarding mechanism provides routers with a Layer 2-like ability to direct traffic across multiple hops of the LSP established by RSVP signaling.

One approach to engineering a backbone is to define a mesh of tunnels from every ingress device to every egress device. The MPLS-TE path calculation and signaling modules determine the path taken by the LSPs for these tunnels, subject to resource availability and the dynamic state of the network.

The IGP (operating at an ingress device) determines which traffic should go to which egress device, and steers that traffic into the tunnel from ingress to egress. A flow from an ingress device to an egress device might be so large that it cannot fit over a single link, so it cannot be carried by a single tunnel. In this case, multiple tunnels between a given ingress and egress can be configured, and the flow is distributed using load sharing among the tunnels.

Protocol-Based CLI

Cisco IOS XR software provides a protocol-based command line interface. The CLI provides commands that can be used with the multiple IGP protocols supported by MPLS-TE.

Differentiated Services Traffic Engineering

MPLS Differentiated Services (Diff-Serv) Aware Traffic Engineering (DS-TE) is an extension of the regular MPLS-TE feature. Regular traffic engineering does not provide bandwidth guarantees to different traffic classes. A single bandwidth constraint is used in regular TE that is shared by all traffic. To support various classes of service (CoS), users can configure multiple bandwidth constraints. These bandwidth constraints can be treated differently based on the requirement for the traffic class using that constraint.

MPLS diff-serv traffic engineering provides the ability to configure multiple bandwidth constraints on an MPLS-enabled interface. Available bandwidths from all configured bandwidth constraints are advertised using IGP. TE tunnel is configured with bandwidth value and class-type requirements. Path calculation and admission control take the bandwidth and class-type into consideration. RSVP is used to signal the TE tunnel with bandwidth and class-type requirements.

Diff-Serv TE can be deployed with either Russian Doll Model (RDM) or Maximum Allocation Model (MAM) for bandwidth calculations.

Cisco IOS XR software supports two DS-TE modes: Prestandard and IETF. Both modes are described in further detail in the sections that follow.

Prestandard DS-TE Mode

Prestandard DS-TE uses the Cisco proprietary mechanisms for RSVP signaling and IGP advertisements. This DS-TE mode does not interoperate with third-party vendor equipment. Note that prestandard DS-TE is enabled only after configuring the sub-pool bandwidth values on MPLS-enabled interfaces.

Prestandard Diff-Serve TE mode supports a single bandwidth constraint model, Russian Doll Model (RDM) with two bandwidth pools, global-pool and sub-pool.

Note TE class map is not used with Prestandard DS-TE mode.

IETF DS-TE Mode

IETF Diff-Serv TE mode uses IETF defined extensions for RSVP and IGP. This mode interoperates with third-party vendor equipment.

IETF mode supports multiple bandwidth constraint models, including the Russian Doll Model (RDM) and the Maximum Allocation Model (MAM) both with two bandwidth pools. Note that in an IETF DS-TE network, identical bandwidth constraint models must be configured on all nodes.

TE class map is used with IETF DS-TE mode and must be configured the same way on all nodes in the network.

Bandwidth Constraint Models

IETF DS-TE mode provides support for the Russian Dolls and Maximum Allocation bandwidth constraints models. Both models support up two bandwidth pools.

Cisco IOS XR provides global configuration for the switching between bandwidth constraint models. Both models can be configured on a single interface to pre-configure the bandwidth constraints before swapping to an alternate bandwidth constraint model.

Note NSF is not guaranteed when you change the bandwidth constraint model or configuration information.

By default, RDM is the default bandwidth constraint model used in both pre-standard and IETF mode.

Maximum Allocation Bandwidth Constraint Model

The MAM constraint model has the following characteristics:

- It is easy to use and intuitive.
- It ensures isolation across class types.
- It simultaneously achieves isolation, bandwidth efficiency, and protection against QoS degradation.

Russian Doll Bandwidth Constraint Model

The RDM constraint model has the following characteristics:

- It allows greater sharing of bandwidth among different class types.

- It simultaneously ensures bandwidth efficiency and protection against QoS degradation of all class types.
- It can be used in conjunction with preemption to simultaneously achieve isolation across class-types such that each class-type is guaranteed its share of bandwidth, bandwidth efficiency, and protection against QoS degradation of all class types.

Note We recommend that RDM not be used in DS-TE environments in which the use of preemption is precluded. While RDM ensures bandwidth efficiency and protection against QoS degradation of class types, it does guarantee isolation across class types.

TE Class Mapping

Each of the eight available bandwidth values advertised in the IGP corresponds to a TE Class. Because the IGP advertises only eight bandwidth values, there can be a maximum of only eight TE classes supported in an IETF DS-TE network.

TE class mapping must be exactly the same on all routers in a DS-TE domain. It is the responsibility of the operator configure these settings properly as there is no way to automatically check or enforce consistency.

The operator must configure TE tunnel class types and priority levels to form a valid TE class. When the TE class map configuration is changed, tunnels already up are brought down. Tunnels in the down state, can be set up if a valid TE class map is found.

Table 4 list the default TE class and attributes.

Table 4 TE Classes and Priority

TE Class	Class Type	Priority
0	0	7
1	1	7
2	Unused	
3	Unused	
4	0	0
5	1	0
6	Unused	
7	Unused	

Note The default mapping includes four class types.

Flooding

Available bandwidth in all configured bandwidth pools is flooded on the network to calculate accurate constraint paths when a new TE tunnel is configured. Flooding uses IGP protocol extensions and mechanisms to determine when to flood the network with bandwidth.

Flooding Triggers

TE Link Management (TE-Link) notifies IGP for both global pool and sub-pool available bandwidth and maximum bandwidth to flood the network in the following events:

- The periodic timer expires (this does not depend on bandwidth pool type).
- The tunnel origination node has out-of-date information for either available global pool, or sub-pool bandwidth, causing tunnel admission failure at the midpoint.
- Consumed bandwidth crosses user-configured thresholds. The same threshold is used for both global pool and sub-pool. If one bandwidth crosses the threshold, both bandwidths are flooded.

Flooding Thresholds

Flooding frequently can burden a network because all routers must send out and process these updates. Infrequent flooding causes tunnel heads (tunnel-originating nodes) to have out-of-date information, causing tunnel admission to fail at the midpoints.

You can control the frequency of flooding by configuring a set of thresholds. When locked bandwidth (at one or more priority levels) crosses one of these thresholds, flooding is triggered.

Thresholds apply to a percentage of the maximum available bandwidth (the global pool), which is locked, and the percentage of maximum available guaranteed bandwidth (the sub-pool), which is locked. If, for one or more priority levels, either of these percentages crosses a threshold, flooding is triggered.

Note Setting up a global pool TE tunnel can cause the locked bandwidth allocated to sub-pool tunnels to be reduced (and hence to cross a threshold). A sub-pool TE tunnel setup can similarly cause the locked bandwidth for global pool TE tunnels to cross a threshold. Thus, sub-pool TE and global pool TE tunnels can affect each other when flooding is triggered by thresholds.

Fast Reroute

Fast Reroute (FRR) provides link protection to LSPs enabling the traffic carried by LSPs that encounter a failed link to be rerouted around the failure. The reroute decision is controlled locally by the router connected to the failed link. The headend router on the tunnel is notified of the link failure through IGP or through RSVP. When it is notified of a link failure, the headend router attempts to establish a new LSP that bypasses the failure. This provides a path to reestablish links that fail, providing protection to data transfer.

FRR (link or node) is supported over sub-pool tunnels the same way as for regular TE tunnels. In particular, when link protection is activated for a given link, TE tunnels eligible for FRR are redirected into the protection LSP, regardless of whether they are sub-pool or global pool tunnels.

Note The ability to configure FRR on a per-LSP basis makes it possible to provide different levels of fast restoration to tunnels from different bandwidth pools.

You should be aware of the following requirements for the backup tunnel path:

- The backup tunnel must not pass through the element it protects.

- The primary tunnel and a backup tunnel should intersect at least at two points (nodes) on the path: point of local repair (PLR) and merge point (MP). PLR is the headend of the backup tunnel and MP is the tailend of the backup tunnel.

Note When you configure TE tunnel with multiple protection on its path and merge point is the same node for more than one protection, you must configure record-route for that tunnel.

IS-IS IP Fast Reroute Loop-free Alternative

For bandwidth protection, there must be sufficient backup bandwidth available to carry primary tunnel traffic. Use the **ipfrr lfa** command to compute loop-free alternates for all links or neighbors in the event of a link or node failure. To enable node protection on broadcast links, IPRR and bidirectional forwarding detection (BFD) must be enabled on the interface under IS-IS.

Note MPLS FRR and IPFRR cannot be configured on the same interface at the same time.

For information about configuring BFD, see *Cisco IOS XR Interface and Hardware Configuration Guide*.

MPLS-TE and Fast Reroute over Link Bundles

MPLS Traffic Engineering (TE) and Fast Reroute (FRR) are supported over bundle interfaces on the Cisco CRS-1 router only. MPLS-TE/FRR over virtual local area network (VLAN) interfaces is supported on the Cisco CRS-1 router only. Bidirectional forwarding detection (BFD) over VLAN is used as an FRR trigger to obtain more than 50 milliseconds of switchover time on the Cisco CRS-1.

The following link bundle types are supported for MPLS-TE/FRR:

- Over POS link bundles
- Over Ethernet link bundles
- Over VLANs over Ethernet link bundles
- Number of links are limited to 100 for MPLS-TE and FRR.
- VLANs go over any Ethernet interface (for example, GigabitEthernet, TenGigE, FastEthernet, and so forth).

FRR is supported over bundle interfaces in the following ways:

- Uses minimum links as a threshold to trigger FRR over a bundle interface.
- Uses the minimum total available bandwidth as a threshold to trigger FRR.

Ignore Intermediate System-to-Intermediate System Overload Bit Setting in MPLS-TE

The Ignore Intermediate System-to-Intermediate System (IS-IS) Overload Bit Setting in MPLS-TE feature ensures that the RSVP-TE LSPs are not broken because of routers that enabled the IS-IS overload bit.

Note The current implementation does not allow nodes that have indicated an overload situation through the IS-IS overload bit.

Therefore, an overloaded node cannot be used. The IS-IS overload bit limitation is an indication of an overload situation in the IP topology. The feature provides a method to prevent an IS-IS overload condition from affecting MPLS-TE.

Generalized MPLS

Generalized Multiprotocol Label Switching (GMPLS) Traffic Engineering consists of extensions to the MPLS-TE mechanisms to control a variety of device types, including optical switches. When GMPLS-TE is used to control an hierarchical optical network—a network with a core of optical switches surrounded by outer layers of routers—it can provide unified control of devices that have very different hardware capabilities. Other control-plane solutions for such network architectures typically use an overlay model, using separate control-planes to manage the optical core and the routed network, respectively, with little or no knowledge passing between them.

GMPLS-TE protocols and extensions include:

- Resource Reservation Protocol (RSVP) for signaling
- Interior Gateway Protocols (IGP) such as Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) for routing
- Link Management Protocol (LMP) for managing link information

The base protocol definitions for RSVP, OSPF, and IS-IS were previously extended for MPLS-TE to provide circuit mechanisms within packet IP networks. These protocols have been extended for GMPLS-TE.

LMP provides facilities similar to Asynchronous Transfer Mode (ATM) Integrated Local Management Interface (ILMI) and Frame Relay Local Management Interface (LMI). LMP also has features addressing the minimal to nonexistent framing support typical of data links on optical switches.

Optical switches differ from packet and cell devices, in that the data links of optical switches typically can carry only transit traffic. This means that traffic entering an optical switch via one data link is required to leave the switch via a different link. For this reason, a data link that connects two neighboring optical devices cannot exchange control frames between the two devices.

Therefore, optical switches typically have separate frame-capable interfaces for sending and receiving control and management traffic. This type of control is referred to as out-of-band. It contrasts with the in-band control of many non-optical networks where control frames and data frames are intermixed on the same link.

To address this characteristic, the GMPLS protocols have been extended to support out-of-band control.

GMPLS Benefits

GMPLS bridges the Internet Protocol (IP) and photonic layers, thereby making possible interoperable and scalable parallel growth in the IP and photonic dimensions.

This allows for rapid service deployment and operational efficiencies, as well as for increased revenue opportunities. A smooth transition becomes possible from a traditional segregated transport and service overlay model to a more unified peer model.

By streamlining support for multiplexing and switching in a hierarchical fashion, and by utilizing the flexible intelligence of MPLS-TE, optical switching GMPLS becomes very helpful for service providers wanting to manage large volumes of traffic in a cost-efficient manner.

GMPLS Support

GMPLS-TE provides support for:

- Open Shortest Path First (OSPF) for bidirectional TE tunnel
- Frame, lambda, and port (fiber) labels
- Numbered/Unnumbered links
- OSPF extensions–Route computation with optical constraints
- RSVP extensions–Graceful Restart
- Graceful deletion
- LSP hierarchy
- Peer model
- Border model Control plane separation
- Interarea/AS-Verbatim
- BGP4/MPLS
- Restoration–Dynamic path computation
- Control channel manager
- Link summary
- Protection and restoration

GMPLS Protection and Restoration

GMPLS provides protection against failed channels (or links) between two adjacent nodes (span protection) and end-to-end dedicated protection (path protection). After the route is computed, signaling to establish the backup paths is carried out through RSVP-TE or CR-LDP. For span protection, 1+1 or M:N protection schemes are provided by establishing secondary paths through the network. In addition, you can use signaling messages to switch from the failed primary path to the secondary path.

Note Only 1:1 end-to-end path protection is supported.

The restoration of a failed path refers to the dynamic establishment of a backup path. This process requires the dynamic allocation of resources and route calculation. The following restoration methods are described:

- Line restoration—Finds an alternate route at an intermediate node.
- Path restoration—Initiates at the source node to route around a failed path within the path for a specific LSP.

Restoration schemes provide more bandwidth usage, because they do not preallocate any resource for an LSP.

GMPLS combines MPLS-FRR and other types of protection, such as SONET/SDH, wavelength, and so forth.

In addition to SONET alarms in POS links, protection and restoration is also triggered by bidirectional forwarding detection (BFD).

1:1 LSP Protection

When one specific protecting LSP or span protects one specific working LSP or span, 1:1 protection scheme occurs. However, normal traffic is transmitted only over one LSP at a time for working or recovery.

1:1 protection with extra traffic refers to the scheme in which extra traffic is carried over a protecting LSP when the protecting LSP is not being used for the recovery of normal traffic. For example, the protecting LSP is in standby mode. When the protecting LSP is required to recover normal traffic from the failed working LSP, the extra traffic is preempted. Extra traffic is not protected, but it can be restored. Extra traffic is transported using the protected LSP resources.

Shared Mesh Restoration and M:N Path Protection

Both shared mesh restoration and M:N (1:N is more practical) path protection offers sharing for protection resources for multiple working LSPs. For 1:N protection, a specific protecting LSP is dedicated to the protection of up to N working LSPs and spans. Shared mesh is defined as preplanned LSP rerouting, which reduces the restoration resource requirements by allowing multiple restoration LSPs to be initiated from distinct ingress nodes to share common resources, such as links and nodes.

End-to-end Recovery

End-to-end recovery refers to an entire LSP from the source for an ingress router endpoint to the destination for an egress router endpoint.

GMPLS Protection Requirements

The GMPLS protection requirements are specific to the protection scheme that is enabled at the data plane. For example, SONET APS or MPLS-FRR are identified as the data level for GMPLS protection.

GMPLS Prerequisites

The following prerequisites are required to implement GMPLS on Cisco IOS XR software:

- You must be in a user group associated with a task group that includes the proper task IDs for GMPLS commands.
- A router that runs Cisco IOS XR software.
- Installation of the Cisco IOS XR software mini-image on the router.

Flexible Name-based Tunnel Constraints

MPLS-TE Flexible Name-based Tunnel Constraints provides a simplified and more flexible means of configuring link attributes and path affinities to compute paths for MPLS-TE tunnels.

In the traditional TE scheme, links are configured with attribute-flags that are flooded with TE link-state parameters using Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF).

MPLS-TE Flexible Name-based Tunnel Constraints lets you assign, or map, up to 32 color names for affinity and attribute-flag attributes instead of 32-bit hexadecimal numbers. After mappings are defined, the attributes can be referred to by the corresponding color name in the command-line interface (CLI).

Furthermore, you can define constraints using **include**, **include-strict**, **exclude**, and **exclude-all** arguments, where each statement can contain up to 10 colors, and define include constraints in both loose and strict sense.

Note You can configure affinity constraints using attribute flags or the Flexible Name Based Tunnel Constraints scheme; however, when configurations for both schemes exist, only the configuration pertaining to the new scheme is applied.

MPLS Traffic Engineering Interarea Tunneling

This section describes the following new extensions of MPLS-TE:

- Interarea Support, page MPC-108
- Multiarea Support, page MPC-109
- Loose Hop Expansion, page MPC-109
- Loose Hop Reoptimization, page MPC-110
- Fast Reroute Node Protection, page MPC-110

Interarea Support

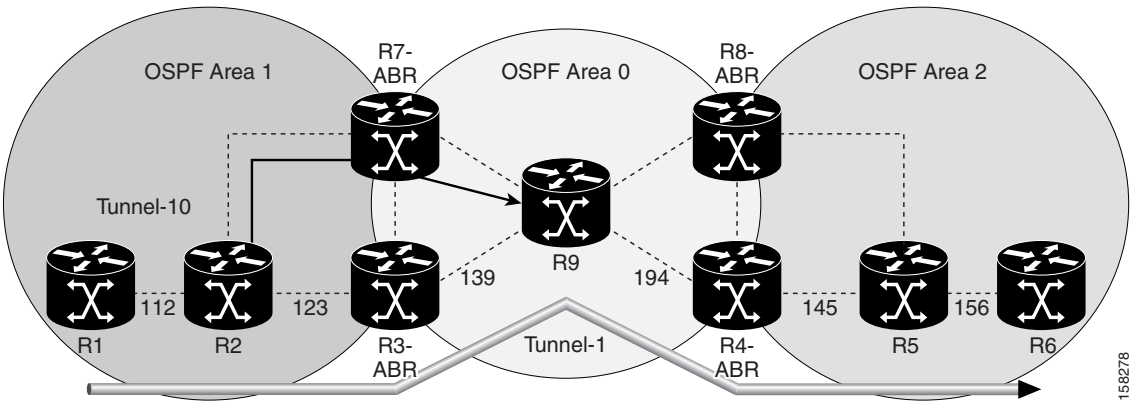
The MPLS-TE interarea tunneling feature allows you to establish TE tunnels spanning multiple Interior Gateway Protocol (IGP) areas and levels, thereby eliminating the requirement that headend and tailend routers reside in a single area.

Interarea support allows the configuration of a TE LSP that spans multiple areas, where its headend and tailend label switched routers (LSRs) reside in different IGP areas.)

Multiarea and Interarea TE are required by the customers running multiple IGP area backbones (primarily for scalability reasons). This lets you limit the amount of flooded information, reduces the SPF duration, and lessens the impact of a link or node failure within an area, particularly with large WAN backbones split in multiple areas.

Figure 10 shows a typical interarea TE network.

Figure 10 Interarea (OSPF) TE Network Diagram



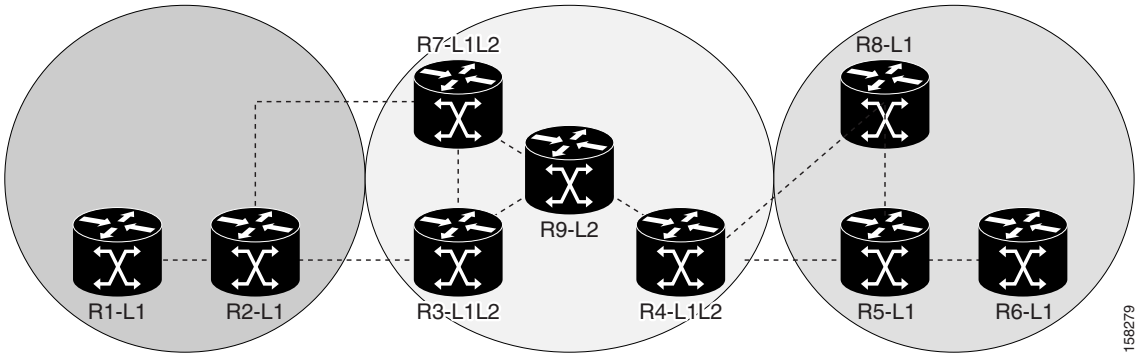
Multiarea Support

Multiarea support allows an ABR LSR to support MPLS-TE in more than one IGP area. A TE LSP will still be confined to a single area.

Multiarea and Interarea TE are required when you run multiple IGP area backbones. The Multiarea and Interarea TE allows you to:

- Limit the volume of flooded information.
- Reduce the SPF duration.
- Decrease the impact of a link or node failure within an area.

Figure 11 Interlevel (IS-IS) TE Network



As shown in Figure 11, R2, R3, R7, and R4 maintain two databases for routing and TE information. For example, R3 has TE topology information related to R2, flooded through Level-1 IS-IS LSPs plus the TE topology information related to R4, R9, and R7, flooded as Level 2 IS-IS Link State PDUs (LSPs) (plus, its own IS-IS LSP).

Note You can configure multiple areas within an IS-IS Level 1. This is transparent to TE. TE has topology information about the IS-IS level, but not the area ID.

Loose Hop Expansion

Loose hop optimization allows the reoptimization of tunnels spanning multiple areas and solves the problem which occurs when an MPLS-TE LSP traverses hops that are not in the LSP's headend's OSPF area and IS-IS level.

Interarea MPLS-TE allows you configure an interarea TE LSP by specifying a loose source route of ABRs along the path. It is the then the responsibility of the ABR (having a complete view of both areas) to find a path obeying the TE LSP constraints within the next area to reach the next hop ABR (as specified on the headend). The same operation is performed by the last ABR connected to the tailend area to reach the tailend LSR.

You must be aware of the following considerations when using loose hop optimization:

- You must specify the router ID of the ABR node (as opposed to a link address on the ABR).

- When multiarea is deployed in a network that contains subareas, you must enable MPLS-TE in the subarea for TE to find a path when loose hop is specified.
- You must specify the reachable explicit path for the interarea tunnel.

Loose Hop Reoptimization

Loose hop reoptimization allows the reoptimization of the tunnels spanning multiple areas and solves the problem which occurs when an MPLS-TE headend does not have visibility into other IGP areas.

Whenever the headend attempts to reoptimize a tunnel, it tries to find a better path to the ABR in the headend area. If a better path is found then the headend initiates the setup of a new LSP. In case a suitable path is not found in the headend area, the headend initiates a querying message. The purpose of this message is to query the ABRs in the areas other than the headend area to check if there exist any better paths in those areas. The purpose of this message is to query the ABRs in the areas other than the headend area, to check if a better path exists. If a better path does not exist, ABR forwards the query to the next router downstream. Alternatively, if better path is found, ABR responds with a special Path Error to the headend to indicate the existence of a better path outside the headend area. Upon receiving the Path Error that indicates the existence of a better path, the headend router initiates the reoptimization.

ABR Node Protection

Since one IGP area does not have visibility into another IGP area, it is not possible to assign backup to protect ABR node. To overcome this problem, node ID sub-object is added into the record route object of the primary tunnel so that at a PLR node, backup destination address can be checked against primary tunnel record-route object and assign a backup tunnel.

Fast Reroute Node Protection

If a link failure occurs within an area, the upstream router directly connected to the failed link generates an RSVP path error message to the headend. As a response to the message, the headend sends an RSVP path tear message and the corresponding path option is marked as invalid for a specified period and the next path-option (if any) is evaluated.

To retry the ABR immediately, a second path option (identical to the first one) should be configured. Alternatively, the retry period (path-option hold-down, 2 minutes by default) can be tuned to achieve a faster retry.

MPLS-TE Forwarding Adjacency

The MPLS-TE Forwarding Adjacency feature allows a network administrator to handle a traffic engineering, label-switched path (LSP) tunnel as a link in an Interior Gateway Protocol (IGP) network based on the Shortest Path First (SPF) algorithm. A forwarding adjacency can be created between routers regardless of their location in the network.

MPLS-TE Forwarding Adjacency Benefits

TE tunnel interfaces are advertised in the IGP network just like any other links. Routers can then use these advertisements in their IGPs to compute the SPF even if they are not the head end of any TE tunnels.

MPLS-TE Forwarding Adjacency Restrictions

The following restrictions are listed for the MPLS-TE Forwarding Adjacency feature:

- Using the MPLS-TE Forwarding Adjacency feature increases the size of the IGP database by advertising a TE tunnel as a link.
- The MPLS-TE Forwarding Adjacency feature is supported by Intermediate System-to-Intermediate System (IS-IS).
- When the MPLS-TE Forwarding Adjacency feature is enabled on a TE tunnel, the link is advertised in the IGP network as a Type-Length-Value (TLV) 22 without any TE sub-TLV.
- MPLS-TE forwarding adjacency tunnels must be configured bidirectionally.

MPLS-TE Forwarding Adjacency Prerequisites

Your network must support the following features before enabling the MPLS -TE Forwarding Adjacency feature:

- MPLS
- IP Cisco Express Forwarding
- Intermediate System-to-Intermediate System (IS-IS)
- OSPF

Unequal Load Balancing

Unequal load balancing permits the routing of unequal proportions of traffic through tunnels to a common destination. Load shares on tunnels to the same destination are determined by TE from the tunnel configuration and passed via the MPLS Label Switching Database (LSD) to the Forwarding Information Base (FIB).

Note Load share values are renormalised by the FIB using values suitable for use by the forwarding code; the exact traffic ratios observed may not, therefore, exactly mirror the configured traffic ratios. This effect is more pronounced if there are many parallel tunnels to a destination, or if the load shares assigned to those tunnels are very different. The exact renormalization algorithm used is platform-dependent.

There are two ways to configure load balancing:

- Explicit configuration—Using this method, load shares are explicitly configured on each tunnel.
- Bandwidth configuration—If a tunnel is not configured with load-sharing parameters, the tunnel bandwidth and load-share values are considered equivalent for load-share calculations between tunnels, and a direct comparison between bandwidth and load-share configuration values is calculated.

Note Load shares are not dependent on any configuration other than the load share and bandwidth configured on the tunnel and the state of the global configuration switch.

Path Computation Element

Path Computation Element (PCE) solves the specific issue of inter-domain path computation for MPLS-TE LSPs, when the head-end router does not possess full network topology information (for example, when the head-end and tail-end routers of an LSP reside in different IGP areas).

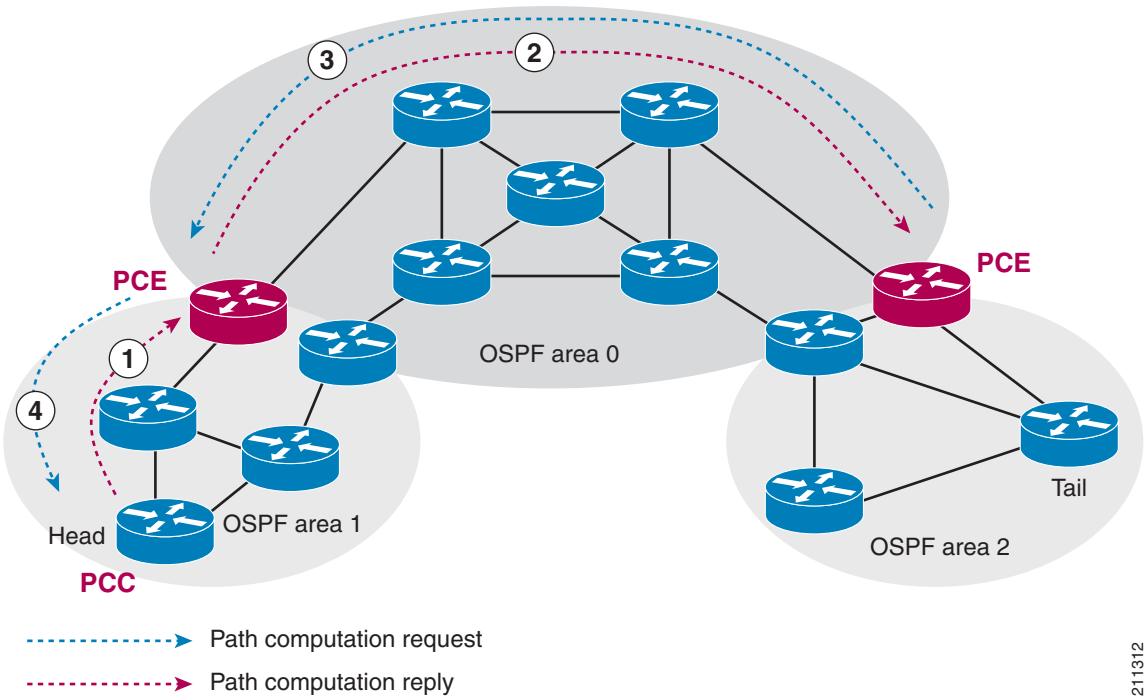
PCE uses area border routers (ABRs) to compute a TE LSP spanning multiple IGP areas as well as computation of Inter-AS TE LSP.

PCE is usually used to define an overall architecture, which is made of several components, as follows:

- Path Computation Element (PCE)—Represents a software module (which can be a component or application) that enables the router to compute paths applying a set of constraints between any pair of nodes within the router's TE topology database. PCEs are discovered through IGP.
- Path Computation Client (PCC)—Represents a software module running on a router that is capable of sending and receiving path computation requests and responses to and from PCEs. The PCC is typically an LSR (Label Switching Router).
- PCC-PCE communication protocol (PCEP)—Specifies that PCEP is a TCP-based protocol defined by the IETF PCE WG, and defines a set of messages and objects used to manage PCEP sessions and to request and send paths for multi-domain TE LSPs. PCEP is used for communication between PCC and PCE (as well as between two PCEs) and employs IGP extensions to dynamically discover PCE.

Figure 12 shows a typical PCE implementation.

Figure 12 Path Computation Element Network Diagram



Path computation elements provides support for the following message types and objects:

- Message types: Open, PCReq, PCRep, PCErr, Close
- Objects: OPEN, CLOSE, RP, END-POINT, LSPA, BANDWIDTH, METRIC and NO-PATH

Policy-based Tunnel Selection

These topics provide information about policy-based tunnel selection (PBTS):

- Policy-based Tunnel Selection Overview, page MPC-113
- Policy-based Tunnel Selection Functions, page MPC-113
- PBTS with Dynamic Tunnel Selection, page MPC-114
- Restrictions, page MPC-114

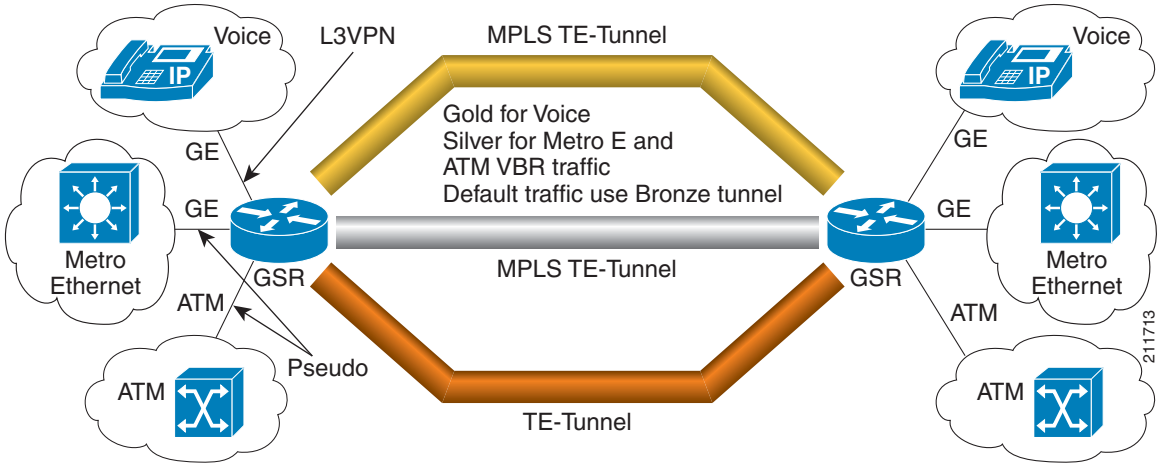
Policy-based Tunnel Selection Overview

PBTS provides a mechanism that lets you direct traffic into specific TE tunnels based on different criteria. PBTS will benefit Internet service providers (ISPs) who carry voice and data traffic through their MPLS and MPLS/VPN networks, who want to route this traffic to provide optimized voice service.

PBTS works by selecting tunnels based on the classification criteria of the incoming packets, which are based on the IP precedence, EXP, or TOS field in the packet. When there are no paths with a default class configured, this traffic is forwarded using the paths with the lowest class value.

Figure 13 illustrates a PBTS implementation.

Figure 13 Policy-based Tunnel Selection Implementation



Policy-based Tunnel Selection Functions

The following PBTS functions are supported on the Cisco CRS-1 router and the Cisco XR 12000 Series Router:

- IPv4 traffic arrives unlabeled on the VRF interface and the non-VRF interface.
- MPLS traffic is supported on the VRF interface and the non-VRF interface.
- Load balancing across multiple TE tunnels with the same traffic class attribute is supported.
- The selected TE tunnels are used to service the lowest tunnel class as default tunnels.
- LDP over TE tunnel and single-hop TE tunnel are supported.

The following PBTS functions are supported only on the Cisco XR 12000 Series Router:

- L2VPN preferred path selection lets traffic be directed to a particular TE tunnel.
- Both Interior Gateway Protocol (IGP) and Label Distribution Protocol (LDP) paths are used as the default path for all traffic that belongs to a class that is not configured on the TE tunnels.
- According to the quality-of-service (QoS) policy, tunnel selection is based on the outgoing experimental (EXP) value and the remarked EXP value.
- IPv6 traffic for both 6VPE and 6PE scenarios are supported.

PBTS with Dynamic Tunnel Selection

Note This feature is supported only on the Cisco XR 12000 Series Router.

Dynamic tunnel selection, which is based on class-of-service-based tunnel selection (CBTS), uses post-QoS EXP to select the tunnel. The TE tunnel contains a class attribute that is based on CoS or EXP. Traffic is forwarded on the TE tunnels based on the class attribute. For the balancing group, the traffic can be load-balanced among the tunnels of the same class. The default path is a LDP LSP or a default tunnel.

Restrictions

When implementing PBTS, the following restrictions are listed:

- When you enable QoS EXP remarking on an interface, the EXP value is used to determine the egress tunnel interface, not the incoming EXP value.
- Egress-side remarking does not affect PBTS tunnel selection.
- For information about the PBTS default path behavior and the **mpls traffic-eng igp-intact (OSPF)** command or **mpls traffic-eng igp-intact (IS-IS)** command, refer to *Cisco IOS XR Routing Command Reference*.

How to Implement Traffic Engineering on Cisco IOS XR Software

Traffic engineering requires coordination among several global neighbor routers, creating traffic engineering tunnels, setting up forwarding across traffic engineering tunnels, setting up FRR, and creating differential service.

This section explains the following procedures:

- Building MPLS-TE Topology, page MPC-115
- Creating an MPLS-TE Tunnel, page MPC-119
- Configuring Forwarding over the MPLS-TE Tunnel, page MPC-121
- Protecting MPLS Tunnels with Fast Reroute, page MPC-123
- Configuring a Prestandard Diff-Serv TE Tunnel, page MPC-127
- Configuring an IETF Diff-Serv TE Tunnel Using RDM, page MPC-129
- Configuring an IETF Diff-Serv TE Tunnel Using MAM, page MPC-131
- Configuring the Ignore Integrated Intermediate System-to-Intermediate System Overload Bit Setting in MPLS-TE, page MPC-134
- Configuring GMPLS on Cisco IOS XR Software, page MPC-135
- Configuring Flexible Name-based Tunnel Constraints, page MPC-165
- Configuring IS-IS to Flood MPLS-TE Link Information, page MPC-170
- Configuring an OSPF Area of MPLS-TE, page MPC-172
- Configuring Explicit Paths with ABRs Configured as Loose Addresses, page MPC-174
- Configuring MPLS-TE Forwarding Adjacency, page MPC-175
- Configuring Unequal Load Balancing, page MPC-177
- Configuring a Path Computation Client and Element, page MPC-180
- Configuring Policy-based Tunnel Selection, page MPC-185

Building MPLS-TE Topology

Perform this task to configure MPLS-TE topology (required for traffic engineering tunnel operations).

Building the MPLS-TE topology is accomplished by performing the following basic steps:

- Enabling MPLS-TE on the port interface.
- Enabling RSVP on the port interface.
- Enabling an IGP such as OSPF or IS-IS for MPLS-TE.

Prerequisites

The following prerequisites are required to build the MPLS-TE topology:

- You must have a router ID for the neighboring router.

- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.
- If you are going to use nondefault holdtime or intervals, you must decide the values to which they are set.

SUMMARY STEPS

1. **configure**
2. **router-id** {*interface-id* | *ip-address*}
3. **mpls traffic-eng**
4. **interface** *type interface-id*
5. **exit**
6. **router ospf** *process-name*
7. **router-id** {*interface-id* | *ip-address*}
8. **area** *area-id*
9. **interface** *type interface-id*
10. **interface** *interface-id*
11. **exit**
12. **mpls traffic-eng router-id**
13. **area** *area-id*
14. **exit**
15. **rsvp interface** *type interface-id*
16. **bandwidth** *bandwidth*
17. **end**
    or
    commit
18. **show mpls traffic topology**
19. **show mpls traffic-eng link-management advertisements**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**router id** {*interface-id* | *ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# router id loopback0`	Specifies the global router ID of the local node. • The router ID can be specified with an interface name or an IP address. By default, MPLS uses the global router ID.

	Command or Action	Purpose
Step 3	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters the MPLS-TE configuration mode.

Step 4	**interface** *type interface-id*
	
	`Example:`
	`RP/0/RP0/CPU0:router(config-mpls-te)# interface POS0/6/0/0`
		Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node.

Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# exit`	Exits the current configuration mode.
Step 6	**router ospf** process-name **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enters a name for the OSPF process.
Step 7	**router-id** {*interface-id* | *ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-router)# router-id 192.168.25.66`	Configures a router ID for the OSPF process using an IP address.
Step 8	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-router)# area 0`	Configures an area for the OSPF process. • Backbone areas have an area ID of 0. • Non-backbone areas have a non-zero area ID.
Step 9	**interface** *type interface-id* ***Example:*** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface pos 0/6/0/0`	Configures one or more interfaces for the area configured in Step 8.
Step 10	**interface** *interface-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface loopback 0`	Enables IGP on the loopback0 MPLS router ID.
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# exit`	Exits the current configuration mode.
Step 12	**mpls traffic-eng router-id loopback 0** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng router-id loopback 0`	Sets the MPLS-TE loopback interface.

	Command or Action	Purpose
Step 13	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Sets the MPLS-TE area.
Step 14	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# exit`	Exits the current configuration mode.

Step 15	**rsvp interface** *type interface-id*
	
	`Example:`
	`RP/0/RP0/CPU0:router(config)# rsvp interface Bundle-POS 500`
		Enters RSVP interface configuration mode and enables RSVP on a particular interface on the originating node (in this case, on the Bundle-POS interface 500).

Step 16	**bandwidth** *bandwidth* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 100`	Sets the reserved RSVP bandwidth available on this interface. **Note** Physical interface bandwidth is not used by MPLS-TE.

Step 17	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

	Command or Action	Purpose
Step 18	**show mpls traffic-eng topology** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng topology`	(Optional) Verifies the traffic engineering topology.
Step 19	**show mpls traffic-eng link-management advertisements** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng link-management advertisements`	(Optional) Displays all the link-management advertisements for the links on this node.

Creating an MPLS-TE Tunnel

Creating an MPLS-TE tunnel is a process of customizing the traffic engineering to fit your network topology.

Perform this task to create an MPLS-TE tunnel after you have built the traffic engineering topology (see "Building MPLS-TE Topology" section on page 115).

Prerequisites

The following prerequisites are required to create an MPLS-TE tunnel:

- You must have a router ID for the neighboring router.
- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.
- If you are going to use nondefault holdtime or intervals, you must decide the values to which they are set.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*
3. **destination** *ip-address*
4. **ipv4 unnumbered loopback** *number*
5. **path-option** *path-id* **dynamic**
6. **signaled bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*}
7. **end**
   or
   commit
8. **show mpls traffic-eng tunnels**
9. **show ipv4 interface brief**
10. **show mpls traffic-eng link-management admission-control**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te 1`	Enters MPLS-TE interface configuration mode.
Step 3	**destination** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125`	Assigns a destination address on the new tunnel. • The destination address is the remote node's MPLS-TE router ID.
Step 4	**ipv4 unnumbered loopback** *number* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered loopback 0`	Assigns a source address so that forwarding can be performed on the new tunnel.
Step 5	**path-option** *path-id* **dynamic** **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option 1 dynamic`	Sets the path option to dynamic and also assigns the path ID.
Step 6	**signaled bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-if)# signaled bandwidth 100`	Sets the CT0 bandwidth required on this interface. Because the default tunnel priority is 7, tunnels use the default TE class map (namely, class-type 1, priority 7).

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show mpls traffic-eng tunnels** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels`	(Optional) Verifies that the tunnel is connected (in the UP state) and displays all configured TE tunnels.
Step 9	**show ipv4 interface brief** **Example:** `RP/0/RP0/CPU0:router# show ipv4 interface brief`	(Optional) Displays all TE tunnel interfaces.
Step 10	**show mpls traffic-eng link-management admission-control** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng link-management admission-control`	(Optional) Displays all the tunnels on this node.

Configuring Forwarding over the MPLS-TE Tunnel

Perform this task to configure forwarding over the MPLS-TE tunnel created in the previous task (see "Creating an MPLS-TE Tunnel" section on page 119).

This procedure allows MPLS packets to be forwarded on the link between network neighbors.

Prerequisites

The following prerequisites are required to configure forwarding over the MPLS-TE tunnel:

- You must have a router ID for the neighboring router.
- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*
3. **ipv4 unnumbered loopback** *number*
4. **autoroute announce**
5. **exit**
6. **router static address-family ipv4 unicast** *prefix mask ip-address interface type*
7. **end**
   or
   commit
8. **ping** {*ip-address* | *hostname*}
9. **show mpls traffic-eng autoroute**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te 1`	Enters MPLS-TE interface configuration mode.
Step 3	**ipv4 unnumbered loopback** *number* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered loopback 0`	Assigns a source address so that forwarding can be performed on the new tunnel.
Step 4	**autoroute announce** **Example:** `RP/0/RP0/CPU0:router(config-if)# autoroute announce`	Enables messages that notify the neighbor nodes about the routes that are forwarding.

	Command or Action	Purpose
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if)# exit`	Exits the current configuration mode.
Step 6	**router static address-family ipv4 unicast** *prefix mask ip-address interface type* **Example:** `RP/0/RP0/CPU0:router(config)# router static address-family ipv4 unicast 2.2.2.2/32 tunnel-te 1`	(Optional) Enables a route using IP version 4 addressing, identifies the destination address and the tunnel where forwarding is enabled. • This configuration is used for static routes when autoroute announce config is not used.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**ping** {*ip-address* | *hostname*} **Example:** `RP/0/RP0/CPU0:router# ping 192.168.12.52`	(Optional) Checks for connectivity to a particular IP address or host name.
Step 9	**show mpls traffic-eng autoroute** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng autoroute`	(Optional) Verifies forwarding by displaying what is advertised to IGP for the TE tunnel.

Protecting MPLS Tunnels with Fast Reroute

Perform this task to protect MPLS-TE tunnels, as created in the previous task (see "Configuring Forwarding over the MPLS-TE Tunnel" section on page 121).

Note Although this task is similar to the previous task, its importance makes it necessary to present as part of the tasks required for traffic engineering on Cisco IOS XR software.

Prerequisites

The following prerequisites are required to protect MPLS-TE tunnels:

- You must have a router ID for the neighboring router.
- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.
- You must first configure a primary and a backup tunnel (see "Creating an MPLS-TE Tunnel" section on page 119).

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*
3. **fast-reroute**
4. **exit**
5. **mpls traffic-eng interface** *type interface-id*
6. **backup-path tunnel-te** *tunnel-number*
7. **exit**
8. **interface tunnel-te** *tunnel-id*
9. **backup-bw** {*bandwidth* | **sub-pool** {*bandwidth* | **unlimited**} | **global-pool** {*bandwidth* | **unlimited**}}
10. **ipv4 unnumbered loopback** *number*
11. **path-option** *path-id* **explicit name** *explicit-path-name*
12. **destination** *A.B.C.D*
13. **end**
    or
    commit
14. **show mpls traffic-eng tunnels backup**
15. **show mpls traffic-eng tunnels protection**
16. **show mpls traffic-eng fast-reroute database**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te1`	Enters MPLS-TE interface configuration mode.
Step 3	**fast-reroute** **Example:** `RP/0/RP0/CPU0:router(config-if)# fast-reroute`	Enables fast reroute.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if)# exit`	Exits the current configuration mode.
Step 5	**mpls traffic-eng interface** *type interface-id* ***Example:*** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng interface pos0/6/0/0`	Enters the MPLS-TE configuration mode, and enables traffic engineering on a particular interface on the originating node.
Step 6	**backup-path tunnel-te** *tunnel-number* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# backup-path tunnel-te 2`	Sets the backup path to the backup tunnel.
Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if)# exit`	Exits the current configuration mode.
Step 8	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te2`	Enters MPLS-TE interface configuration mode.
Step 9	**backup-bw** {*bandwidth* | **sub-pool** {*bandwidth* | **unlimited**} | **global-pool** {*bandwidth* | **unlimited**}} **Example:** `RP/0/RP0/CPU0:router(config-if)# backup-bw global-pool 5000`	Sets the CT0 bandwidth required on this interface. **Note** Because the default tunnel priority is 7, tunnels use the default TE class map.

	Command or Action	Purpose
Step 10	**ipv4 unnumbered loopback** *number* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered loopback 0`	Assigns a source address to set up forwarding on the new tunnel.
Step 11	**path-option** *path-id* **explicit name** *explicit-path-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option 1 explicit name backup-path`	Sets the path option to explicit with a given name (previously configured) and assigns the path ID.
Step 12	**destination** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125`	Assigns a destination address on the new tunnel. • The destination address is the remote node's MPLS-TE router ID. • The destination address is the merge point between backup and protected tunnels. **Note** When you configure TE tunnel with multiple protection on its path and merge point is the same node for more than one protection, you must configure record-route for that tunnel.

Step 13	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 14	**show mpls traffic-eng tunnels backup** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels backup`	(Optional) Displays the backup tunnel information.

	Command or Action	Purpose
Step 15	**show mpls traffic-eng tunnels protection** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels protection`	(Optional) Displays the tunnel protection information.
Step 16	**show mpls traffic-eng fast-reroute database** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng fast-reroute database`	(Optional) Displays the protected tunnel state (for example, the tunnel's current ready or active state).

Configuring a Prestandard Diff-Serv TE Tunnel

Perform this task to configure a Prestandard Diff-Serv TE tunnel.

Prerequisites

The following prerequisites are required to configure a Prestandard Diff-Serv TE tunnel:

- You must have a router ID for the neighboring router.
- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** *type interface-id*
3. **bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}]
4. **exit**
5. **interface tunnel-te** *number*
6. **signaled bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**rsvp interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# rsvp interface pos0/6/0/0`	Enters RSVP configuration mode and selects an RSVP interface.
Step 3	**bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}] **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 100 150 sub-pool 50`	Sets the reserved RSVP bandwidth available on this interface. **Note** Physical interface bandwidth is not used by MPLS-TE.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# exit`	Exits the current configuration mode.
Step 5	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te2`	Enters MPLS-TE interface configuration mode.

	Command or Action	Purpose
Step 6	**signaled bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-if)# bandwidth sub-pool 10`	Sets the bandwidth required on this interface. Because the default tunnel priority is 7, tunnels use the default TE class map (namely, class-type 1, priority 7).

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an IETF Diff-Serv TE Tunnel Using RDM

Perform this task to create an IETF mode differentiated services traffic engineering tunnel using RDM.

Prerequisites

The following prerequisites are required to create an IETF mode differentiated services traffic engineering tunnel using RDM:

- You must have a router ID for the neighboring router.
- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** *type interface-id*
3. **bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}]
4. **exit**

5. **mpls traffic-eng**
6. **ds-te mode ietf**
7. **exit**
8. **interface tunnel-te** *number*
9. **signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*}
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**rsvp interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# rsvp interface pos0/6/0/0`	Enters RSVP configuration mode and selects an RSVP interface.
Step 3	**bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}] **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth rdm 100 150`	Sets the reserved RSVP bandwidth available on this interface. **Note** Physical interface bandwidth is not used by MPLS-TE.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# exit`	Exits the current configuration mode.
Step 5	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 6	**ds-te mode ietf** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# ds-te mode ietf`	Enables IETF DS-TE mode and default TE class map. Configure IETF DS-TE mode on all network nodes.
Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# exit`	Exits the current configuration mode.

	Command or Action	Purpose
Step 8	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te4`	Enters MPLS-TE interface configuration mode.
Step 9	**signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 10 class-type 1`	Configures the bandwidth required for an MPLS TE tunnel. Because the default tunnel priority is 7, tunnels use the default TE class map (namely, class-type 1, priority 7).

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an IETF Diff-Serv TE Tunnel Using MAM

Perform this task to configure an IETF mode differentiated services traffic engineering tunnel using the Maximum Allocation Model (MAM) bandwidth constraint model.

Prerequisites

The following prerequisites are required to configure an IETF mode differentiated services traffic engineering tunnel using the MAM bandwidth constraint model:

- You must have a router ID for the neighboring router.
- A stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** *type interface-id*
3. **bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}]
4. **exit**
5. **mpls traffic-eng**
6. **ds-te mode ietf**
7. **ds-te bc-model mam**
8. **exit**
9. **interface tunnel-te** *number*
10. **signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*}
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**rsvp interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# rsvp interface pos0/6/0/0`	Enters RSVP configuration mode and selects the RSVP interface.
Step 3	**bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}] **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth mam max-reservable-bw 400 bc0 300 bc1 200`	Sets the reserved RSVP bandwidth available on this interface. **Note** Physical interface bandwidth is not used by MPLS-TE.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# exit`	Exits the current configuration mode.
Step 5	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.

	Command or Action	Purpose
Step 6	**ds-te mode ietf** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# ds-te mode ietf`	Enables IETF DS-TE mode and default TE class map. Configure IETF DS-TE mode on all nodes in the network.
Step 7	**ds-te bc-model mam** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# ds-te bc-model mam`	Enables the MAM bandwidth constraint model globally.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# exit`	Exits the current configuration mode.
Step 9	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te4`	Enters MPLS-TE interface configuration mode.
Step 10	**signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 10 class-type 1`	Configures the bandwidth required for an MPLS TE tunnel. Because the default tunnel priority is 7, tunnels use the default TE class map (namely, class-type 1, priority 7).

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Ignore Integrated Intermediate System-to-Intermediate System Overload Bit Setting in MPLS-TE

Perform this task to configure an overload node avoidance to MPLS-TE. When the overload bit is enabled, tunnels are brought down when the overload node is found in the tunnel path.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng path-selection ignore overload**
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**mpls traffic-eng path-selection ignore overload** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng path-selection ignore overload`	Ignores the Intermediate System-to-Intermediate System (IS-IS) overload bit setting for MPLS-TE.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring GMPLS on Cisco IOS XR Software

To fully configure GMPLS, you must complete the following high-level tasks in order:

- Configuring IPCC Control Channel Information, page MPC-136
- Configuring Local and Remote TE Links, page MPC-139
- Configuring Numbered and Unnumbered Optical TE Tunnels, page MPC-154
- Configuring LSP Hierarchy, page MPC-159
- Configuring Border Control Model, page MPC-160
- Configuring Path Protection, page MPC-161

Note These high-level tasks are broken down into, in some cases, several subtasks.

Configuring IPCC Control Channel Information

This section includes the following subtasks:

- Configuring Router IDs, page MPC-136
- Configuring OSPF over IPCC, page MPC-138

Note You must configure each subtask on both the headend and tailend router.

Configuring Router IDs

Perform this task to configure the router ID for the headend and tailend routers.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id*
3. **ipv4 address** *A.B.C.D/prefix*
4. **exit**
5. **configure**
6. **router-id** {*interface-id* | *ip-address*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface POS0/6/0/0`	Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node.

	Command or Action	Purpose
Step 3	**ipv4 address** *A.B.C.D/prefix* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.0.0.0`	Specifies a primary or secondary IPv4 address for an interface. • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and a number (prefix length). The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address compose the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# exit`	Exits the current configuration mode.
Step 5	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Re-enters global configuration mode.
Step 6	**router id** {*interface-id* | *ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-if)# router id loopback 0`	Specifies the global router ID of the local node. • The router ID can be specified with an interface name or an IP address. By default, MPLS uses the global router ID.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring OSPF over IPCC

Perform this task to configure OSPF over IPCC on both the headend and tailend routers.

The IGP interface ID is configured for control network, specifically for the signaling plane in the optical domain.

Note IPCC support is restricted to routed, out-of-fiber, and out-of-band.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **area** *area-id*
4. **interface** *interface-id*
5. **exit**
6. **mpls traffic-eng router-id** {*interface-id* | *ip-address*}
7. **mpls traffic-eng area** *area-id*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Configures OSPF routing and assigns a process name.
Step 3	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Configures an area ID for the OSPF process (either as a decimal value or IP address): • Backbone areas have an area ID of 0. • Non-backbone areas have a nonzero area ID.
Step 4	**interface** *interface-id* **Example:** `RP/0/RP0/CPU0:router((config-ospf-ar)# interface Loopback 0`	Enables IGP on the interface. **Note** Use this command to configure any interface included in the control network.

	Command or Action	Purpose
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit`	Exits the current configuration mode.
Step 6	**mpls traffic-eng router-id** {*interface-id* | *ip-address*} **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng router id 192.168.25.66`	Configures a router ID for the OSPF process using an IP address.
Step 7	**mpls traffic-eng area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng area 0`	Configures the MPLS-TE area.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Local and Remote TE Links

The subtasks in this section describe how to configure local and remote MPLS-TE link parameters for numbered and unnumbered TE links on both headend and tailend routers.

This section includes the following subtasks:

- Configuring Numbered and Unnumbered Links, page MPC-140
- Configuring Local Reservable Bandwidth, page MPC-142
- Configuring Local Switching Capability Descriptors, page MPC-143
- Configuring Persistent Interface Index, page MPC-145

- Enabling LMP Message Exchange, page MPC-146
- Configuring Remote TE Link Adjacency Information for Numbered Links, page MPC-150

Configuring Numbered and Unnumbered Links

Perform this task to configure numbered and unnumbered links.

Note Unnumbered TE links use the IP address of the associated interface.

SUMMARY OF STEPS

1. **configure**
2. **interface** *type interface-id*
3. **ipv4 address** *ipv4-address mask*
   or
   ipv4 unnumbered interface *type interface-id*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface POS0/6/0/0`	Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node.
Step 3	**ipv4 address** *ipv4-address mask* **or** **ipv4 unnumbered interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.0.0.0`	Specifies a primary or secondary IPv4 address for an interface. • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and a number (prefix length). The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address compose the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash. or • Enables IPv4 processing on a point-to-point interface without assigning an explicit IPv4 address to that interface. **Note** If you configured a unnumbered GigE interface in Step 2 and selected the ipv4 unnumbered interface type option in this step, you must enter the **ipv4 point-to-point** command to configure point-to-point interface mode.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Local Reservable Bandwidth

Perform this task to configure the local reservable bandwidth for the data bearer channels.

SUMMARY STEPS

1. **configure**
2. **rsvp interface** *type interface-id*
3. **bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**rsvp interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# rsvp interface POS0/6/0/0`	Enters RSVP configuration mode and selects an RSVP interface ID.

	Command or Action	Purpose
Step 3	**bandwidth** [*0 - 4294967295*] [**bc0**] [*global-pool*] [**mam** {*0-4294967295* | *max-reservable-bandwidth*}] [**rdm** {*0-4294967295* | **bc0** | *global-pool*}] **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 2488320 2488320`	Sets the reserved RSVP bandwidth available on this interface. **Note** MPLS-TE can use only the amount of bandwidth specified using this command on the configured interface.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Local Switching Capability Descriptors

Perform this task to configure the local switching capability descriptor.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **interface** *type interface-id*
4. **flooding-igp ospf** *instance-id* **area** *area-id*
5. **switching key** *cap*
6. **encoding {sonet/sdh | ethernet}**
7. **capability** {**psc1 | lsc | fsc**}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 3	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# interface POS0/6/0/0`	Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node.
Step 4	**flooding-igp ospf** *instance-id* **area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# flooding-igp ospf 0 1`	Specifies the IGP OSPF interface ID and area where the TE links are to be flooded.
Step 5	**switching key** *cap* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# switching key 0`	Specifies the switching configuration for the interface and enters switching key submode where you will configure encoding and capability. **Note** The recommended switch key value is 0.
Step 6	**encoding** {**sonet/sdh** | **ethernet**} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# encoding ethernet`	Specifies the interface encoding type, as follows: • sonet/sdh, or POS • ethernet, or GigE

	Command or Action	Purpose
Step 7	**capability** {**psc1** | **lsc** | **fsc**} **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# capability psc1`	Specifies the interface switching capability type. The recommended switch capability type is **psc1**.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Persistent Interface Index

Perform this task to preserve the LMP interface index across all interfaces on the router.

SUMMARY STEPS

1. **configure**
2. **snmp-server ifindex persist**
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**snmp-server ifindex persist** **Example:** `RP/0/RP0/CPU0:router(config)# snmp-server ifindex persist`	Enables ifindex persistence globally on all Simple Network Management Protocol (SNMP) interfaces.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling LMP Message Exchange

Perform the following task to enable LMP message exchange.

LMP is enabled by default. You can disable LMP on a per neighbor basis using the **lmp static** command in LMP protocol neighbor submode.

Note LMP is recommended unless the peer optical device does not support LMP (in which case it is necessary to disable it at both ends).

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **lmp neighbor** *name*

4. **ipcc routed**
5. **remote node-id** *node-id*
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 3	**lmp neighbor** *name* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# lmp neighbor OXC1`	Configures or updates a LMP neighbor and its associated parameters.
Step 4	**ipcc routed** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-nbr-OXC1)# ipcc routed`	Configures a routable Internet Protocol Control Channel (IPCC).

	Command or Action	Purpose
Step 5	**remote node-id** *node-id* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# remote node-id 2.2.2.2`	Configures the remote node ID for an LMP neighbor. **Note** The *node-id* value can also be an IPv4 address

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling LMP Message Exchange

Perform the following task to disable LMP message exchange.

LMP is enabled by default. You can disable LMP on a per neighbor basis using the **lmp static** command in LMP protocol neighbor submode.

Note LMP is recommended unless the peer optical device does not support LMP (in which case it is necessary to disable it at both ends).

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **lmp neighbor** *name*
4. **lmp static**
5. **ipcc routed**
6. **remote node-id** *node-id*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 3	**lmp neighbor** *name* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# lmp neighbor OXC1`	Configures or updates a LMP neighbor and its associated parameters.
Step 4	**lmp static** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# lmp static`	Disables dynamic LMP procedures for the specified neighbor, including LMP hello and LMP link summary. **Note** Use this command for neighbors that do not support dynamic lmp procedures.
Step 5	**ipcc routed** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-nbr-OXC1)# ipcc routed`	Configures a routable IPCC.

	Command or Action	Purpose
Step 6	**remote node-id** *node-id* **Example:** `RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# remote node-id 2.2.2.2`	Configures the remote node ID for an LMP neighbor. **Note** The node ID value must be an IPv4 address.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# end`
	or
	`RP/0/RP0/CPU0:router(config-rsvp-if-sw-0x1)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Remote TE Link Adjacency Information for Numbered Links

Perform this task to configure remote TE link adjacency information for numbered links.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **interface** *type interface-id*
4. **lmp data-link adjacency**
5. **remote switching-capability** {**fsc** | **lsc** | **psc1**}
6. **remote interface-id unnum** *value*
7. **remote te-link ipv4 A.B.C.D**
8. **exit**
9. **lmp neighbor** *name*
10. **remote node-id** *A.B.C.D*
11. **end**
    or
    commit
12. **show mpls lmp**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 3	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# interface POS0/6/0/0`	Enters MPLS-TE interface configuration mode and enables TE on a particular interface on the originating node.
Step 4	**lmp data-link adjacency** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# lmp data-link adjacency`	Configures LMP neighbor remote TE links.
Step 5	**remote switching-capability** {**fsc** | **lsc** | **psc1**} **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote switching-capability lsc`	Configures the remote LMP MPLS-TE interface switching capability.
Step 6	**remote interface-id unnum** *interface identifier* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote interface-id unnum 7`	Configures the unnumbered interface identifier. **Note** Identifiers you specify using this command are the values assigned by the neighbor at the remote side.
Step 7	**remote te-link ipv4 A.B.C.D** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote te-link ipv4 10.10.10.10`	Configures the remote LMP MPLS-TE link ID address.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit`	Exits the current configuration mode.
Step 9	**lmp neighbor** *name* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# neighbor OXC1`	Configures or updates an LMP neighbor and its associated parameters.

	Command or Action	Purpose
Step 10	**remote node-id** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote te-link-id ipv4 10.10.10.10`	Configures the remote LMP MPLS-TE link ID address.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 12	**show mpls lmp** **Example:** `RP/0/RP0/CPU0:router# show mpls lmp`	Verifies the assigned value for the local interface identifiers.

Configuring Remote TE Link Adjacency Information for Unnumbered Links

Perform this task to configure remote TE link adjacency information for unnumbered links.

Note To display the assigned value for the local interface identifiers, use the **show mpls lmp** command.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **interface** *type interface-id*
4. **lmp data-link adjacency**
5. **neighbor** *name*
6. **remote te-link-id unnum**
7. **remote interface-id unnum**

8. **remote switching-capability**
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 3	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# interface POS0/6/0/0`	Enters MPLS-TE interface configuration mode and enables TE on a particular interface on the originating node.
Step 4	**lmp data link adjacency** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# lmp data-link adjacency`	Configures LMP neighbor remote TE links.
Step 5	**neighbor** *name* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# neighbor OXC1`	Configures or updates a LMP neighbor and its associated parameters.
Step 6	**remote te-link-id unnum** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote te-link-id unnum 111`	Configures the unnumbered interface and identifier.
Step 7	**remote interface-id unnum** *interface identifier* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote interface-id unnum 7`	Configures the unnumbered interface identifier. **Note** Identifiers you specify using this command are the values assigned by the neighbor at the remote side.

	Command or Action	Purpose
Step 8	**remote switching-capability** {**fsc** | **lsc** | **psc1**} **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# remote switching-capability lsc`	Configures emote the LMP MPLS-TE interface switching capability.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-te-if-adj)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Numbered and Unnumbered Optical TE Tunnels

This section includes the following subtasks:

- Configuring an Optical TE Tunnel Using Dynamic Path Option, page MPC-154
- Configuring an Optical TE Tunnel Using Explicit Path Option, page MPC-157

Note Before you can successfully bring optical TE tunnels "up," you must complete the procedures in the preceding sections.

The following characteristics can apply to the headend (or, signaling) router:

- Tunnels can be numbered or unnumbered.
- Tunnels can be dynamic or explicit.

The following characteristics can apply to the tailend (or, passive) router:

- Tunnels can be numbered or unnumbered.
- Tunnels must use the explicit path-option.

Configuring an Optical TE Tunnel Using Dynamic Path Option

Perform this task to configure a numbered or unnumbered optical tunnel on a router; in this example, the dynamic path option on the headend router.

The dynamic option does not require that you specify the different hops to be taken along the way. The hops are calculated automatically.

Note This section provides two examples that describe how to configure a optical tunnels. It does not include procedures for every option available on the headend and tailend routers.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*
3. **ipv4 address** *A.B.C.D/prefix*
   or
   ipv4 unnumbered interface *type interface-id*
4. **switching transit** *switching type* **encoding** *encoding type*
5. **priority** *setup-priority hold-priority*
6. **signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*}
7. **destination** *A.B.C.D*
8. **path-option** *path-id* **dynamic**
9. **direction** [**bidirectional**]
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te1`	Enters MPLS-TE interface configuration mode.

	Command or Action	Purpose
Step 3	**ipv4 address** *A.B.C.D/prefix* or **ipv4 unnumbered interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.0.0.0`	Specifies a primary or secondary IPv4 address for an interface. • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and a number (prefix length). The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address compose the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash. or • Enables IPv4 processing on a point-to-point interface without assigning an explicit IPv4 address to that interface.
Step 4	**switching transit** *switching type* **encoding** *encoding type* **Example:** `RP/0/RP0/CPU0:router(config-if)# switching transit lsc encoding sonetsdh`	Specifies the switching capability and encoding types for all transit TE links used to signal the optical tunnel.
Step 5	**priority** *setup-priority hold-priority* **Example:** `RP/0/RP0/CPU0:router(config-if)# priority 1 1`	Configures setup and reservation priorities for MPLS-TE tunnels.
Step 6	**signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 10 class-type 1`	Sets the CT0 bandwidth required on this interface. Because the default tunnel priority is 7, tunnels use the default TE class map (namely, class-type 1, priority 7).
Step 7	**destination** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125`	Assigns a destination address on the new tunnel. • The destination address is the remote node's MPLS-TE router ID. • The destination address is the merge point between backup and protected tunnels.
Step 8	**path-option** *path-id* **dynamic** **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option 1 dynamic`	Configures the dynamic path option and path ID.

	Command or Action	Purpose
Step 9	**direction** [**bidirectional**] **Example:** `RP/0/RP0/CPU0:router(config-if)# direction bidirection`	Configures a bidirectional optical tunnel for GMPLS.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an Optical TE Tunnel Using Explicit Path Option

Perform this task to configure a numbered or unnumbered optical TE tunnel on a router.

This task can apply to both the headend and tailend router.

Note You cannot configure dynamic tunnels on the tailend router.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*
3. **ipv4 address** *ipv4-address mask*
   or
   ipv4 unnumbered interface *type interface-id*
4. **passive**
5. **match identifier**
6. **destination** *A.B.C.D*

7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router# interface POS9/0`	Moves configuration to the interface level, directing subsequent configuration commands to the specified interface.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router# interface POS9/0`	Enters MPLS-TE interface configuration mode.
Step 3	**ipv4 address** ipv4-address mask **or** **ipv4 unnumbered interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router# interface POS9/0`	Specifies a primary or secondary IPv4 address for an interface. • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and a number (prefix length). The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address compose the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash. or • Enables IPv4 processing on a point-to-point interface without assigning an explicit IPv4 address to that interface.
Step 4	**passive** **Example:** `RP/0/RP0/CPU0:router# passive`	Configures a passive interface. **Note** The tailend (passive) router does not signal the tunnel, it simply accepts a connection from the headend router. The tailend router supports the same configuration as the headend router.

	Command or Action	Purpose
Step 5	**match identifier** **Example:** `RP/0/RP0/CPU0:router# match identifier`	Configures the match identifier. You must enter the hostname for the head router then underscore _t, and the tunnel number for the head router. If tunnel-te1 is configured on the head router with a hostname of gmpls1, CLI is match identifier gmpls1_t1. **Note** The match identifier must correspond to the tunnel-te number configured on the headend router. Together with the address specified using the **destination** keyword, this identifier uniquely identifies acceptable incoming tunnel requests.
Step 6	**destination** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125`	Assigns a destination address on the new tunnel. • The destination address is the remote node's MPLS-TE router ID. • The destination address is the merge point between backup and protected tunnels.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring LSP Hierarchy

This section describes the high-level steps required to configure LSP hierarchy.

LSP hierarchy allows standard MPLS-TE tunnels to be established over GMPLS-TE tunnels.

Consider the following information when configuring LSP hierarchy:

- LSP hierarchy supports numbered optical TE tunnels with IPv4 addresses only.
- LSP hierarchy supports numbered optical TE tunnels using numbered or unnumbered TE links.

Note Before you can successfully configure LSP hierarchy, you must first establish a numbered optical tunnel between the headend and tailend routers, as described in Configuring Numbered and Unnumbered Optical TE Tunnels, page MPC-154.

To configure LSP hierarchy, you must perform a series of tasks that have been previously described in this GMPLS configuration section. The tasks, which must be completed in the order presented, are as follows:

1. Establish an optical TE tunnel.
2. Configure an optical TE tunnel under IGP.
3. Configure the bandwidth on the optical TE tunnel.
4. Configure the optical TE tunnel as a TE link.
5. Configure an MPLS-TE tunnel.

Configuring Border Control Model

Border model lets you specify the optical core tunnels to be advertised to edge packet topologies. Using this model, the entire topology is stored in a separate packet instance, allowing packet networks where these optical tunnels are advertised to use LSP hierarchy to signal an MPLS tunnel over the optical tunnel.

Consider the following information when configuring protection and restoration:

- The GMPLS optical TE tunnel must be numbered and have a valid IPv4 address.
- The router ID, which is used for the IGP area and interface ID, must be consistent in all areas.
- The OSPF interface ID may be a numeric or alphanumeric.

Note Border model control functionality is provided for multiple IGP instances in one area or in multiple IGP areas.

To configure border control model functionality, you will perform a series of tasks that have been previously described in this GMPLS configuration section. The tasks, which must be completed in the order presented, are as follows:

1. Configure two optical tunnels on different interfaces.

Note When configuring IGP, you must keep the optical and packet topology information in separate routing tables.

2. Configure OSPF adjacency on each tunnel.
3. Configure bandwidth on each tunnel.
4. Configure packet tunnels.

Configuring Path Protection

This section provides the following sections to configure path protection:

- Configuring an LSP, page MPC-161
- Forcing Reversion of the LSP, page MPC-164

Configuring an LSP

Perform this task to configure an LSP for an explicit path.

Path protection is enabled on a tunnel by adding an additional path option configuration at the active end. The path can be configured either explicitly or dynamically.

Note When the dynamic option is used for both working and protecting LSPs, CSPF extensions are used to determine paths with different degrees of diversity. When the paths are computed, they are used over the lifetime of the LSPs. The nodes on the path of the LSP determine if the PSR is or is not for a given LSP. This determination is based on information that is obtained at signaling.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*
3. **ipv4 address** *ipv4-address mask*
   or
   ipv4 unnumbered interface *type interface-id*
4. **signalled-name** *name*
5. **switching transit** *capability switching type* **encoding** *encoding type*
6. **switching endpoint** *capability switching type* **encoding** *encoding type*
7. **priority** *setup-priority hold-priority*
8. **signalled-bandwidth** {*bandwidth* **[class-type** *ct*] | **sub-pool** *bandwidth*}
9. **destination** *A.B.C.D*
10. **direction** [**bidirectional]**
11. **path-option** *path-id* **explicit** {**name** *pathname* | *path-number*}
12. **path-option protecting** *path-id* **explicit** {**name** *pathname* | *path-number*}
13. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te1`	Enters tunnel-te interface configuration mode.
Step 3	**ipv4 address** ipv4-address mask or **ipv4 unnumbered interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 99.99.99.2 255.255.255.254`	Specifies a primary or secondary IPv4 address for an interface. • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and a number (prefix length). The prefix length is a decimal value that indicates how many of the high-order contiguous bits of the address compose the prefix (the network portion of the address). A slash must precede the decimal value, and there is no space between the IP address and the slash. or • Enables IPv4 processing on a point-to-point interface without assigning an explicit IPv4 address to that interface.
Step 4	**signalled-name** *name* **Example:** `RP/0/RP0/CPU0:router(config-if)# signalled-name tunnel-te1`	Configures the name of the tunnel required for an MPLS TE tunnel. • Use the *name* argument to specify the signal for the tunnel.
Step 5	**switching transit** *capability switching type* **encoding** *encoding type* **Example:** `RP/0/RP0/CPU0:router(config-if)# switching transit lsc encoding sonetsdh`	Specifies the switching capability and encoding types for all transit TE links used to signal the optical tunnel to configure an optical LSP.
Step 6	**switching endpoint** *capability* switching type **encoding** *encoding type* **Example:** `RP/0/RP0/CPU0:router(config-if)# switching endpoint psc1 encoding sonetsdh`	Specifies the switching capability and encoding types for all endpoint TE links used to signal the optical tunnel that is mandatory to set up the GMPLS LSP.

	Command or Action	Purpose
Step 7	**priority** *setup-priority hold-priority* **Example:** `RP/0/RP0/CPU0:router(config-if)# priority 2 2`	Configures setup and reservation priorities for MPLS-TE tunnels.
Step 8	**signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 2488320`	Configures the bandwidth required for an MPLS TE tunnel. The **signalled-bandwidth** command supports two bandwidth pools (class-types) for **Diff-Serv Aware TE** (DS-TE) feature.
Step 9	**destination** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-if)# destination 24.24.24.24`	Assigns a destination address on the new tunnel. • The destination address is the remote node's MPLS-TE router ID. • The destination address is the merge point between backup and protected tunnels.
Step 10	**direction** [**bidirectional**] **Example:** `RP/0/RP0/CPU0:router(config-if)# direction bidirection`	Configures a bidirectional optical tunnel for GMPLS.
Step 11	**path-option** *path-id* **explicit** {**name** *pathname* | path-number} **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option 1 explicit name po4`	Configures the explicit path option and path ID.

	Command or Action	Purpose
Step 12	**path-option protecting** *path-id* **explicit** {**name** *pathname* | *path-number*} **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option protecting 1 explicit name po6`	Configures the path setup option to protect a path.

Step 13	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Forcing Reversion of the LSP

Perform this task to allow a forced reversion of the LSPs, which is only applicable to 1:1 LSP protection.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng path-protection switchover** {*tunnel name* | *number*}
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng path-protection switchover** {*tunnel name* | *number*} **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng path-protection switchover 1`	Specifies a manual switchover for path protection for a GMPLS optical LSP. The tunnel ID is configured for a switchover. The **mpls traffic-eng path-protection switchover** command must be issued on both head and tail router of the GMPLS LSP to achieve the complete path switchover at both ends.

Step 3	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Flexible Name-based Tunnel Constraints

To fully configure MPLS-TE Flexible Name-based Tunnel Constraints, you must complete the following high-level tasks in order:

1. Assigning Color Names to Numeric Values, page MPC-166
2. Associating Affinity-Names with TE Links, page MPC-167
3. Associating Affinity Constraints for TE Tunnels, page MPC-168

Assigning Color Names to Numeric Values

The first task in enabling the new coloring scheme is to assign a numerical value (in hexadecimal) to each value (color).

Note An affinity color name cannot exceed 64 characters. An affinity value cannot exceed a single digit. For example, magenta1.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **affinity-map** {*affinity name* | *affinity value*}
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic engineering** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic eng`	Enters MPLS-TE mode.

	Command or Action	Purpose
Step 3	**affinity-map {*affinity name* | *affinity value*}** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# affinity-map red 1`	Enters an affinity name, or a map value, using a color name (repeat this command to assign multiple colors up to a maximum of 64 colors). An affinity color name cannot exceed 64 characters. The value you assign to a color name must be a single digit.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-te)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-te)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating Affinity-Names with TE Links

The next step in the configuration of MPLS-TE Flexible Name-based Tunnel Constraints is to assign affinity names and values to TE links.

You can assign up to a maximum of 32 colors. Before you assign a color to a link, you must define the name-to-value mapping for each color as described in Assigning Color Names to Numeric Values, page MPC-166.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng interface** *type interface-id*
3. **attribute-names** *color1 color2*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic eng interface tunnel-te2`	Enters MPLS-TE mode to configure an interface.
Step 3	**attribute-names** *color1 color2* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te-if)# red`	Assigns colors to TE links over the selected interface.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-te-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-te-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating Affinity Constraints for TE Tunnels

The final step in the configuration of MPLS-TE Flexible Name-based Tunnel Constraints requires that you associate a tunnel with affinity constraints.

Using this model, there are no masks. Instead, there is support for four types of affinity constraints:

- include
- include-strict
- exclude
- exclude-all

Note For the affinity constraints above, all but the exclude-all constraint may be associated with up to 10 colors.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **affinity** *index* {**include** | **include-strict** | **exclude** | **exclude-all**} *color*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *tunnel-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te 1`	Selects the a tunnel/interface.

	Command or Action	Purpose
Step 3	**affinity** *index* {**include** | **include-strict** | **exclude** | **exclude-all**} *color* **Example:** `RP/0/RP0/CPU0:router(config-if)# affinity 0 include red`	Enter link attributes for links comprising tunnel. UP TO TEN COLORS. There can be multiple include statements under tunnel configuration as in the above configuration. With the following configuration, a link is eligible for CSPF if it has at least red color OR has at least green color. Thus, a link with red and any other colors as well as a link with green and any additional colors meet the above constraint.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IS-IS to Flood MPLS-TE Link Information

Perform this task to configure a router running the Intermediate System-to-Intermediate System (IS-IS) protocol to flood MPLS-TE link information into multiple IS-IS levels.

This procedure shows how to enable MPLS-TE in both IS-IS Level 1 and Level 2.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **net** *network-entity-title*
4. **address-family {ipv4 | ipv6} {unicast}**
5. **metric-style wide**
6. **mpls traffic-eng** *level*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# interface POS9/0`	Enters global configuration mode.
Step 2	**router isis** *instance-id* Example: `RP/0/RP0/CPU0:router(config)# router is-is 1`	Enters an IS-IS instance.
Step 3	**net** ***network-entity-title*** **Example:** `RP/0/RP0/CPU0:router(config-isis)# net 47.0001.0000.0000.0002.00`	Enters an IS-IS network entity title (NET) for the routing process.
Step 4	**address-family {ipv4 | ipv6} {unicast}** **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Enters address family configuration mode for configuring IS-IS routing that use IPv4 and IPv6 address prefixes.
Step 5	**metric-style wide** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide`	Enter the new-style type, length, and value (TLV) objects.

	Command or Action	Purpose
Step 6	**mpls traffic-eng** ***level*** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng level-1-2`	Enter the required MPLS-TE level or levels.

Step 7	**end**
	 or
	 commit
	
	 Example:
	 `RP/0/RP0/CPU0:router(config-isis-af)# end`
	 or
	 `RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		 • When you issue the **end** command, the system prompts you to commit changes:
		 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		 – Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		 – Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		 – Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		 • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an OSPF Area of MPLS-TE

Perform this task to configure an OSPF area for MPLS-TE in both the OSPF backbone area 0 and area 1.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **mpls traffic-eng router-id** *type-interface*
4. **area** *area-id*
5. **mpls traffic-eng**
6. **interface** *type interface-id*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 100`	Enters a name that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.
Step 3	**mpls traffic-eng router-id** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng router-id Loopback0`	Enters the MPLS interface type. For more information, use the question mark (?) online help function.
Step 4	**area** ***area-id*** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters an OSPF area identifier. The *area-id* argument can be specified as either a decimal value or an IP address.
Step 5	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# area 0`	Enters an OSPF area identifier. The *area-id* argument can be specified as either a decimal value or an IP address.

	Command or Action	Purpose
Step 6	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/2/0/0`	Enters an interface ID. For more information, use the question mark (?) online help function.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Explicit Paths with ABRs Configured as Loose Addresses

Perform this task to specify an IPv4 explicit path with ABRs configured as loose addresses.

SUMMARY STEPS

1. **configure**
2. **explicit-path** *name*
3. **index** *number* **next-address loose ipv4 unicast** *A.B.C.D*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# interface POS9/0`	Enters global configuration mode.
Step 2	**explicit-path** *name* **Example:** `RP/0/RP0/CPU0:router(config)# explicit-path interarea1`	Enters a name for the explicit path.
Step 3	**index** *number* **next-address loose ipv4 unicast** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-expl-path)# index 1 next-address loose ipv4 unicast 10.10.10.10`	Includes a path entry at a specific index.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-expl-path)# end`
	or
	`RP/0/RP0/CPU0:router(config-expl-path)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring MPLS-TE Forwarding Adjacency

Perform this task to configure forwarding adjacency on a specific tunnel-te interface.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *number*

3. **forwarding-adjacency holdtime** *value*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# interface POS9/0`	Enters global configuration mode.
Step 2	**interface tunnel-te** *number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te 1`	Enters MPLS-TE interface configuration mode.
Step 3	**forwarding-adjacency holdtime** *value* **Example:** `RP/0/RP0/CPU0:router(config-if)# forwarding-adjacency holdtime 60`	Configures forwarding adjacency using an optional specific holdtime value. By default, this value is 0 (milliseconds).

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Unequal Load Balancing

Perform the following tasks to configure unequal load balancing:

- Setting Unequal Load Balancing Parameters, page MPC-177
- Enabling Unequal Load Balancing, page MPC-178

Setting Unequal Load Balancing Parameters

The first step you must take to configure unequal load balancing requires that you set the parameters on each specific interface.

The default load share for tunnels with no explicit configuration is the configured bandwidth.

Note Equal load-sharing occurs if there is no configured bandwidth.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id*
3. **load-share** *value*
4. **end**
   or
   commit
5. **show mpls traffic-eng tunnels**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# config`	Enters global configuration mode.
Step 2	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# interface tunnel-te1.`	Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node. **Note** Only tunnel-te interfaces are permitted.
Step 3	**load-share** *value* **Example:** `RP/0/RP0/CPU0:router(config-if)# load-share 1000`	Configures the load-sharing parameters for the specified interface.

	Command or Action	Purpose

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show mpls traffic-eng tunnels** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels`	Verifies the state of unequal load balancing, including bandwidth and load-share values.

Enabling Unequal Load Balancing

This task describes how to enable unequal load balancing. (Quite simply, this is a global switch used to turn unequal load-balancing on or off.)

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. load-share **unequal**
4. **end**
   or
   commit
5. **show mpls traffic-eng tunnels**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# config`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters the MPLS-TE configuration mode.
Step 3	**load-share unequal** **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# load-share unequal`	Enables unequal load sharing across TE tunnels to the same destination.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-te)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-te)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show mpls traffic-eng tunnels** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels`	Verifies the state of unequal load balancing, including bandwidth and load-share values.

Configuring a Path Computation Client and Element

Perform the following tasks to configure PCE:

- Configuring a Path Computation Client, page MPC-180
- Configuring a Path Computation Element Address, page MPC-181
- Configuring PCE Parameters, page MPC-182

Configuring a Path Computation Client

Perform this task to configure a TE tunnel as a PCC.

Note Only one TE-enabled IGP instance can be used at a time.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **path-option** {*number*} **dynamic pce** [*address]*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# config`	Enters global configuration mode.
Step 2	**interface tunnel-te** *tunnel-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te 6`	Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node.

	Command or Action	Purpose
Step 3	**path-option** {*number*} **dynamic pce** [*address*] **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option 1 dynamic pce`	Configures a TE tunnel as a PCC.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Path Computation Element Address

Perform this task to configure a PCE address.

Note Only one TE-enabled IGP instance can be used at a time.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **pce address ipv4** *address*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters the MPLS-TE configuration mode.
Step 3	**pce address ipv4** *address* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce address ipv4 10.1.1.1`	Configures a PCE IPv4 address.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-te)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-te)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring PCE Parameters

Perform this task to configure PCE parameters, including a static PCE peer, periodic reoptimization timer values, and request timeout values.

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng**
3. **pce address ipv4** *address*

4. **pce peer ipv4 address** *address*
5. **pce keepalive** *interval*
6. **pce deadtimer** *value*
7. **pce reoptimize** *value*
8. **pce request-timeout** *value*
9. **pce tolerance keepalive** *value*
10. **end**
    or
    commit
11. **show mpls traffic pce peer [***address* **| all]**
12. **show mpls traffic-eng pce tunnels**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# config`	Enters global configuration mode.
Step 2	**mpls traffic-eng** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng`	Enters MPLS-TE configuration mode.
Step 3	**pce address ipv4** *address* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce address ipv4 10.1.1.1`	Configures a PCE IPv4 address.
Step 4	**pce peer address ipv4** *address* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce peer address ipv4 10.1.1.1`	(Optional) Configures a static PCE peer address. This step is optional; PCE peers are also discovered dynamically via OSPF/ISIS.
Step 5	**pce keepalive** *interval* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce keepalive 10`	Configures a PCEP keepalive interval. The range is 0 to 255 seconds. When the keepalive interval is 0, the LSR does not send keepalive messages.
Step 6	**pce deadtimer** *value* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce deadtimer 50`	Configures a PCE deadtimer value. The range is 0 to 255 seconds. When the dead interval is 0, the LSR does not timeout a PCEP session to a remote peer.

	Command or Action	Purpose
Step 7	**pce reoptimize** *value* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce reoptimize 200`	Configures a periodic reoptimization timer value. The range is 60 to 604800 seconds. When the dead interval is 0, the LSR does not timeout a PCEP session to a remote peer.
Step 8	**pce request-timeout** *value* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce request-timeout 10`	Configures a PCE request-timeout. Range is 5 to 100 seconds. PCC/PCE keeps a pending path request only for the request-timeout period.
Step 9	**pce tolerance keepalive** *value* **Example:** `RP/0/RP0/CPU0:router(config-mpls-te)# pce tolerance keepalive 10`	(Optional) Configures a PCE tolerance keepalive value (which is the minimum acceptable peer proposed keepalive).

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show mpls traffic pce peer [***address* **| all]** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng pce peer`	(Optional) Verifies the PCE peer address and state.
Step 12	**show mpls traffic-eng pce tunnels** **Example:** `RP/0/RP0/CPU0:router# show mpls traffic-eng pce tunnels`	(Optional) Verifies status PCE tunnels.

Configuring Policy-based Tunnel Selection

Perform this task to configure policy-based tunnel selection (PBTS).

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **ipv4 unnumbered loopback** *number*
4. **signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*}
5. **autoroute announce**
6. **destination** *A.B.C.D*
7. **policy-class** *1 - 7*
8. **path-option** *path-id* **explicit name** *explicit-path-name*
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-te** *tunnel-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-te 6`	Enters MPLS-TE interface configuration mode and enables traffic engineering on a particular interface on the originating node.
Step 3	**ipv4 unnumbered loopback** *number* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered loopback 0`	Assigns a source address so that forwarding can be performed on the new tunnel.
Step 4	**signalled-bandwidth** {*bandwidth* [**class-type** *ct*] | **sub-pool** *bandwidth*} **Example:** `RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 10 class-type 1`	Configures the bandwidth required for an MPLS TE tunnel. Because the default tunnel priority is 7, tunnels use the default TE class map (namely, class-type 1, priority 7).

	Command or Action	Purpose
Step 5	**autoroute announce** **Example:** `RP/0/RP0/CPU0:router(config-if)# autoroute announce`	Enables messages that notify the neighbor nodes about the routes that are forwarding.
Step 6	**destination** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125`	Assigns a destination address on the new tunnel. • The destination address is the remote node's MPLS-TE router ID. • The destination address is the merge point between backup and protected tunnels.
Step 7	**policy-class** *1 - 7* **Example:** `RP/0/RP0/CPU0:router(config-if)# policy-class 1`	Configures PBTS to direct traffic into specific TE tunnels.
Step 8	**path-option** *path-id* **explicit name** *explicit-path-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# path-option 1 explicit name backup-path`	Sets the path option to explicit with a given name (previously configured) and assigns the path ID.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Cisco MPLS-TE

This section provides the following examples:

- Configuring Fast Reroute and SONET APS: Example, page MPC-187
- Building MPLS-TE Topology and Tunnels: Example, page MPC-188
- Configuring IETF Diff-Serv TE Tunnels: Example, page MPC-189
- Configuring the Ignore IS-IS Overload Bit Setting in MPLS-TE: Example, page MPC-189
- Configuring GMPLS: Example, page MPC-189
- Configuring Flexible Name-based Tunnel Constraints: Example, page MPC-191
- Configuring an Interarea Tunnel: Example, page MPC-193
- Configuring Forwarding Adjacency: Example, page MPC-193
- Configuring Unequal Load Balancing: Example, page MPC-193
- Configuring PCE: Example, page MPC-194
- Configure Policy-based Tunnel Selection: Example, page MPC-195

Configuring Fast Reroute and SONET APS: Example

When SONET Automatic Protection Switching (APS) is configured on a router, it does not offer protection for tunnels; because of this limitation, fast reroute (FRR) still remains the protection mechanism for MPLS-TE.

When APS is configured in a SONET core network, an alarm might be generated toward a router downstream. If this router is configured with FRR, the hold-off timer must be configured at the SONET level to prevent FRR from being triggered while the core network is performing a restoration. Enter the following commands to configure the delay:

```
RP/0/RP0/CPU0:Route-3(config)# controller sonet 0/6/0/0 delay trigger line 250
RP/0/RP0/CPU0:Route-3(config)# controller sonet 0/6/0/0 path delay trigger 300
```

Building MPLS-TE Topology and Tunnels: Example

The following examples show how to build an OSPF and IS-IS topology:

```
(OSPF)
...
configure
  mpls traffic-eng
  interface pos 0/6/0/0
  router id loopback 0
  router ospf 1
  router-id 192.168.25.66
  area 0
  interface pos 0/6/0/0
  interface loopback 0
  mpls traffic-eng router-id loopback 0
  mpls traffic-eng area 0
  rsvp
  interface pos 0/6/0/0
  bandwidth 100
  commit
show mpls traffic-eng topology
show mpls traffic-eng link-management advertisement
!
(IS-IS)
...
configure
  mpls traffic-eng
  interface pos 0/6/0/0
  router id loopback 0
  router isis lab
  address-family ipv4 unicast
  mpls traffic-eng level 2
  mpls traffic-eng router-id Loopback 0
  !
  interface POS0/0/0/0
  address-family ipv4 unicast
!
```

The following example shows how to configure tunnel interfaces:

```
interface tunnel-te1
  destination 192.168.92.125
  ipv4 unnumbered loopback 0
  path-option 1 dynamic
  bandwidth 100
  commit
show mpls traffic-eng tunnels
show ipv4 interface brief
show mpls traffic-eng link-management admission-control
!
interface tunnel-te1
  autoroute announce
  route ipv4 192.168.12.52/32 tunnel-te1
  commit
ping 192.168.12.52
show mpls traffic autoroute
!
interface tunnel-te1
  fast-reroute
  mpls traffic-eng interface pos 0/6/0/0
  backup-path tunnel-te 2
  interface tunnel-te2
  backup-bw global-pool 5000
```

```
  ipv4 unnumbered loopback 0
  path-option 1 explicit name backup-path
  destination 192.168.92.125
 commit
show mpls traffic-eng tunnels backup
show mpls traffic-eng fast-reroute database
!
rsvp
  interface pos 0/6/0/0
  bandwidth 100 150 sub-pool 50
  interface tunnel-te1
  bandwidth sub-pool 10
commit
```

Configuring IETF Diff-Serv TE Tunnels: Example

The following example shows how to configure DiffServ-TE:

```
rsvp
 interface pos 0/6/0/0
 bandwidth rdm 100 150 bc1 50
 mpls traffic-eng
 ds-te mode ietf
 interface tunnel-te 1
 bandwidth 10 class-type 1
 commit

configure
 rsvp interface 0/6/0/0
 bandwidth mam max-reservable-bw 400 bc0 300 bc1 200
 mpls traffic-eng
 ds-te mode ietf
 ds-te model mam
 interface tunnel-te 1bandwidth 10 class-type 1
 commit
```

Configuring the Ignore IS-IS Overload Bit Setting in MPLS-TE: Example

The following example shows how to configure the IS-IS overload bit setting in MPLS-TE:

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

Configuring GMPLS: Example

This example shows how to set up headend and tailend routers with bidirectional optical unnumbered tunnels using numbered TE links:

Headend Router

```
router ospf roswell
 router-id 11.11.11.11
 nsf cisco
 area 23
 !
```

```
 area 51
  interface Loopback 0
  !
  interface MgmtEth0/0/CPU0/1
  !
  interface POS0/4/0/1
  !
 !
 mpls traffic-eng router-id Loopback 0
 mpls traffic-eng area 51
!

rsvp
 interface POS0/2/0/3
  bandwidth 2000
 !
!
interface tunnel-te1
 ipv4 unnumbered Loopback 0
 switching transit fsc encoding sonetsdh
 switching endpoint psc1 encoding packet
 priority 3 3
 signalled-bandwidth 500
 destination 55.55.55.55
 direction bidirectional
 path-option 1 dynamic
!

mpls traffic-eng
 interface POS0/2/0/3
  flooding-igp ospf roswell area 51
  switching key 1
   encoding packet
   capability psc1
  !
  switching link
   encoding sonetsdh
   capability fsc
  !
  lmp data-link adjacency
   neighbor gmpls5
   remote te-link-id ipv4 10.0.0.5
   remote interface-id unnum 12
   remote switching-capability psc1
  !
 !
 lmp neighbor gmpls5
  ipcc routed
  remote node-id 55.55.55.55
 !
!
```

Tailend Router

```
router ospf roswell
 router-id 55.55.55.55
 nsf cisco
 area 23
 !
 area 51
  interface Loopback 0
  !
  interface MgmtEth0/0/CPU0/1
  !
```

```
  interface POS0/4/0/2
  !
 !
 mpls traffic-eng router-id Loopback 0
 mpls traffic-eng area 51
!

mpls traffic-eng
 interface POS0/2/0/3
  flooding-igp ospf roswell area 51
  switching key 1
   encoding packet
   capability psc1
  !
  switching link
   encoding sonetsdh
   capability fsc
  !
  lmp data-link adjacency
   neighbor gmpls1
   remote te-link-id ipv4 10.0.0.1
   remote interface-id unnum 12
   remote switching-capability psc1
  !
 !
 lmp neighbor gmpls1
  ipcc routed
  remote node-id 11.11.11.11
 !
!
rsvp
 interface POS0/2/0/3
  bandwidth 2000
 !
!
interface tunnel-te1
 ipv4 unnumbered Loopback 0
 passive
 match identifier head_router_hostname_t1
 destination 11.11.11.11
!
```

Configuring Flexible Name-based Tunnel Constraints: Example

The following configuration shows the three-step process used to configure Flexible Name-based Tunnel Constraints.

```
R2
line console
 exec-timeout 0 0
 width 250
!
logging console debugging
explicit-path name mypath
 index 1 next-address loose ipv4 unicast 3.3.3.3 !
explicit-path name ex_path1
 index 10 next-address loose ipv4 unicast 2.2.2.2  index 20 next-address loose ipv4
unicast 3.3.3.3 !
interface Loopback0
 ipv4 address 22.22.22.22 255.255.255.255 !
interface tunnel-te1
 ipv4 unnumbered Loopback0
```

```
 signalled-bandwidth 1000000
 destination 3.3.3.3
 affinity include green
 affinity include yellow
 affinity exclude white
 affinity exclude orange
 path-option 1 dynamic
!
router isis 1
 is-type level-1
 net 47.0001.0000.0000.0001.00
 nsf cisco
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-1
  mpls traffic-eng router-id Loopback0
 !
 interface Loopback0
  passive
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/1/0/0
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/1/0/1
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/1/0/2
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/1/0/3
  address-family ipv4 unicast
  !
 !
!
rsvp
 interface GigabitEthernet0/1/0/0
  bandwidth 1000000 1000000
 !
 interface GigabitEthernet0/1/0/1
  bandwidth 1000000 1000000
 !
 interface GigabitEthernet0/1/0/2
  bandwidth 1000000 1000000
 !
 interface GigabitEthernet0/1/0/3
  bandwidth 1000000 1000000
 !
!
mpls traffic-eng
 interface GigabitEthernet0/1/0/0
  attribute-names red purple
 !
 interface GigabitEthernet0/1/0/1
  attribute-names red orange
 !
 interface GigabitEthernet0/1/0/2
  attribute-names green purple
 !
 interface GigabitEthernet0/1/0/3
```

```
  attribute-names green orange
 !
 affinity-map red 1
 affinity-map blue 2
 affinity-map black 80
 affinity-map green 4
 affinity-map white 40
 affinity-map orange 20
 affinity-map purple 10
 affinity-map yellow 8
!
```

Configuring an Interarea Tunnel: Example

The following configuration example shows how to configure a traffic engineering interarea tunnel. Router R1 is the headend for tunnel1, and router R2 (20.0.0.20) is the tailend. Tunnel1 is configured with a path option that is loosely routed through Ra and Rb.

Note Specifying the tunnel tailend in the loosely router path is optional.

```
config
interface Tunnel-te1
ipv4 unnumbered Loopback0
destination 192.168.20.20
signalled-bandwidth 300
path-option 1 explicit name path-tunnel1
explicit-path name path-tunnel1
next-address loose 192.168.40.40
next-address loose 192.168.60.60
next-address loose 192.168.20.20
```

Note Generally for an interarea tunnel you should configure multiple loosely routed path options that specify different combinations of ABRs (for OSPF) or level-1-2 boundary routers (for IS-IS) to increase the likelihood that the tunnel is successfully signaled. In this simple topology there are no other loosely routed paths.

Configuring Forwarding Adjacency: Example

The following configuration example shows how to configure an MPLS-TE forwarding adjacency on tunnel-te 68 with a holdtime value of 60:

```
configure
 interface tunnel-te 68
 forwarding-adjacency holdtime 60
 commit
```

Configuring Unequal Load Balancing: Example

The following configuration example illustrates unequal load balancing configuration:

```
configure
  interface tunnel-te0
```

```
    destination 1.1.1.1
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
  interface tunnel-te1
    destination 1.1.1.1
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    load-share 5
  interface tunnel-te2
    destination 1.1.1.1
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 5
  interface tunnel-te10
    destination 2.2.2.2
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 10
  interface tunnel-te11
    destination 2.2.2.2
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 10
  interface tunnel-te12
    destination 2.2.2.2
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 20
  interface tunnel-te20
    destination 3.3.3.3
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 10
  interface tunnel-te21
    destination 3.3.3.3
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 10
    load-share 20
  interface tunnel-te30
    destination 4.4.4.4
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 10
    load-share 5
  interface tunnel-te31
    destination 4.4.4.4
    path-option 1 dynamic
    ipv4 unnumbered Loopback0
    signalled-bandwidth 10
    load-share 20
  mpls traffic-eng
    load-share unequal
end
```

Configuring PCE: Example

The following configuration example illustrates a PCE configuration:

```
configure
mpls traffic-eng
```

```
  interface pos 0/6/0/0
  pce address ipv4 192.168.25.66
  router id loopback 0
  router ospf 1
  router-id 192.168.25.66
  area 0
  interface pos 0/6/0/0
  interface loopback 0
  mpls traffic-eng router-id loopback 0
  mpls traffic-eng area 0
  rsvp
  interface pos 0/6/0/0
  bandwidth 100
  commit
```

The following configuration example illustrates PCC configuration:

```
configure
  int tunnel-te 10
  ipv4 unnumbered loopback 0
  destination 1.2.3.4
  path-option 1 dynamic pce
  mpls traffic-eng
  interface pos 0/6/0/0
  router id loopback 0
  router ospf 1
  router-id 192.168.25.66
  area 0
  interface pos 0/6/0/0
  interface loopback 0
  mpls traffic-eng router-id loopback 0
  mpls traffic-eng area 0
  rsvp
  interface pos 0/6/0/0
  bandwidth 100
  commit
```

Configure Policy-based Tunnel Selection: Example

The following configuration example illustrates a PBTS configuration:

```
configure
 interface tunnel-te0
 ipv4 unnumbered Loopback3
 signalled-bandwidth 50000
 autoroute announce
 destination 1.5.177.2
 policy-class 2
 path-option 1 dynamic
```

Additional References

For additional information related to implementing MPLS-TE, refer to the following references:

Related Documents

Related Topic	Document Title
MPLS-TE commands	*MPLS Traffic Engineering Commands on Cisco IOS XR Software* module in the *Cisco IOS XR MPLS Command Reference*
Cisco CRS-1 router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
4124	*Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering.* F. Le Faucheur, Ed. June 2005. (Format: TXT=79265 bytes) (Status: PROPOSED STANDARD)
4125	*Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering.* F. Le Faucheur, W. Lai. June 2005. (Format: TXT=22585 bytes) (Status: EXPERIMENTAL)
4127	*Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering.* F. Le Faucheur, Ed. June 2005. (Format: TXT=23694 bytes) (Status: EXPERIMENTAL)

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS Optical User Network Interface Protocol on Cisco IOS XR Software

The Optical User Network Interface (O-UNI) is specified by the Optical Internetworking Forum (OIF). The O-UNI standard specifies a means by which client devices, such as routers, Synchronous Optical Network (SONET)/Synchronous Digital Hierarchy (SDH) Add Drop Multiplexers (ADMs), and other devices with SONET/SDH interfaces may request optical layer connectivity services of an optical transport network (OTN). Such services include the establishment of connections between two client devices, the deletion of connections, and the query of connection status.

Note The term MPLS O-UNI is often used instead of O-UNI, as it emphasizes that the OIF's O-UNI is based upon many MPLS standards developed by the Internet Engineering Task Force (IETF).

Feature History for Implementing MPLS O-UNI on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	No modification.
Release 3.3.0	No modification.
Release 3.4.0	No modification.
Release 3.4.1	No modification.
Release 3.5.0	No modification.
Release 3.6.0	No modification.
Release 3.7.0	No modification.

Contents

- Prerequisites for Implementing MPLS O-UNI, page MPC-200
- Information About Implementing MPLS O-UNI, page MPC-200
- How to Implement MPLS O-UNI on Cisco IOS XR Software, page MPC-202
- Configuration Examples for MPLS O-UNI, page MPC-210
- Additional References, page MPC-212

Prerequisites for Implementing MPLS O-UNI

The following prerequisites are required to implement MPLS O-UNI:

- You must be in a user group associated with a task group that includes the proper task IDs for MPLS O-UNI commands.
- A router that runs Cisco IOS XR software.
- Installation of the Cisco IOS XR software mini-image on the router.
- Installation of the Cisco IOS XR MPLS software package on the router.

Information About Implementing MPLS O-UNI

O-UNI offers the ability to establish OIF standards-based connections through a SONET/SDH-based heterogeneous optical network. These connections can be made across optical transport networks (OTNs) composed of Cisco equipment or third-party vendor equipment.

An OTN provides transport services to interconnect the optical interfaces of O-UNI client devices, such as IP routers and SONET ADMs. In Figure 14, two routers running Cisco IOS XR software with O-UNI client (O-UNI-C) support are connected to SONET/SDH cross-connects, which provide O-UNI Network (O-UNI-N) services. These cross-connects sit at the edge of the OTN, and O-UNI client devices may request services from them. The client devices have no knowledge of the OTN structure, and all services are invoked at the edge of the OTN. These services include connection establishment, deletion, and query for a given data link, where a data link corresponds to a unique SONET/SDH interface on an O-UNI-C device.

To complete a connection request, an O-UNI-N node needs a database to determine its route within the OTN. The algorithms used to determine the connection path, although not standardized in the OIF's O-UNI 1.0 standard, must consider the connection characteristics requested by the O-UNI-C device, including connection bandwidth, framing type, cyclic redundancy check (CRC) type, and scrambling.

Routers request O-UNI services using the following RSVP messages:

- path
- reservation
- reservation confirmation
- path error
- path tear
- reservation tear
- refresh

These RSVP messages are transported over IP Control Channels (IPCC) between the router and the O-UNI-N device. The IPCCs rely on IP connectivity between O-UNI-C and O-UNI-N devices, represented in dotted lines in Figure 14. When services from the OTN are requested, the following parameters are included in the RSVP messages transmitted:

- A unique data link identifier
- Bandwidth requested
- Framing type requested (that is, SONET or SDH)
- CRC 16 or 32

- Scrambling type
- IP address of the node to receive the request

A unique identifier exists for every interface participating in an O-UNI connection. This identifier consists of a TNA and an interface ID. The TNA addresses are unique within the OTN, and represent the address of one or more data links between an O-UNI-N device and an O-UNI-C device. Cisco IOS XR software supports the use of IPv4 TNA addresses.

The interface ID is used to uniquely identify a given data link interface connected between an O-UNI-N device and an O-UNI-C device. The interface ID is a 32-bit value with local significance, generated by the device on which an interface resides; for example, a POS interface on a router connected to an O-UNI-N device would have an interface ID generated by the router and is only unique on this router. To avoid reconfiguration of LMP information, it is important that the interface ID values are persistent across control plane restarts and router reloads.

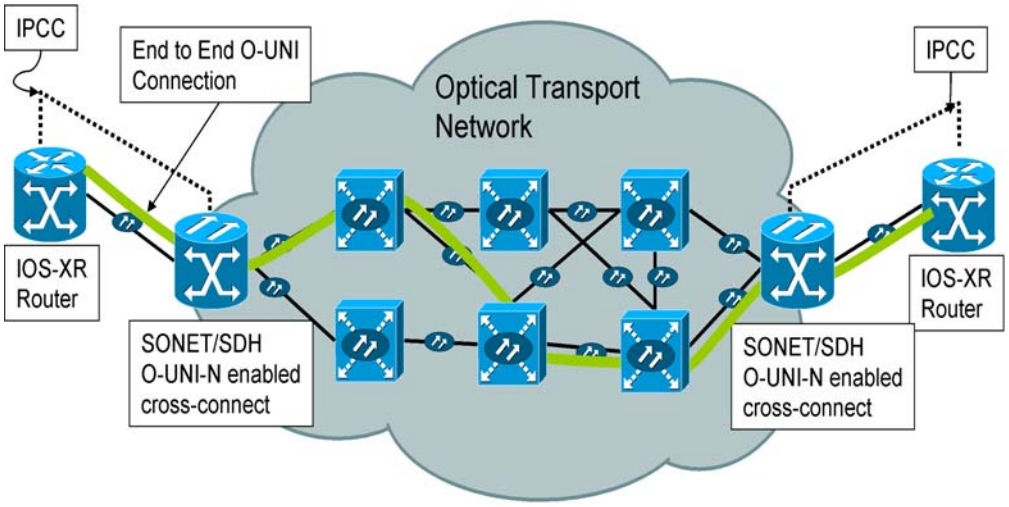
To establish an O-UNI connection, the messaging exchanges must include data link information from other devices. This information is provisioned using a static version of the LMP. The LMP commands allow the provisioning of the following capabilities:

- The TNA associated with the data link. This value is assigned by the operator of the OTN.
- The interface ID of the neighboring device. In Figure 14, this is the interface ID on the SONET/SDH cross-connect referred to as the remote interface ID.
- The node ID of the data link adjacent device. In Figure 14, this is the IPv4 address used to send RSVP messages to a directly attached SONET/SDH cross-connect.

Local information is configured to enable the establishment of O-UNI connections. This information includes:

- The router ID used as the source IPv4 address for RSVP messaging. This value is also configured on neighbor devices. Note that the terms node ID and router ID are often used synonymously. Node ID represents the generic term, while router ID refers to the node ID of a router.
- The TNA of the data link on which to terminate the connection.
- The operational mode of the interface that participates in an O-UNI connection. This interface can be configured to only terminate a connection or to initiate a connection.

Figure 14 O-UNI Network



How to Implement MPLS O-UNI on Cisco IOS XR Software

O-UNI requires setting up data links with neighbor nodes and establishing Internet Protocol Control Channel (IPCC) channels to setup O-UNI connections.

If IP connectivity is established over the RP management port and a standby RP card is present, the following conditions ensure NSF in case of RP failover:

- Standby management port is not shutdown and operational up.
- Standby management port has an IP address assigned to it.
- Proxy-ARP is not enabled (proxy-ARP is disabled by default).
- Active and standby ports have the same IP subnet configured.
- An IP virtual address with the same subnet as the active and standby ports is configured.
- The virtual address above is used as next hop in any static routes configured on neighbor O-UNI-N nodes.

This section contains the following procedures:

- Setting Up an MPLS O-UNI Connection, page MPC-202 (required)
- Tearing Down an MPLS O-UNI Connection, page MPC-205 (required)
- Verifying MPLS O-UNI Configuration, page MPC-207 (required)
- Configuration Examples for MPLS O-UNI, page MPC-210 (optional)

Setting Up an MPLS O-UNI Connection

Perform this task to configure and set up an MPLS O-UNI connection.

Prerequisites

The following prerequisites are required to configure and set up an O-UNI connection:

- To configure the data link parameters you must have a node ID for the neighboring node.
- A stable node ID is required at both ends of the O-UNI data link to ensure the configuration is successful. If you do not assign a node ID (also known as a router ID), the system defaults to the configured global router ID.

SUMMARY STEPS

1. **configure**
2. **snmp-server ifindex persist**
3. **snmp-server interface** *type number* **ifindex persist**
4. **mpls optical-uni**
5. **router-id {*ip-address* | *interface-id*}**
6. **lmp neighbor** *neighbor-name*
7. **ipcc routed**
8. **remote node-id** *ip-address*

9. **exit**
10. **interface** *type number*
11. **lmp data-link adjacency**
12. **neighbor** *neighbor-name*
13. **remote interface-id** *interface-id*
14. **tna ipv4** *ip-address*
15. **exit**
16. **destination address ipv4** *ip-address*
    or
    passive
17. **end**
    or
    commit
18. **show mpls optical-uni**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**snmp-server ifindex persist** **Example:** `RP/0/RP0/CPU0:router(config)# snmp-server ifindex persist`	Uses SNMP generated ifindexes to uniquely identify interfaces, and corresponds to O-UNI's concept of an interface ID. • To ensure that O-UNI interface IDs are persistent across reloads, SNMP must save the ifindexes generated for the interfaces. These identifiers are used for the requested interfaces.
Step 3	**snmp-server interface** *type interface-id* **index persistence** **Example:** `RP/0/RP0/CPU0:router(config)# snmp-server interface pos0/4/0/1 index persistence`	Indicates that an interface ID for this interface is to be generated. • If the **snmp-server ifindex persist** command is entered, this interface ID is made persistent.
Step 4	**mpls optical-uni** **Example:** `RP/0/RP0/CPU0:router(config)# mpls optical-uni`	Enters O-UNI configuration mode.
Step 5	**router-id {*ip-address* | *interface-id*}** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni)# router-id loopback10`	Sets the router ID to the IPv4 address of the interface loopback10.

	Command or Action	Purpose
Step 6	**lmp neighbor** *neighbor-name* **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni)# lmp neighbor router1`	Enters neighbor configuration mode where you enter specific properties for the O-UNI-N neighbor.
Step 7	**ipcc routed** **Example:** `RP/0/RP0/CPU0:router(config-ouni-nbr-router1)# ipcc routed`	Configures a routed IPCC for the O-UNI-N neighbor router1. • Routing determines which interface is used to forward signaling messages to the neighbor.
Step 8	**remote node-id** ***ip-address*** **Example:** `RP/0/RP0/CPU0:router(config-ouni-nbr-router1)# remote node-id 172.34.1.12`	Configures the node ID of the O-UNI-N neighbor router1. • This address is used as the destination address of O-UNI signaling messages sent to the neighbor.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ouni-nbr-router1)# exit`	Returns to the previous mode (MPLS O-UNI).
Step 10	**interface** ***type number*** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni)# interface pos0/4/0/1`	Enters interface configuration mode.
Step 11	**lmp data-link adjacency** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if)# lmp data-link adjacency`	Enters LMP data-link adjacency mode.
Step 12	**neighbor** *neigbor-name* **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if-adj)# neighbor router1`	Associates the interface with the specified neighbor. • In this example, POS interface 0/4/0/1 (the configured interface) is associated with the neighbor router1.
Step 13	**remote interface-id** ***interface-id*** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if-adj)# remote interface-id 345.`	Configures the remote data-link interface ID. • In this example, configures POS interface 0/4/0/1 as connected to an interface on neighbor router1, where the interface ID of 345 is assigned.
Step 14	**tna ipv4** ***ip-address*** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if-adj)# tna ipv4 10.5.8.32`	Configures the data-link TNA to the IPv4 address 10.5.8.32.

	Command or Action	Purpose
Step 15	**exit** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if-adj)# exit`	Exits LMP data-link adjacency submode and returns to MPLS Optical-UNI interface submode.
Step 16	**destination address ipv4** ***ip-address*** **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if)# destination address ipv4 50.5.7.4`	Configures the address of the remote end of the O-UNI connection to be established. • In this example, the address 50.5.7.4 corresponds to the TNA address assigned to the destination O-UNI data link.
	passive **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if)# passive`	Configures the router to accept an incoming connection request.

Step 17	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-ouni-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-ouni-if)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Step 18	**show mpls optical-uni** **Example:** `RP/0/RP0/CPU0:router# show mpls optical-uni`	(Optional) Use the **show mpls optical-uni** command to check that the interface connection has been set up (the output should report the interface).

Tearing Down an MPLS O-UNI Connection

Perform this task to tear down an existing MPLS O-UNI connection.

SUMMARY STEPS

1. **configure**
2. **mpls optical-uni**
3. **interface** *type number*
4. **no destination address ipv4** ***ip-address***
   or
   no passive
5. **end**
   or
   commit
6. **show mpls optical-uni**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters configuration mode.
Step 2	**mpls optical-uni** **Example:** `RP/0/RP0/CPU0:router(config)# mpls optical-uni`	Enters O-UNI configuration mode.
Step 3	**interface** *type number* **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni)# interface pos 0/4/0/1`	Enters O-UNI interface configuration mode for the interface identified by *type* and *number*.
Step 4	**no destination address ipv4** *ipaddress* **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if)# no destination address ipv4 50.5.7.4`	Removes the destination address configuration, causing the O-UNI connection to be deleted. If a passive configuration was entered, Step 5 should be used.
	no passive **Example:** `RP/0/RP0/CPU0:router(config-mpls-ouni-if)# no passive`	Removes the passive configuration, causing the deletion of an existing O-UNI connection.

	Command or Action	Purpose

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-mpls-ouni-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-mpls-ouni-if)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Step 6	**show mpls optical-uni** **Example:** `RP/0/RP0/CPU0:router# show mpls optical-uni`	(Optional) Use the **show mpls optical-uni** command to check that the interface connection has been torn-down. The output should not report the interface.

Verifying MPLS O-UNI Configuration

Verify the configuration of the MPLS O-UNI connection by running the **show** commands in this section.

SUMMARY STEPS

1. **show mpls optical-uni lmp neighbor**
2. **show mpls optical-uni lmp**
3. **show mpls optical uni lmp ipcc**
4. **show mpls lmp clients**
5. **show mpls optical-uni lmp interface** *type number*
6. **show mpls optical-uni**
7. **show mpls optical-uni interface** *type number*
8. **show mpls optical-uni diagnostics interface** *type number*

DETAILED STEPS

1. Use the **show mpls optical-uni lmp neighbor** command to display LMP neighbor information, as shown in the following sample output:

   `RP/0/RP0/CPU0:router# show mpls optical-uni lmp neighbor`

```
LMP Neighbor
Name: oxc-uni-n-source, IP: 10.56.57.58, Owner: Optical UNI
 IPCC ID: 1, State Up
  Known via             : Configuration
  Type                  : Routed
  Destination IP        : 10.56.57.58
  Source IP             : None

Data LinkI/F |LclDataLink ID|Link TNA Addr|Data Link LMP state
-------------------------------------------------------------
  POS0/2/0/2              2    10.0.0.5      Up Allocated
```

2. Use the **show mpls optical-uni lmp** command to display LMP information, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls optical-uni lmp

Local OUNI CLI LMP Node ID: 10.56.57.58
(Source: OUNI LMP CLI configuration, I/F: Loopback0)

LMP Neighbor
Name: oxc-uni-n-dest, IP: 10.12.13.14, Owner: Optical UNI
 IPCC ID: 2, State Up
  Known via             : Configuration
  Type                  : Routed
  Destination IP        : 10.12.13.14
  Source IP             : None

Data LinkI/F |LclDataLink ID|Link TNA Addr|Data Link LMP state
-------------------------------------------------------------
  POS0/2/0/2              2    10.0.0.5      Up Allocated
```

3. Use the **show mpls optical uni lmp ipcc** command to display LMP IPCC information, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls optical-uni lmp ipcc

                  IPCC
  ID |    Type       | IP         |  Status |  Neighbor Name
-------------------------------------------------------------
   1      Routed      10.56.57.58      Up     oxc-uni-n-source
```

4. Use the **show mpls lmp clients** command to display information about MPLS LMP clients, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls lmp clients

Current time: Tue Nov  4 13:20:50 2003
Total Number of Clients = 2

Client   | Job ID |   Node    |Uptime|         Since
-------------------------------------------------------------------
ucp_ouni   304      node0_0_0  5m45s   Tue Nov 4 13:15:05 2003
rsvp       261      node0_0_0  5m44s   Tue Nov 4 13:15:06 2003
```

5. Use the **show mpls optical-uni lmp interface** command to display LMP information for a specified interface, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls optical-uni lmp interface pos 0/2/0/2

                          Interface: POS0/2/0/2
                              Owner: Optical UNI
            Local data link ID type: Unnumbered
                 Local data link ID: Hex = 0x2, Dec = 2
```

```
                    TNA address type: IPv4
                         TNA address: 10.0.0.5
  Local TE link switching capability: Packet-Switch Capable
                Remote neighbor name: oxc-uni-n-source
             Remote neighbor node ID: 10.56.57.58
            Remote data link ID type: Unnumbered
                 Remote data link ID: Dec = 2, Hex = 0x2
 Remote TE link switching capability: TDM Capable (TDM)
                 Data link I/F state: Up
                 Data link LMP state: Up/Allocated
                   TE link LMP state: Up
         Data link allocation status: Allocated
                             IPCC ID: 1
                           IPCC type: Routed
         IPCC destination IP address: 10.56.57.58
```

6. Use the **show mpls optical-uni** command to display the state of O-UNI network connections, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls optical-uni

Index of abbreviations:
----------------------
 M=O-UNI configuration Mode.
 P=Passive
 AR =active/receiver
 AS=active/sender
 U=Unknown

Interface     TunID    M    ig State    CCT Up Since
-------------------------------------------------------------
POS0/2/0/2    000004   AS   Connected    04/11/2003 13:16:18
```

7. Use the **show mpls optical-uni interface** command to display detailed O-UNI information for a specific interface, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls optical-uni interface pos 0/2/0/2

Interface POS0/2/0/2
Configuration: Active->User
Signaling State: Connected since 04/11/2003 13:16:18
TNA: 10.0.0.5
Sender NodeID/Tunnel ID: 10.12.13.14/4
Local Data Link ID: 2
Remote Data Link ID: 2
Local Switching Capability: PSC 1
Remote Switching Capability: TDM
Primary IPCC:  Interface: Routed
               Local IP Address:  10.0.0.0
               Remote IP Address: 10.56.57.58
```

8. Use the **show mpls optical-uni diagnostics interface** command to display diagnostics information for an O-UNI connection on a specific interface, as shown in the following sample output:

```
RP/0/RP0/CPU0:router# show mpls optical-uni diagnostics interface pos 0/2/0/2

Interface [POS0/2/0/2]
Configuration: Active->User
Signaling State: [Connected] since 04/11/2003 13:16:18
Connection to OLM/LMP established? Yes
OUNI to OLM/LMP DB sync. status: Synchronized
Connection to RSVP established? Yes
RSVP to OLM/LMP DB sync. status: Synchronized
The neighbor [oxc-uni-n-source] has been configured, and has the node id [10.56.
```

```
57.58]
Found a route to the neighbor [oxc-uni-n-source]
Remote switching capability is TDM.
TNA [10.0.0.5] configured.
All required configs have been entered.
Global Code: No Error/ Success @ unknown time
Datalink Code: No Error/ Success @ unknown time
```

Configuration Examples for MPLS O-UNI

This section provides the following configuration examples:

- MPLS O-UNI Neighbor and Data Link Configuration: Examples, page MPC-210
- O-UNI Connection Establishment: Example, page MPC-211
- O-UNI Connection Tear-Down: Example, page MPC-211

MPLS O-UNI Neighbor and Data Link Configuration: Examples

The following configuration examples are provided in this section:

- O-UNI Router ID Configuration, page MPC-210
- O-UNI-N Neighbor Configuration, page MPC-210
- O-UNI Data Link Configuration, page MPC-210

O-UNI Router ID Configuration

```
configure
 mpls optical-uni
 router-id Loopback 0
 commit
```

O-UNI-N Neighbor Configuration

```
configure
 optical-uni
 lmp neighbor oxc-uni-n-source
 ipcc routed
 remote node-id 10.56.57.58
 commit
```

O-UNI Data Link Configuration

```
configure
 mpls optical-uni
 interface pos 0/2/0/2
 lmp data-link adjacency
 neighbor oxc-uni-n-source
 interface-id 2
 tna ipv4 10.0.0.5
 commit
```

O-UNI Connection Establishment: Example

The following configuration examples are provided in this section:

- O-UNI Connection Configuration at Active Side, page MPC-211
- O-UNI Connection Configuration at Passive Side, page MPC-211

O-UNI Connection Configuration at Active Side

```
configure
 mpls optical-uni
 interface pos 0/2/0/2
 destination address ipv4 10.0.0.7
 commit
```

O-UNI Connection Configuration at Passive Side

```
configure
 mpls optical-uni
 interface pos 0/2/0/2
 passive
 commit
```

O-UNI Connection Tear-Down: Example

The following configuration examples are shown in this section:

- O-UNI Connection Deletion at Active Side, page MPC-211
- O-UNI Connection Deletion at Passive Side, page MPC-211

O-UNI Connection Deletion at Active Side

```
configure
 mpls optical-uni
 interface pos 0/2/0/2
 no destination address ipv4 10.0.0.7
 commit
```

O-UNI Connection Deletion at Passive Side

```
configure
 mpls optical-uni
 interface pos 0/2/0/2
 no passive
 commit
```

Additional References

For additional information related to O-UNI, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR software O-UNI commands	*MPLS Optical User Network Interface Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco IOS XR software RSVP commands	*MPLS RSVP Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco IOS XR software RSVP configuration guide	*Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco CRS-1 router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
OIF UNI 1.0	*User Network Interface (UNI) 1.0 Signaling Specification*

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs[1]	Title
RFC 3471	*Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description*
RFC 3473	*Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions*
draft-ietf-ccamp-gmpls-sonet-sdh-*xx*.txt	*Generalized Multi-Protocol Label Switching Extensions for SONET and SDH Control*
LMP IETF draft	*Link Management Protocol (LMP)* http://www.ietf.org/internet-drafts/draft-ietf-ccamp-lmp-10.txt
draft-ietf-ccamp-gmpls-architecture-*xx*.txt	*Generalized Multi-Protocol Label Switching Architecture*
draft-ietf-ccamp-lmp-*xx*.txt	*Link Management Protocol (LMP)*

1. Not all supported RFCs are listed.

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software

This module provides the conceptual and configuration information for MPLS Layer 2 virtual private networks (VPNs) on Cisco IOS XR software.

For the functionality of MPLS VPNs over IP Tunnels, see Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software in *Cisco IOS XR MPLS Configuration Guide*.

Note For more information about MPLS Layer 2 VPN on the Cisco IOS XR software and for descriptions of the commands listed in this module, see the "Related Documents" section. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing MPLS Layer 2 VPN on Cisco IOS XR Configuration Module

Release	Modification
Release 3.4.0	This feature was introduced on the Cisco CRS-1 and Cisco XR 12000 Series Router.
Release 3.4.1	Support was added for: • Virtual Circuit Connection Verification (VCCV) on L2VPN • Layer 2 VPN (L2VPN) Quality of Service (QoS) for Ethernet-over-MPLS (EoMPLS) on the Cisco CRS-1 • QinQ mode and QinAny mode for EoMPLS on the Cisco XR 12000 Series Router
Release 3.5.0	Support was added for: • EoMPLS Inter-AS mode • Mac-in-Mac protocol

Release 3.6.0	Support was added for: • Ethernet Remote Port Shutdown • Preferred Tunnel Path on the Cisco CRS-1,
Release 3.7.0	Support was added for: • ATM over MPLS (ATMoMPLS) with Layer 2VPN capability on the Cisco CRS-1 and the Cisco XR 12000 Series Router. • Static point-to-point cross-connects procedure was modified. • Static point-to-point cross-connect configuration example was modified.

Contents

- Prerequisites for Implementing MPLS L2VPN on Cisco IOS XR Software, page MPC-216
- Information About Implementing L2VPN, page MPC-216
- How to Implement L2VPN, page MPC-224
- Configuration Examples for L2VPN, page MPC-235
- Additional References, page MPC-238

Prerequisites for Implementing MPLS L2VPN on Cisco IOS XR Software

You must be in a user group associated with a task group that includes the proper task IDs for MPLS L2VPN commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*.

Information About Implementing L2VPN

To implement MPLS L2VPN, you should understand the following concepts:

- L2VPN Overview, page MPC-217
- ATMoMPLS with L2VPN Capability, page MPC-217
- Virtual Circuit Connection Verification on L2VPN, page MPC-218
- Ethernet over MPLS, page MPC-218
- Quality of Service, page MPC-222
- High Availability, page MPC-223
- Preferred Tunnel Path, page MPC-223

L2VPN Overview

Layer 2 VPN (L2VPN) emulates the behavior of a LAN across an IP or MPLS-enabled IP network allowing Ethernet devices to communicate with each other as they would when connected to a common LAN segment.

As Internet service providers (ISPs) look to replace their Frame Relay or Asynchronous Transfer Mode (ATM) infrastructures with an IP infrastructure, there is a need for to provide standard methods of using an IP infrastructure to provide a serviceable L2 interface to customers; specifically, to provide standard ways of using an IP infrastructure to provide virtual circuits between pairs of customer sites.

Building a L2VPN system requires coordination between the ISP and the customer. The ISP provides L2 connectivity; the customer builds a network using data link resources obtained from the ISP. In an L2VPN service, the ISP does not require information about a the customer's network topology, policies, routing information, point-to-point links, or network point-to-point links from other ISPs.

The ISP requires provider edge (PE) routers with the following capabilities:

- Encapsulation of L2 protocol data units (PDU) into Layer 3 (L3) packets.
- Interconnection of any-to-any L2 transports.
- Emulation of L2 quality-of-service (QoS) over a packet switch network.
- Ease of configuration of the L2 service.
- Support for different types of tunneling mechanisms (MPLS, L2TPv3, IPSec, GRE, and others).
- L2VPN process databases include all information related to circuits and their connections.

ATMoMPLS with L2VPN Capability

Note This feature is supported on the Cisco CRS-1 router and the Cisco XR 12000 Series Router.

These topics describe the ATM over MPLS (ATMoMPLS) with L2VPN feature:

- ATMoMPLS with L2VPN Overview, page MPC-217
- Layer 2 Local Switching Overview, page MPC-218
- ATM Adaptation Layer 5, page MPC-218

ATMoMPLS with L2VPN Overview

The ATMoMPLS feature supports ATM Adaptation Layer 5 (AAL5) transport. ATMoMPLS is a type of Layer 2 point-to-point connection over an MPLS core. ATMoMPLS and ATM local switching are supported only for ATM-to-ATM interface-to-interface switching combinations.

To implement the ATMoMPLS feature, the Cisco CRS-1 router plays the role of provider edge (PE) router at the edge of a provider network in which customer edge (CE) devices are connected to the Cisco CRS-1 routers.

Layer 2 Local Switching Overview

Local switching lets you to switch Layer 2 data between two interfaces of the same type (for example, ATM-to-ATM, or Frame Relay-to-Frame Relay) or between interfaces of different types (for example, Frame Relay to ATM) on the same router. The interfaces are on the same line card or on two different cards. During these types of switching, Layer 2 address is used instead of the Layer 3 address.

In addition, same-port local switching lets you to switch Layer 2 data between two circuits on the same interface.

ATM Adaptation Layer 5

AAL5 lets you transport AAL5 PDUs from various customers over an MPLS backbone. ATM AAL5 extends the usability of the MPLS backbone by enabling it to offer Layer 2 services in addition to already existing Layer 3 services. You can enable the MPLS backbone network to accept AAL5 PDUs by configuring the provider edge (PE) routers at both ends of the MPLS backbone.

To transport AAL5 PDUs over MPLS, a virtual circuit is set up from the ingress PE router to the egress PE router. This virtual circuit transports the AAL5 PDUs from one PE router to the other. Each AAL5 PDU is transported as a single packet.

Virtual Circuit Connection Verification on L2VPN

Virtual Circuit Connection Verification (VCCV) is an L2VPN Operations, Administration, and Maintenance (OAM) feature that allows network operators to run IP-based provider edge-to-provider edge (PE-to-PE) keepalive protocol across a specified pseudowire to ensure that the pseudowire data path forwarding does not contain any faults. The disposition PE receives VCCV packets on a control channel, which is associated with the specified pseudowire. The control channel type and connectivity verification type, which are used for VCCV, are negotiated when the pseudowire is established between the PEs for each direction.

Two types of packets can arrive at the disposition egress:

- Type 1—Specifies normal Ethernet-over-MPLS (EoMPLS) data packets.
- Type 2—Specifies VCCV packets.

Cisco IOS XR software supports Label Switched Path (LSP) VCCV Type 1, which uses an inband control word if enabled during signaling. The VCCV echo reply is sent as IPv4 that is the reply mode is IPv4. The reply is forwarded as IP, MPLS, or a combination of both.

VCCV pings counters that are counted in MPLS forwarding on the egress side. However, on the ingress side, they are sourced by the route processor and do not count as MPLS forwarding counters.

Ethernet over MPLS

Ethernet-over-MPLS (EoMPLS) provides a tunneling mechanism for Ethernet traffic through an MPLS-enabled L3 core and encapsulates Ethernet protocol data units (PDUs) inside MPLS packets (using label stacking) to forward them across the MPLS network.

EoMPLS features are described in the following subsections:

- Ethernet Port Mode, page MPC-219
- Ethernet Remote Port Shutdown, page MPC-219

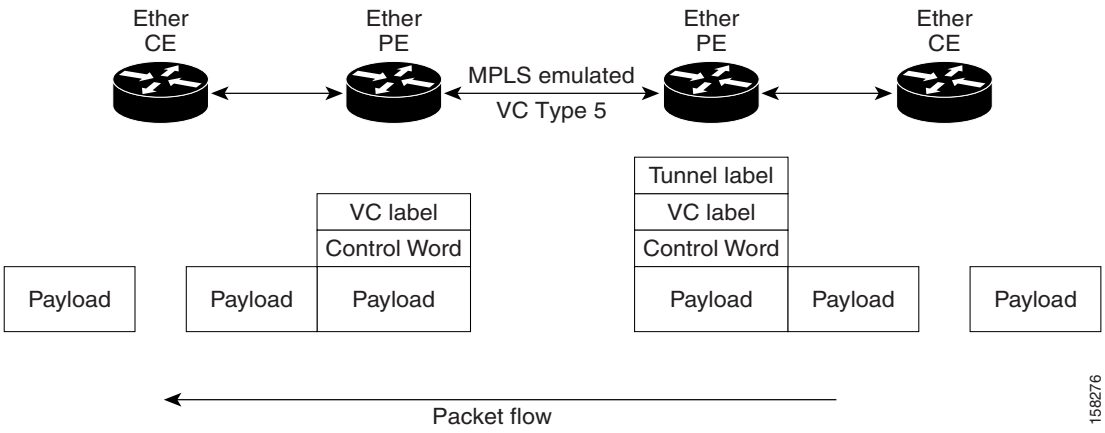
- VLAN Mode, page MPC-220
- Inter-AS Mode, page MPC-221
- QinQ Mode, page MPC-221
- QinAny Mode, page MPC-222
- Mac-in-Mac Protocol (Provide Backbone Bridging), page MPC-222

Ethernet Port Mode

In Ethernet port mode, both ends of a pseudowire are connected to Ethernet ports. In this mode, the port is tunneled over the pseudowire or, using local switching (also known as an *attachment circuit-to-attachment circuit cross-connect*) switches packets or frames from one attachment circuit (AC) to another AC attached to the same PE node.

Figure 15 provides an example of Ethernet port mode.

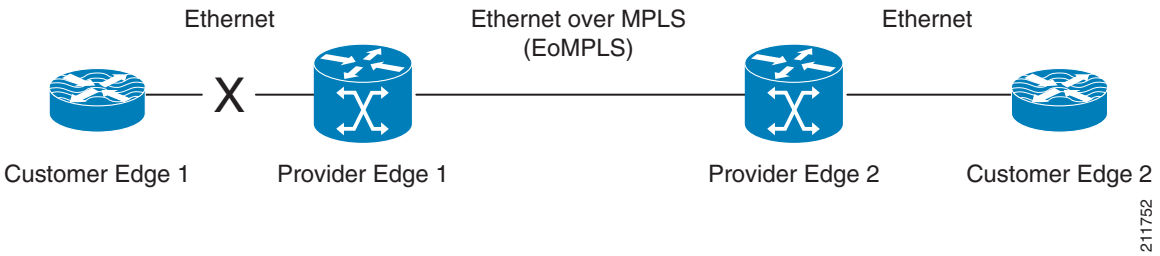
Figure 15 Ethernet Port Mode Packet Flow



Ethernet Remote Port Shutdown

Ethernet remote port shutdown provides a mechanism for the detection and propagation of remote link failure for port mode EoMPLS on a Cisco CRS-1 line card. This lets a service provider edge router on the local end of an Ethernet-over-MPLS (EoMPLS) pseudowire detect a cross-connect or remote link failure and cause the shutdown of the Ethernet port on the local customer edge router. Shutting down the Ethernet port on the local customer edge router prevents or mitigates a condition where that router would otherwise lose data by forwarding traffic continuously to the remote failed link, especially if the link were configured as a static IP route (see Figure 16).

Figure 16 Remote Link Outage in EoMPLS Wide Area Network



To enable this functionality, see the **l2transport propagate** command in *Cisco IOS XR MPLS Command Reference*.

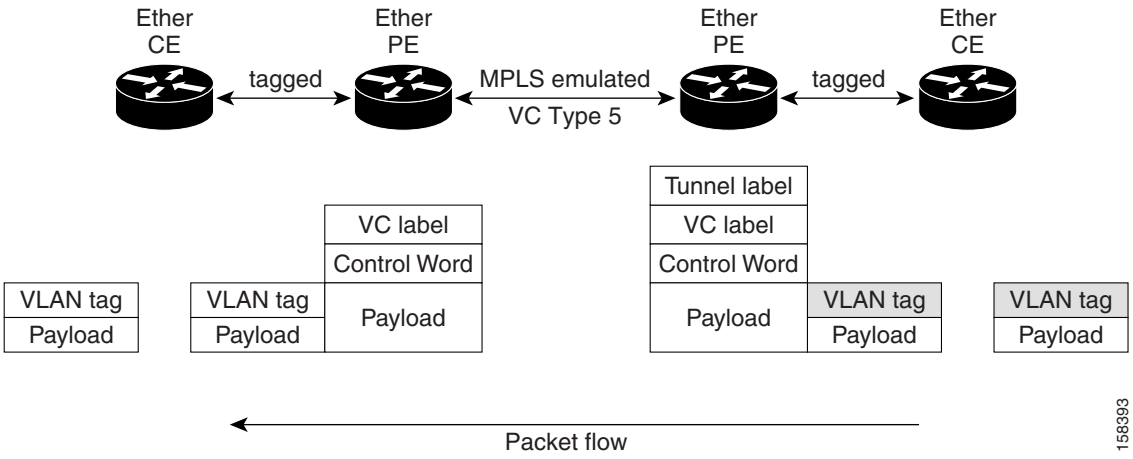
Note Ethernet remote port shutdown is supported only on the Cisco CRS-1 router.

VLAN Mode

In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4 or VC type 5. VC type 5 is the default mode.

As illustrated in Figure 17, the Ethernet PE associates an internal VLAN-tag to the Ethernet port for switching the traffic internally from the ingress port to the pseudowire; however, before moving traffic into the pseudowire, it removes the internal VLAN tag.

Figure 17 VLAN Mode Packet Flow



At the egress VLAN PE, the PE associates a VLAN tag to the frames coming off of the pseudowire and after switching the traffic internally, it sends out the traffic on an Ethernet trunk port.

Note Because the port is in trunk mode, the VLAN PE doesn't remove the VLAN tag and forwards the frames through the port with the added tag.

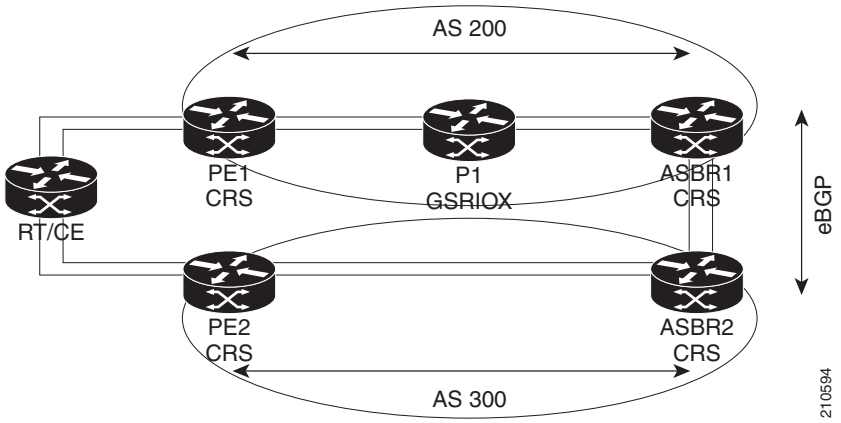
Inter-AS Mode

Inter-AS is a peer-to-peer type model that allows extension of VPNs through multiple provider or multi-domain networks. This lets service providers peer up with one another to offer end-to-end VPN connectivity over extended geographical locations.

EoMPLS support can assume a single AS topology where the pseudowire connecting the PE routers at the two ends of the point-to-point EoMPLS cross-connects resides in the same autonomous system; or multiple AS topologies in which PE routers can reside on two different ASs using i-BGP and e-BGP peering.

Figure 18 illustrates MPLS over Inter-AS with a basic double AS topology with iBGP/LDP in each AS.

Figure 18 EoMPLS over Inter-AS: Basic Double AS Topology

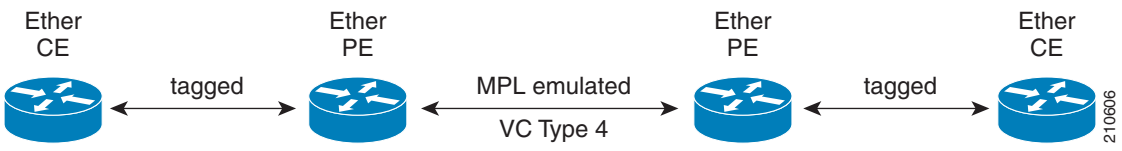


QinQ Mode

In QinQ mode, each CE VLAN is carried into an SP VLAN. QinQ mode should use VC type 5, but VC type 4 is also supported. On each Ethernet PE, you must configure both the inner (CE VLAN) and outer (SP VLAN).

Figure 19 illustrates QinQ using VC type 4.

Figure 19 EoMPLS over QinQ Mode



QinAny Mode

In the QinAny mode, the service provider VLAN tag is configured on both the ingress and the egress nodes of the provider edge VLAN. QinAny mode is similar to QinQ mode using a Type 5 VC, except that the customer edge VLAN tag is carried in the packet over the pseudowire, as the customer edge VLAN tag is unknown.

Note The QinAny mode is supported on the Cisco XR 12000 Series Router only.

Mac-in-Mac Protocol (Provide Backbone Bridging)

The Mac-in-Mac (or, Provider Backbone Bridging) protocol lets service providers scale networks using Ethernet technology to maintain management and operational simplicity, and reduce operating costs.

Mac-In-Mac encapsulates the customer MAC header with a service provider MAC header. Instead of using additional Q-tags to separate end customers, a 24-bit service tag in the service provider encapsulating MAC header is used, which provides support for up to 16-million service instances.

Note Mac-In-Mac is standardized as IEEE 802.1ah.

Quality of Service

Using L2VPN technology, you can assign a quality of service (QoS) level to both Port and VLAN modes of operation.

L2VPN technology requires that QoS functionality on PE routers be strictly L2-payload-based on the edge-facing interfaces (also know as *attachment circuits*). Figure 20 illustrates L2 and L3 QoS service policies in a typical L2VPN network.

Figure 20 L2VPN QoS Feature Application

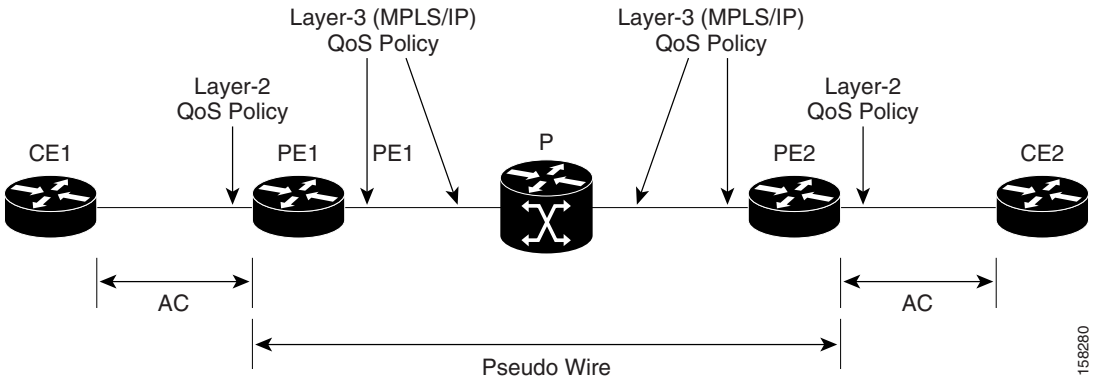
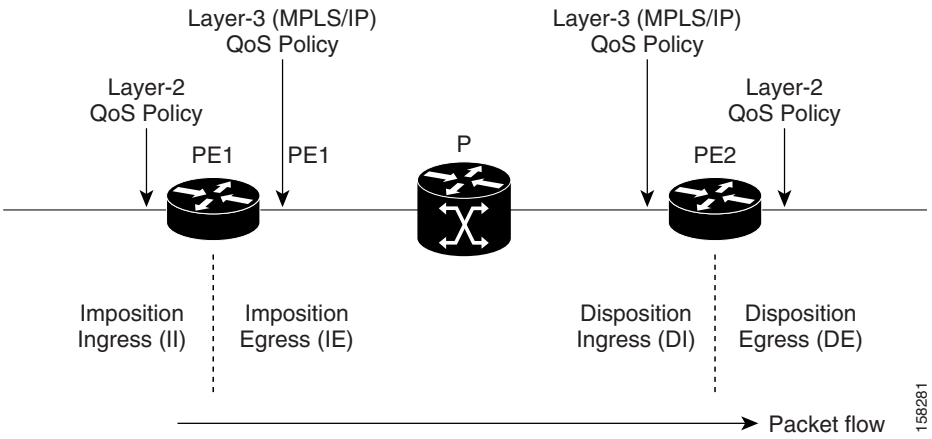


Figure 21 shows four packet processing paths within a provider edge device where a QoS service policy can be attached. In an L2VPN network, packets are received and transmitted on the edge-facing interfaces as L2 packets and transported on the core-facing interfaces as MPLS (EoMPLS) or IP (L2TP) packets.

Figure 21 L2VPN QoS Reference Model



High Availability

L2VPN uses control planes in both route processors and line cards, as well as forwarding plane elements in the line cards.

Note The l2tp_mgr process does not support high availability.

The availability of L2VPN meets the following requirements:

- A control plane failure in either the route processor or the line card will not affect the circuit forwarding path.
- The router processor control plane supports failover without affecting the line card control and forwarding planes.
- L2VPN integrates with existing Label Distribution Protocol (LDP) graceful restart mechanism.

Preferred Tunnel Path

Preferred tunnel path functionality lets you map pseudowires to specific traffic-engineering tunnels. Attachment circuits are cross-connected to specific MPLS traffic engineering tunnel interfaces instead of remote PE router IP addresses (reachable using IGP or LDP). Using preferred tunnel path, it is always assumed that the traffic engineering tunnel that transports the L2 traffic runs between the two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router).

Note

- Currently, preferred tunnel path configuration applies only to MPLS encapsulation.
- The fallback enable option is supported only on the Cisco XR 12000 Series Router.

How to Implement L2VPN

This section describes the tasks required to implement L2VPN:

- Configuring an Interface or Connection for L2VPN, page MPC-224
- Configuring Static Point-to-Point Cross-Connects, page MPC-226
- Configuring Dynamic Point-to-Point Cross-Connects, page MPC-228
- Configuring Inter-AS, page MPC-230
- Configuring L2VPN Quality of Service, page MPC-230
- Configuring Preferred Tunnel Path, page MPC-234

Configuring an Interface or Connection for L2VPN

Perform this task to configure an interface or a connection for L2VPN.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id*
3. **l2transport**
4. **exit**
5. **interface** *type interface-id*
6. **dot1q native vlan** *vlan-id*
7. **end**
   or
   commit
8. **show interface** *type interface-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/0/0/0`	Enters interface configuration mode and configures an interface.

	Command or Action	Purpose
Step 3	**l2transport** **Example:** `RP/0/RP0/CPU0:router(config-if)# l2transport`	Enables L2 transport on the selected interface.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if-l2)# exit`	Exits the current configuration mode.
Step 5	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0/0`	Enters interface configuration mode and configures an interface.
Step 6	**dot1q native vlan** *vlan ID* **Example:** `RP/0/RP0/CPU0:router(config-if)# dot1q vlan 1`	Assigns the native VLAN ID of a physical interface trunking 802.1Q VLAN traffic.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:show interface GigabitEthernet0/0/0/0`	Displays the configuration settings you committed for the interface.

Configuring Static Point-to-Point Cross-Connects

Perform this task to configure static point-to-point cross-connects.

Please consider the following information about cross-connects when you configure static point-to-point cross-connects:

- An cross-connect is uniquely identified with the pair; the cross-connect name must be unique within a group.
- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect.
- A static VC local label is globally unique and can be used in one pseudowire only.
- No more than 16,000 cross-connects can be configured per router.

Note Static pseudowire connections do not use LDP for signaling.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group name*
4. **p2p** *xconnect name*
5. **interface** *type interface-id*
6. **neighbor** *A.B.C.D* **pw-id** *pseudowire ID*
7. **mpls static label local** {*value*} **remote** {*value*}
8. **end**
   or
   commit
9. **show l2vpn xconnect group** *group name*

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`l2vpn`
	
	Example:
	`RP/0/RP0/CPU0:router(config)# l2vpn`
		Enters L2VPN configuration mode.

	Command or Action	Purpose
Step 3	**xconnect group** *group name* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group vlan_grp_1`	Enters the name of the cross-connect group.
Step 4	**p2p** *xconnect name* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p vlan1`	Enters a name for the point-to-point cross-connect.
Step 5	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1`	Specifies the interface type ID. The choices are: • GigabitEthernet: GigabitEthernet/IEEE 802.3 interfaces. • TenGigE: TenGigabitEthernet/IEEE 802.3 interfaces.
Step 6	**neighbor** *A.B.C.D* **pw-id** *pseudowire ID* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000`	Configures the pseudowire segment for the cross-connect. Optionally, you can disable the control word or set the transport-type to "Ethernet" or "VLAN".
Step 7	**mpls static label local** {*value*} **remote** {*value*} **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# mpls static label local 699 remote 890`	Configures local and remote label ID values.

	Command or Action	Purpose

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**show l2vpn xconnect group** *group name* **Example:** `RP/0/RP0/CPU0:show l2vpn xconnect group p2p`	Displays the name of the Point-to-Point cross-connect group you created.

Configuring Dynamic Point-to-Point Cross-Connects

Perform this task to configure dynamic point-to-point cross-connects.

Note For dynamic cross-connects, LDP must be up and running.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group name*
4. **p2p** *xconnect name*
5. **interface** *type interface-id*
6. **neighbor** *A.B.C.D* **pw-id** *pseudowire ID*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/RP0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.
Step 3	**xconnect group** *group name* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group grp_1`	Enters the name of the cross-connect group.
Step 4	**p2p** *xconnect name* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p vlan1`	Enters a name for the point-to-point cross-connect.
Step 5	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1`	Specifies the interface type ID. The choices are: • GigabitEthernet: GigabitEthernet/IEEE 802.3 interfaces. • TenGigE: TenGigabitEthernet/IEEE 802.3 interfaces.

	Command or Action	Purpose
Step 6	**neighbor** *A.B.C.D* **pw-id** *pseudowire ID* **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000`	Configures the pseudowire segment for the cross-connect. Optionally, you can disable the control word or set the transport-type to "Ethernet" or "vlan".

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# end`
	or
	`RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Inter-AS

The Inter-AS configuration procedure is identical to the L2VPN cross-connect configuration tasks (see "Configuring Static Point-to-Point Cross-Connects" section on page MPC-226 and "Configuring Dynamic Point-to-Point Cross-Connects" section on page MPC-228) except that the remote PE IP address used by the cross-connect configuration is now reachable through iBGP peering.

Note You must be knowledgeable about IBGP, EBGP, and ASBR terminology and configurations to complete this configuration.

Configuring L2VPN Quality of Service

This section describes how to configure L2VPN quality of service (QoS) in port mode and VLAN mode.

Restrictions

The **l2transport** command cannot be used with any IP address, L3, or CDP configuration.

Configuring an L2VPN Quality of Service Policy in Port Mode

This procedure describes how to configure an L2VPN QoS policy in port mode.

Note In port mode, the interface name format does not include a subinterface number; for example, GigabitEthernet0/1/0/1.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id.subinterface*
3. **l2transport**
4. **service-policy** [**input** | **output**] [*policy-map-name*]
5. **end**
   or
   commit
6. **show qos interface** *type interface-path-id.subinterface* **service-policy** [**input** | **output**] [*policy-map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**interface** *type interface-id.subinterface* **Example:** `RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0/0.1`	Specifies the interface attachment circuit.
Step 3	**l2transport** **Example:** `RP/0/RP0/CPU0:router(config-if)# l2transport`	Configures an interface or connection for L2 switching.
Step 4	**service-policy** *[***input**	**output***] [policy-map-name]* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy input servpol1`	Attaches a QoS policy to an input or output interface to be used as the service policy for that interface.

	Command or Action	Purpose

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**show qos interface** *type interface-path-id.subinterface* **service-policy [input | output]** *[policy-map-name]* **Example:** `RP/0/RP0/CPU0:router# show qos interface GigabitEthernet0/0/0/0.1 input servpol1`	Displays the QoS service policy you defined.

Configuring an L2VPN Quality of Service Policy in VLAN Mode

This procedure describes how to configure a L2VPN QoS policy in VLAN mode.

Note In VLAN mode, the interface name must include a subinterface; for example, GigabitEthernet0/1/0/1.1; and the l2transport command must follow the interface type on the same CLI line (for example, "interface GigabitEthernet0/0/0/0.1 l2transport").

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id.subinterface* **l2transport**
3. **service-policy** [**input** | **output**] [*policy-map-name*]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**interface** *type interface-id.subinterface* **l2transport** **Example:** `RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0/0.1 l2transport`	Configures an interface or connection for L2 switching. **Note** In VLAN Mode, you must enter the **l2transport** keyword on the same line as the interface.
Step 3	**service-policy** *[***input** | **output***] [policy-map-name]* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy input servpol1`	Attaches a QoS policy to an input or output interface to be used as the service policy for that interface.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Preferred Tunnel Path

This procedure describes how to configure a preferred tunnel path.

Note The tunnel used for the preferred path configuration is an MPLS Traffic Engineering (MPLS-TE) tunnel.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** {*name*}
4. **encapsulation mpls**
5. **preferred-path** {**interface**} {**tunnel-te** *value*} [**fallback disable**]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/RP0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.
Step 3	**pw-class** {*name*} **Example:** `RP/0/RP0/CPU0:router(config-l2vpn)# pw-class path1`	Configures the pseudowire class name.
Step 4	**encapsulation mpls** **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls`	Configures the pseudowire encapsulation to MPLS.
Step 5	**preferred-path** {**interface**} {**tunnel-te** *value*} [**fallback disable**] **Example:** `RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel 11 fallback disable`	Configures preferred path tunnel settings. If the fallback disable configuration is used and once the TE tunnel is configured as the preferred path goes down, the corresponding pseudowire can also go down.

	Command or Action	Purpose

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# end`
	
	or
	
	`RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-mpls-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 7	**show l2vpn all**	

Configuration Examples for L2VPN

In the following example, two traffic classes are created and their match criteria are defined. For the first traffic class called class1, ACL 101 is used as the match criterion. For the second traffic class called class2, ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

This section includes the following configuration examples:

- L2VPN Interface Configuration: Example, page MPC-236
- Point-to-Point Cross-connect Configuration: Examples, page MPC-236
- Inter-AS: Example, page MPC-236
- L2VPN Quality of Service: Example, page MPC-238
- Preferred Path: Example, page MPC-238

L2VPN Interface Configuration: Example

The following example shows how to configure an L2VPN interface:

```
configure
 interface GigabitEthernet0/0/0/0.1 l2transport
 dot1q vlan 1
 end
```

Point-to-Point Cross-connect Configuration: Examples

This section includes configuration examples for both static and dynamic point-to-point cross-connects.

Static Configuration

The following example shows how to configure a static point-to-point cross-connect:

```
configure
 l2vpn
  xconnect group vlan_grp_1
   p2p vlan1
   interface GigabitEthernet0/0/0/0.1
   neighbor 2.2.2.2 pw-id 2000
    mpls static label local 699 remote 890
    commit
```

Dynamic Configuration

The following example shows how to configure a dynamic point-to-point cross-connect:

```
configure
 l2vpn
  xconnect group vlan_grp_1
   p2p vlan1
   interface GigabitEthernet0/0/0/0.1
   neighbor 2.2.1.1 pw-id 1
   commit
```

Inter-AS: Example

The following example shows how to set up an AC to AC cross connect from AC1 to AC2:

```
router-id Loopback0

interface Loopback0
 ipv4 address 127.0.0.1 255.255.255.255
!
interface GigabitEthernet0/1/0/0.1 l2transport dot1q vlan 1!
!
interface POS0/0/0/3
 ipv4 address 127.0.0.1 255.255.255.0
 keepalive disable
!
interface POS0/0/0/4
 ipv4 address 127.0.0.1 255.255.255.0
 keepalive disable
!
router ospf 100
```

```
 log adjacency changes detail
 area 0
  interface Loopback0
  !
  interface POS0/0/0/3
  !
  interface POS0/0/0/4
  !
 !
!
router bgp 100
 address-family ipv4 unicast
  allocate-label all
 !
 neighbor 40.0.0.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family ipv4 labeled-unicast
  !
 !
!
l2vpn
 xconnect group xc1
  p2p ac2ac1
   interface GigabitEthernet0/1/0/0.1
   neighbor 20.0.0.5 pw-id 101
  !
  p2p ac2ac2
   interface GigabitEthernet0/1/0/0.2
   neighbor 20.0.0.5 pw-id 102
  !
  p2p ac2ac3
   interface GigabitEthernet0/1/0/0.3
   neighbor 20.0.0.5 pw-id 103
  !
  p2p ac2ac4
   interface GigabitEthernet0/1/0/0.4
   neighbor 20.0.0.5 pw-id 104
  !
  p2p ac2ac5
   interface GigabitEthernet0/1/0/0.5
   neighbor 20.0.0.5 pw-id 105
  !
  p2p ac2ac6
   interface GigabitEthernet0/1/0/0.6
   neighbor 20.0.0.5 pw-id 106
  !
  p2p ac2ac7
   interface GigabitEthernet0/1/0/0.7
   neighbor 20.0.0.5 pw-id 107
  !
  p2p ac2ac8
   interface GigabitEthernet0/1/0/0.8
   neighbor 20.0.0.5 pw-id 108
  !
  p2p ac2ac9
   interface GigabitEthernet0/1/0/0.9
   neighbor 20.0.0.5 pw-id 109
  !
  p2p ac2ac10
   interface GigabitEthernet0/1/0/0.10
   neighbor 20.0.0.5 pw-id 110
```

```
  !
 !
!
mpls ldp
 router-id Loopback0
 log
  neighbor
 !
 interface POS0/0/0/3
 !
 interface POS0/0/0/4
 !
!
end
```

L2VPN Quality of Service: Example

The following example shows how to attach a service-policy to an L2 interface in port mode:

```
configure
  interface type interface-id
  l2transport
  service-policy [input | output] [policy-map-name]
commit
```

Preferred Path: Example

The following example shows how to configure preferred tunnel path:

```
configure
 l2vpn
 pw-class path1
  encapsulation mpls
   preferred-path interface tunnel value fallback disable
```

Additional References

For additional information related to implementing MPLS Layer 2 VPN, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS VPN-related commands	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*

Related Topic	Document Title
MPLS Layer 3 VPNs	*Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
Cisco CRS-1 router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 4447	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*, April 2006
RFC 4448	*Encapsulation Methods for Transport of Ethernet over MPLS Networks*, April 2006

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing Virtual Private LAN Services on Cisco IOS XR Software

This module provides the conceptual and configuration information for Virtual Private LAN Services (VPLS) on Cisco IOS XR software. VPLS supports Layer 2 VPN technology and provides transparent multipoint Layer 2 connectivity for customers.

This approach enables service providers to host a multitude of new services such as broadcast TV, Layer 2 VPNs, and so forth.

For MPLS Layer 2 virtual private networks (VPNs), see Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software module.

Note For more information about MPLS Layer 2 VPN on Cisco IOS XR software and for descriptions of the commands listed in this module, see the "Related Documents" section. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing Virtual Private LAN Services on Cisco IOS XR Configuration Module

Release	Modification
Release 3.7.0	This feature was introduced on the Cisco XR 12000 Series Router.

Contents

- Prerequisites for Implementing Virtual Private LAN Services on Cisco IOS XR Software, page MPC-242
- Restrictions for Implementing Virtual Private LAN Services on Cisco IOS XR Software, page MPC-242
- Information About Implementing Virtual Private LAN Services, page MPC-242
- How to Implement Virtual Private LAN Services, page MPC-247
- Configuration Examples for Virtual Private LAN Services, page MPC-278
- Additional References, page MPC-280

Prerequisites for Implementing Virtual Private LAN Services on Cisco IOS XR Software

Before you configure VPLS, ensure that the network is configured as follows:

- Configure IP routing in the core so that the provider edge (PE) routers can reach each other through IP.
- Configure MPLS and Label Distribution Protocol (LDP) in the core so that a label switched path (LSP) exists between the PE routers.
- Configure a loopback interface to originate and terminate Layer 2 traffic. Make sure that the PE routers can access the other router's loopback interface.

Note The loopback interface is not needed in all cases. For example, tunnel selection does not need a loopback interface when VPLS is directly mapped to a TE tunnel.

You must be in a user group associated with a task group that includes the proper task IDs for MPLS L2VPN commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module in the *Cisco IOS XR System Security Configuration Guide*.

Restrictions for Implementing Virtual Private LAN Services on Cisco IOS XR Software

The following Engine 3 hardware restrictions are listed for implementing VPLS:

- All attachment circuits in a bridge domain on an Engine 3 line card must be the same type (for example, port, dot1q, qinq, or qinany), value (VLAN ID), and EtherType (for example, 0x8100, 0x9100, or 0x9200).
- The same line card cannot simultaneously have attachment circuits and MPLS-enabled on any one of its interfaces. The line card cannot be Edge-facing and Core-facing at the same time.
- The line card requires ternary content addressable memory (TCAM) Carving configuration.

For the Engine 5 line card, version 1 of the Ethernet SPA does not support QinQ mode and QinAny mode.

Note For the Engine 5 line card, version 2 of the Ethernet SPA supports all VLAN modes, such as VLAN mode, QinQ mode, or QinAny mode.

Information About Implementing Virtual Private LAN Services

To implement Virtual Private LAN Services (VPLS), you should understand the following concepts:

- Virtual Private LAN Services Overview, page MPC-243
- VPLS for an MPLS-based Provider Core, page MPC-243
- Signaling, page MPC-244
- Bridge Domain, page MPC-244

- MAC Address-related Parameters, page MPC-244
- LSP Ping over VPWS and VPLS, page MPC-247

Virtual Private LAN Services Overview

Virtual Private LAN Service (VPLS) enables geographically separated local-area network (LAN) segments to be interconnected as a single bridged domain over an MPLS network. The full functions of the traditional LAN such as MAC address learning, aging, and switching are emulated across all the remotely connected LAN segments that are part of a single bridged domain. A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

VPLS transports Ethernet 802.3, VLAN 802.1q, and VLAN-in-VLAN (Q-in-Q) traffic across multiple sites that belong to the same Layer 2 broadcast domain. VPLS offers simple Virtual LAN services that include flooding broadcast, multicast, and unknown unicast frames that are received on a bridge. The VPLS solution requires a full mesh of pseudowires that are established among provider edge (PE) routers. The VPLS implementation is based on Label Distribution Protocol (LDP)-based pseudowire signaling.

VPLS is based on the characteristic of virtual forwarding instance (VFI). A VFI is a virtual bridge port that is capable of performing native bridging functions, such as forwarding, based on the destination MAC address, source MAC address learning and aging, and so forth.

After provisioning attachment circuits, neighbor relationships across the MPLS network for this specific instance are established through a set of manual commands identifying the end PEs. When the neighbor association is complete, a full mesh of pseudowires is established among the network-facing provider edge devices, which is a gateway between the MPLS core and the customer domain.

Now, the service provider network starts switching the packets within the bridged domain specific to the customer by looking at destination MAC addresses. All traffic with unknown, broadcast, and multicast destination MAC addresses is flooded to all the connected customer edge devices, which connect to the service provider network. The network-facing provider edge devices learn the source MAC addresses as the packets are flooded. The traffic is unicasted to the customer edge device for all the learned MAC addresses.

VPLS requires the provider edge device to be MPLS-capable. The VPLS provider edge device holds all the VPLS forwarding MAC tables and Bridge Domain information. In addition, it is responsible for all flooding broadcast frames and multicast replications.

VPLS for an MPLS-based Provider Core

VPLS is a multipoint Layer 2 VPN technology that connects two or more customer devices using bridging techniques. The VPLS architecture allows for the end-to-end connection between the Provider Edge (PE) routers to provide Multipoint Ethernet Services.

From an end-user perspective, VPLS emulates an Ethernet switch. VPLS requires the creation of a bridge domain (Layer 2 broadcast domain) on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment circuits.

The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

The MPLS/IP provider core simulates a virtual bridge that connects the multiple attachment circuits on each of the PE devices together to form a single broadcast domain. This also requires all of the PE routers that are participating in a VPLS instance to form emulated VCs among them.

A VFI is created on the PE router for each VPLS instance. The PE routers make packet-forwarding decisions by looking up the VFI of a particular VPLS instance. The VFI acts like a virtual bridge for a given VPLS instance. More than one attachment circuit belonging to a given VPLS are connected to the VFI. The PE router establishes emulated VCs to all the other PE routers in that VPLS instance and attaches these emulated VCs to the VFI. Packet forwarding decisions are based on the data structures maintained in the VFI.

Signaling

An important aspect of VPN technologies, including VPLS, is the ability of network devices to automatically signal to other devices about an association with a particular VPN, often referred to as *signaling mechanisms*. For VPLS, this includes discovery of other peers and MAC address withdrawal.

Note Border Gateway Protocol (BGP) auto-discovery and signaling are not supported.

The implementation of VPLS in a network requires the establishment of a full mesh of pseudowires between the provider edge (PE) routers. The signaling of pseudowires between provider edge devices, described in *draft-ietf-l2vpn-vpls-ldp-09*, uses targeted LDP sessions to exchange label values and attributes and to setup the pseudowires. LDP is an efficient mechanism for signaling pseudowire status for Ethernet point-to-point and multipoint services.

Bridge Domain

The native bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports (including VFI). Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain. A learned address is aged out. Incoming frames are mapped to a bridge domain, based on either the ingress port or a combination of both an ingress port and a MAC header field.

By default, split horizon is enabled on a bridge domain. In other words, any packets that are coming on either the attachment circuits or pseudowires are not returned on the same attachment circuits or pseudowires. In addition, the packets that are received on one pseudowire are not replicated on other pseudowires in the same VFI.

MAC Address-related Parameters

The MAC address table contains a list of the known MAC addresses and their forwarding information. In the current VPLS design, the MAC address table and its management are distributed. In other words, a copy of the MAC address table is maintained on the Route Processor (RP) card and the line cards. The RP card manages the master-copy of the MAC table, and is responsible to insert or delete the MAC addresses from the table and to distribute the new information to all line cards.

These topics provide information about the MAC address-related parameters:

- MAC Address Flooding, page MPC-245
- MAC Address-based Forwarding, page MPC-245
- MAC Address Source-based Learning, page MPC-245
- MAC Address Aging, page MPC-245
- MAC Address Limit, page MPC-246
- MAC Address Withdrawal, page MPC-246

MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To obtain flooding within VPLS broadcast models, all unknown unicast, broadcast, and multicast frames are flooded over the corresponding pseudowires and to all attachment circuits. Therefore, a PE must replicate packets across both attachment circuits and pseudowires.

MAC Address-based Forwarding

To forward a frame, a PE must associate a destination MAC address with a pseudowire or attachment circuit. This type of association is provided through a static configuration on each PE or through dynamic learning, which is flooded to all bridge ports.

Note Split horizon forwarding applies in this case, for example, frames that are coming in on an attachment circuit or pseudowire are sent out of the same pseudowire. The pseudowire frames, which are received on one psuedowire, are not replicated on other pseudowires in the same virtual forwarding instance (VFI).

MAC Address Source-based Learning

When a frame arrives on a bridge port (for example, pseudowire or attachment circuit) and the source MAC address is unknown to the receiving PE router, the source MAC address is associated with the pseudowire or attachment circuit. Outbound frames to the MAC address are forwarded to the appropriate pseudowire or attachment circuit.

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. The updated MAC tables are sent to all line cards (LCs) and program the hardware for the router.

The number of learned MAC addresses is limited through configurable per-port and per-bridge domain MAC address limits.

MAC Address Aging

A MAC address in the MAC table is considered valid only for the duration of the MAC address aging time. When the time expires, the relevant MAC entries are repopulated. When the MAC aging time is configured only under a bridge domain, all the pseudowires and attachment circuits in the bridge domain use that configured MAC aging time.

A bridge forwards, floods, or drops packets based on the bridge table. The bridge table maintains both static entries and dynamic entries. Static entries are entered by the network manager or by the bridge itself. Dynamic entries are entered by the bridge learning process. A dynamic entry is automatically removed after a specified length of time, known as *aging time*, from the time the entry was created or last updated.

If hosts on a bridged network are likely to move, decrease the aging-time to enable the bridge to adapt to the change quickly. If hosts do not transmit continuously, increase the aging time to record the dynamic entries for a longer time, thus reducing the possibility of flooding when the hosts transmit again.

MAC Address Limit

The MAC address limit is used to limit the number of learned MAC addresses. The limit is set at the bridge domain level and the port level. When the MAC address limit is violated, the system is configured to take one of the actions that are listed in Table 5.

Table 5 MAC Address Limit Actions

Action	Description
Limit flood	Discards the new MAC addresses.
Limit no-flood	Discards the new MAC addresses. Flooding of unknown unicast packets is disabled.
Shutdown	Disables the bridge domain or bridge port. When the bridge domain is down, none of the bridging functions, such as learning, flooding, forwarding, and so forth take place for the bridge domain. If a bridge port is down as a result of the action, the interface or pseudowire representing the bridge port remains up but the bridge port is not participating in the bridge. When disabled, the port or bridge domain is manually brought up by using an EXEC CLI.

When a limit is exceeded, the system is configured to perform the following notifications:

- Syslog (default)
- Simple Network Management Protocol (SNMP) trap
- Syslog and SNMP trap
- None (no notification)

To clear the MAC limit condition, the number of MACs must go below 75 percent of the configured limit.

MAC Address Withdrawal

For faster VPLS convergence, you can remove or unlearn the MAC addresses that are learned dynamically. The Label Distribution Protocol (LDP) Address Withdrawal message is sent with the list of MAC addresses, which need to be withdrawn to all other PEs that are participating in the corresponding VPLS service.

For the Cisco IOS XR VPLS implementation, a portion of the dynamically learned MAC addresses are cleared by using the MAC addresses aging mechanism by default. The MAC address withdrawal feature is added through the LDP Address Withdrawal message. To enable the MAC address withdrawal feature,

use the **withdrawal** command in l2vpn bridge group bridge domain MAC configuration mode. To verify that the MAC address withdrawal is enabled, use the **show l2vpn bridge-domain** command with the **detail** keyword.

Note By default, the LDP MAC Withdrawal feature is disabled on Cisco IOS XR.

The LDP MAC Withdrawal feature is generated due to the following events:

- Attachment circuit goes down. You can remove or add the attachment circuit through the CLI.
- MAC withdrawal messages are received over a VFI pseudowire and are not propagated over access pseudowires. RFC 4762 specifies that both wildcards (by means of an empty Type, Length and Value [TLV]) and a specific MAC address withdrawal. Cisco IOS XR supports only a wildcard MAC address withdrawal.

LSP Ping over VPWS and VPLS

For Cisco IOS XR, the existing support for the Label Switched Path (LSP) ping and traceroute verification mechanisms for point-to-point pseudowires (signaled using LDP FEC128) is extended to cover the pseudowires that are associated with the VFI (VPLS). Currently, the support for the LSP ping and traceroute is limited to manually configured VPLS pseudowires (signaled using LDP FEC128). For information about Virtual Circuit Connection Verification (VCCV) support and the **ping mpls pseudowire** command, see *Cisco IOS XR MPLS Command Reference*.

How to Implement Virtual Private LAN Services

This section describes the tasks that are required to implement VPLS:

- Configuring a Bridge Domain, page MPC-247
- Configuring a Layer 2 Virtual Forwarding Instance, page MPC-257
- Configuring the MAC Address-related Parameters, page MPC-269

Configuring a Bridge Domain

These topics describe how to configure a bridge domain:

- Creating a Bridge Domain, page MPC-247
- Configuring a Pseudowire, page MPC-249
- Associating Members with a Bridge Domain, page MPC-251
- Configuring Bridge Domain Parameters, page MPC-253
- Disabling a Bridge Domain, page MPC-255

Creating a Bridge Domain

Perform this task to create a bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

	Command or Action	Purpose

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Pseudowire

Perform this task to configure a pseudowire under a bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **exit**
7. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures the virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.
		• Use the *vfi name* argument to configure the name of the specified virtual forwarding interface.

Step 6	**exit**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# exit`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Exits the current configuration mode.

	Command or Action	Purpose

Step 7	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating Members with a Bridge Domain

After a bridge domain is created, perform this task to assign interfaces to the bridge domain. The following types of bridge ports are associated with a bridge domain:

- Ethernet and VLAN
- VFI

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **interface** *interface name*
6. **static-mac-address** {*MAC address*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**interface** interface name
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)#`
		Adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

	Command or Action	Purpose
Step 6	**static-mac-address** {*MAC address*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# static-mac-address 1.1.1`	Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Bridge Domain Parameters

To configure the bridge domain parameters, associate the following parameters with a bridge domain:

- Maximum transmission unit (MTU)—Specifies that all members of a bridge domain have the same MTU. The bridge domain member with a different MTU size is not used by the bridge domain even though it is still associated with a bridge domain.
- Flooding—Enables or disables flooding on the bridge domain. By default, flooding is enabled.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **flooding disable**

6. **mtu** *bytes*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**flooding disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)# flooding disable`	Configures flooding for traffic at the bridge domain level or at the bridge port level.

	Command or Action	Purpose
Step 6	**mtu** *bytes* **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mtu 1000`	Adjusts the maximum packet size or maximum transmission unit (MTU) size for the bridge domain. • Use the *bytes* argument to specify the MTU size, in bytes. The range is from 64 to 65535.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling a Bridge Domain

Perform this task to disable a bridge domain. When a bridge domain is disabled, all VFIs that are associated with the bridge domain are disabled. You are still able to attach or detach members to the bridge domain and the VFIs that are associated with the bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **shutdown**
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

	Command or Action	Purpose
Step 5	**shutdown** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`	Shuts down a bridge domain to bring the bridge and all attachment circuits and pseudowires under it to admin down state.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Layer 2 Virtual Forwarding Instance

These topics describe how to configure a Layer 2 virtual forwarding instance (VFI):

- Adding the Virtual Forwarding Instance Under the Bridge Domain, page MPC-257
- Associating Pseudowires with the Virtual Forwarding Instance, page MPC-259
- Associating a Virtual Forwarding Instance to a Bridge Domain, page MPC-261
- Attaching Pseudowire Classes to Pseudowires, page MPC-263
- Configuring Any Transport over Multiprotocol Pseudowires By Using Static Labels, page MPC-265
- Disabling a Virtual Forwarding Instance, page MPC-267

Adding the Virtual Forwarding Instance Under the Bridge Domain

Perform this task to create a Layer 2 Virtual Forwarding Instance (VFI) on all provider edge devices under the bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

	Command or Action	Purpose

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating Pseudowires with the Virtual Forwarding Instance

After a VFI is created, perform this task to associate one or more pseudowires with the VFI.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.

	Command or Action	Purpose

Step 6	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating a Virtual Forwarding Instance to a Bridge Domain

Perform this task to associate a VFI to be a member of a bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}

7. **static-mac-address** {*MAC address*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.

Step 6	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

	Command or Action	Purpose
Step 7	**static-mac-address** {*MAC address*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# static-mac-address 1.1.1`	Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Attaching Pseudowire Classes to Pseudowires

Perform this task to attach a pseudowire class to a pseudowire.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
7. **pw-class** {*class name*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.

Step 6	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

	Command or Action	Purpose
Step 7	**pw-class** {*class name*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada`	Configures the pseudowire class template name to use for the pseudowire.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Any Transport over Multiprotocol Pseudowires By Using Static Labels

Perform this task to configure the Any Transport over Multiprotocol (AToM) pseudowires by using the static labels. A pseudowire becomes a static AToM pseudowire by setting the MPLS static labels to local and remote.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}

7. **mpls static label** {**local** *value*} {**remote** *value*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.

Step 6	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

	Command or Action	Purpose
Step 7	**mpls static label** {**local** *value*} {**remote** *value*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# mpls static label local 800 remote 500`	Configures the MPLS static labels and the static labels for the access pseudowire configuration. You can set the local and remote pseudowire labels.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling a Virtual Forwarding Instance

Perform this task to disable a VFI. When a VFI is disabled, all the previously established pseudowires that are associated with the VFI are disconnected. LDP advertisements are sent to withdraw the MAC addresses that are associated with the VFI. However, you can still attach or detach attachment circuits with a VFI after a shutdown.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **shutdown**

7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters l2vpn bridge group bridge domain VFI configuration mode.

Step 6	**shutdown** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown`	Disables the virtual forwarding interface (VFI).

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the state of the VFI. For example, if you shut down the VFI, the VFI is shown as shut down under the bridge domain.

Configuring the MAC Address-related Parameters

These topics describe how to configure the MAC address-related parameters:

- Configuring the MAC Address Source-based Learning, page MPC-269
- Enabling the MAC Address Withdrawal, page MPC-271
- Configuring the MAC Address Limit, page MPC-274
- Configuring the MAC Address Aging, page MPC-276

The MAC table attributes are set for the bridge domains.

Configuring the MAC Address Source-based Learning

Perform this task to configure the MAC address source-based learning.

SUMMARY STEPS

1. **configure**
2. **l2vpn**

3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **learning disable**
7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6	**learning disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# learning disable`	Overrides the MAC learning configuration of a parent bridge or sets the MAC learning configuration of a bridge.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details that the MAC address source-based learning is disabled on the bridge.

Enabling the MAC Address Withdrawal

Perform this task to enable the MAC address withdrawal for a specified bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **withdrawal**
7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6	**withdrawal** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# withdrawal`	Enables the MAC address withdrawal for a specified bridge domain.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show l2vpn bridge-domain** [**detail**] **Example:** `P/0/RP0/CPU0:router# show l2vpn bridge-domain detail`	Displays detailed sample output to specify that the MAC address withdrawal is enabled. In addition, the sample output displays the number of MAC withdrawal messages that are sent over or received from the pseudowire.

The following sample output shows the MAC address withdrawal fields:

```
RP/0/0/CPU0:router# show l2vpn bridge-domain detail

Bridge group: siva_group, bridge-domain: siva_bd, id: 0, state: up, ShgId: 0, MSTi: 0
  MAC Learning: enabled
  MAC withdraw: enabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown Unicast: enabled
  MAC address aging time: 300 s Type: inactivity
  MAC address limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no
  Security: disabled
  DHCPv4 Snooping: disabled
  MTU: 1500
  MAC Filter:  Static MAC addresses:
  ACs: 1 (1 up), VFIs: 1, PWs: 2 (1 up)
  List of ACs:
    AC: GigabitEthernet0/4/0/1, state is up
      Type Ethernet
      MTU 1500; XC ID 0x5000001; interworking none; MSTi 0 (unprotected)
      MAC Learning: enabled
      MAC withdraw: disabled
```

```
    Flooding:
      Broadcast & Multicast: enabled
      Unknown Unicast: enabled
    MAC address aging time: 300 s Type: inactivity
    MAC address limit: 4000, Action: none, Notification: syslog
    MAC limit reached: no
    Security: disabled
    DHCPv4 Snooping: disabled
    Static MAC addresses:
    Statistics:
      packet totals: receive 6,send 0
      byte totals: receive 360,send 4
List of Access PWs:
List of VFIs:
  VFI siva_vfi
    PW: neighbor 1.1.1.1, PW ID 1, state is down ( local ready )
      PW class not set, XC ID 0xff000001
      Encapsulation MPLS, protocol LDP
      PW type Ethernet, control word enabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
              MPLS         Local                          Remote
        ------------ ------------------------------ -------------------------
             Label          30005                          unknown
             Group ID     0x0                              0x0
             Interface    siva/vfi                         unknown
             MTU          1500                             unknown
             Control word enabled                       unknown
             PW type      Ethernet                        unknown
        ------------ ------------------------------ -------------------------
      Create time: 19/11/2007 15:20:14 (00:25:25 ago)
      Last time status changed: 19/11/2007 15:44:00 (00:01:39 ago)
      MAC withdraw message: send 0 receive 0
```

Configuring the MAC Address Limit

Perform this task to configure the parameters for the MAC address limit.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **limit**
7. **maximum** {*value*}
8. **action** {**flood** | **no-flood** | **shutdown**}
9. **notification** {**both** | **none** | **trap**}
10. **end**
    or
    commit
11. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6	**limit**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# limit`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)#`
		Sets the MAC address limit for action, maximum, and notification and enters l2vpn bridge group bridge domain MAC limit configuration mode.

Step 7	**maximum** {*value*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# maximum 5000`	Configures the specified action when the number of MAC addresses learned on a bridge is reached.
Step 8	**action** {**flood** | **no-flood** | **shutdown**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# action flood`	Configures the bridge behavior when the number of learned MAC addresses exceed the MAC limit configured.

	Command or Action	Purpose
Step 9	**notification** {**both** | **none** | **trap**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# notification both`	Specifies the type of notification that is sent when the number of learned MAC addresses exceeds the configured limit.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details about the MAC address limit.

Configuring the MAC Address Aging

Perform this task to configure the parameters for MAC address aging.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **aging**
7. **time** {*seconds*}

8. **type** {**absolute** | **inactivity**}
9. **end**
   or
   commit
10. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6	**aging**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# aging`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#`
		Enters the MAC aging configuration submode to set the aging parameters such as time and type.

Step 7	**time** {*seconds*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# time 300`	Configures the maximum aging time. • Use the *seconds* argument to specify the maximum age of the MAC address table entry. The range is from 120 to 1000000 seconds. Aging time is counted from the last time that the switch saw the MAC address. The default value is 300 seconds.

	Command or Action	Purpose
Step 8	**type** {**absolute** | **inactivity**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# type absolute`	Configures the type for MAC address aging. • Use the **absolute** keyword to configure the absolute aging type. • Use the **inactivity** keyword to configure the inactivity aging type.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details about the aging fields.

Configuration Examples for Virtual Private LAN Services

This section includes the following configuration examples:

- Virtual Private LAN Services Configuration for Provider Edge-to-Provider Edge: Example, page MPC-279
- Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example, page MPC-280

Virtual Private LAN Services Configuration for Provider Edge-to-Provider Edge: Example

These configuration examples show how to create a Layer 2 VFI with a full-mesh of participating VPLS provider edge (PE) nodes.

The following configuration example shows how to configure PE 1:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE1-VPLS-A
    GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 2.2.2.2 pw-id 1---PW1
     neighbor 3.3.3.3 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 1.1.1.1 255.255.255.25
  commit
```

The following configuration example shows how to configure PE 2:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE2-VPLS-A
    interface GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 1.1.1.1 pw-id 1---PW1
     neighbor 3.3.3.3 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 2.2.2.2 255.255.255.25
  commit
```

The following configuration example shows how to configure PE 3:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE3-VPLS-A
    interface GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 1.1.1.1 pw-id 1---PW1
     neighbor 2.2.2.2 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 3.3.3.3 255.255.255.25
  commit
```

Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example

The following configuration shows how to configure VPLS for a PE-to-CE nodes:

```
configure
 interface GigabitEthernet0/0
  l2transport---AC interface
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end

configure
 interface GigabitEthernet0/0
  l2transport
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end

configure
 interface GigabitEthernet0/0
  l2transport
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
```

Additional References

For additional information related to implementing VPLS, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS VPLS-related commands	*MPLS Virtual Private LAN Services Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*

Related Topic	Document Title
Cisco CRS-1 router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 4447	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*, April 2006
RFC 4448	*Encapsulation Methods for Transport of Ethernet over MPLS Networks*, April 2006

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing IPv6 VPN Provider Edge Transport over MPLS on Cisco IOS XR Software

IPv6 VPN Provider Edge (6PE) uses the existing MPLS IPv4 core infrastructure for IPv6 transport. 6PE enables IPv6 sites to communicate with each other over an MPLS IPv4 core network using MPLS label switched paths (LSPs).

This feature relies heavily on multiprotocol Border Gateway Protocol (BGP) extensions in the IPv4 network configuration on the provider edge (PE) router to exchange IPv6 reachability information (in addition to an MPLS label) for each IPv6 address prefix. Edge routers are configured as dual-stack, running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange.

For detailed information about the commands used to configure L2TP functionality, see *Cisco IOS XR Routing Command Reference*.

Feature History for Implementing 6PE on Cisco IOS XR Software

Release	Modification
Release 3.5.0	This feature was introduced on the Cisco XR 12000 Series Router.
Release 3.6.0	No modification.
Release 3.7.0	Support was added for: • Cisco CRS-1 • Inter-AS 6PE

Contents

- Prerequisites for Implementing 6PE, page MPC-284
- Information About 6PE, page MPC-284
- How to Implement 6PE, page MPC-286
- Configuration Examples for 6PE, page MPC-288
- Additional References, page MPC-289

Prerequisites for Implementing 6PE

The following prerequisites are required to implement 6PE:

- You must be familiar with MPLS and BGP4 configuration and troubleshooting.
- You must be in a user group associated with a task group that includes the proper task IDs for MPLS 6PE commands. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*.

Information About 6PE

To configure the 6PE feature, you should understand the following concepts, which are described in the following sections:

- Overview of 6PE, page MPC-284
- Benefits of 6PE, page MPC-284
- Deploying IPv6 over MPLS Backbones, page MPC-285
- IPv6 on the Provider Edge and Customer Edge Routers, page MPC-285
- IPv6 Provider Edge Multipath, page MPC-286

Overview of 6PE

Multiple techniques are available to integrate IPv6 services over service provider core backbones:

- Dedicated IPv6 network running over various data link layers
- Dual-stack IPv4-IPv6 backbone
- Leveraging of an existing MPLS backbone

These solutions are deployed on service providers' backbones when the amount of IPv6 traffic and the revenue generated are in line with the necessary investments and the risks agreed to. Conditions are favorable for the introduction of native IPv6 service, from the edge, in a scalable way, without any IPv6 addressing restrictions and without putting a well-controlled IPv4 backbone in jeopardy. Backbone stability is key for service providers that recently stabilized their IPv4 infrastructure.

Service providers running an MPLS/IPv4 infrastructure follow the same trends, as several integration scenarios are possible to offer IPv6 services on an MPLS network. Cisco Systems specially developed Cisco 6PE, or, IPv6 Provider Edge Router over MPLS, to meet all of those requirements.

Inter-AS support for 6PE requires support of Border Gateway Protocol (BGP) to enable the address families and to allocate and distribute the PE and ASBR labels.

Benefits of 6PE

Service providers that currently deploy MPLS will experience the following benefits of Cisco 6PE:

- Minimal operational cost and risk—No impact on existing IPv4 and MPLS services.
- Provider edge routers upgrade only—A 6PE router can be an existing PE router or a new one dedicated to IPv6 traffic.

- No impact on IPv6 customer edge routers—The ISP can connect to any customer CE running Static, IGP or EGP.
- Ready for production services—An ISP can delegate IPv6 prefixes.
- IPv6 introduction into an existing MPLS service—6PE routers can be added at any time.
- It is possible to switch up to OC-192 speed in the core.

Deploying IPv6 over MPLS Backbones

Backbones enabled by 6PE (IPv6 over MPLS) allow IPv6 domains to communicate with each other over an MPLS IPv4 core network. This implementation requires no backbone infrastructure upgrades and no reconfiguration of core routers, because forwarding is based on labels rather than on the IP header itself. This provides a very cost-effective strategy for IPv6 deployment.

Additionally, the inherent virtual private network (VPN) and traffic engineering (TE) services available within an MPLS environment allow IPv6 networks to be combined into VPNs or extranets over an infrastructure that supports IPv4 VPNs and MPLS-TE.

IPv6 on the Provider Edge and Customer Edge Routers

Service Provider Edge Routers

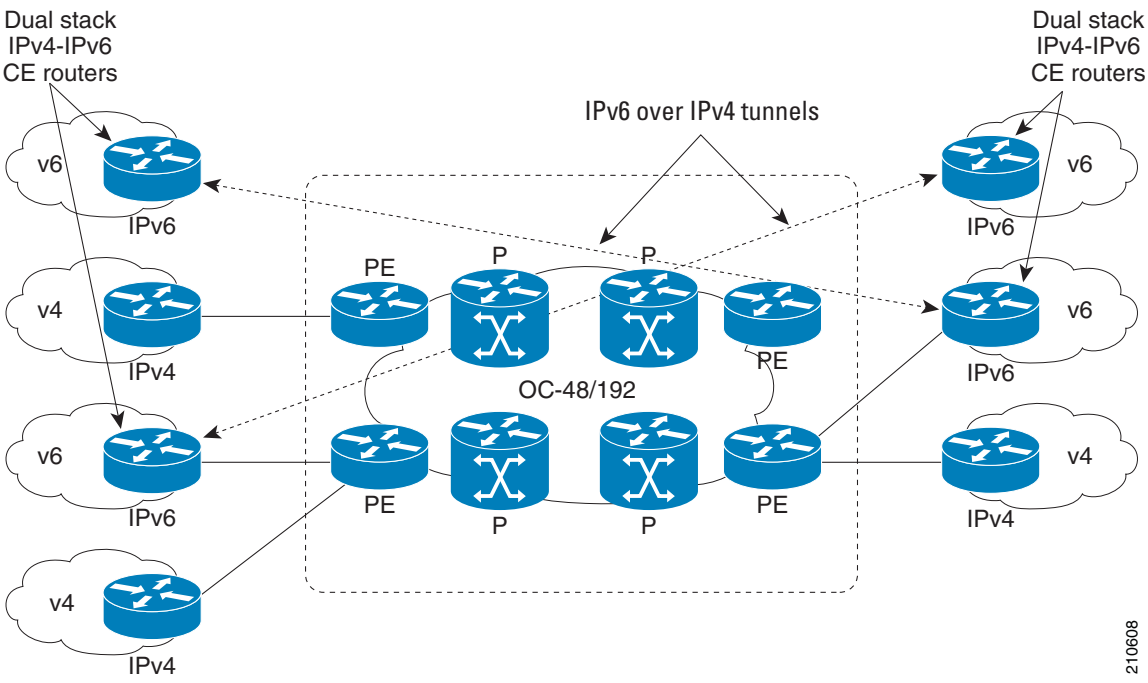
6PE is particularly applicable to service providers who currently run an MPLS network. One of its advantages is that there is no need to upgrade the hardware, software, or configuration of the core network, and it eliminates the impact on the operations and the revenues generated by the existing IPv4 traffic. MPLS is used by many service providers to deliver services to customers. MPLS as a multiservice infrastructure technology is able to provide layer 3 VPN, QoS, traffic engineering, fast re-routing and integration of ATM and IP switching.

Customer Edge Routers

Using tunnels on the CE routers is the simplest way to deploy IPv6 over MPLS networks. It has no impact on the operation or infrastructure of MPLS and requires no changes to the P routers in the core or to the PE routers. However, tunnel meshing is required as the number of CEs to connect increases, and it is difficult to delegate a global IPv6 prefix for an ISP.

Figure 22 illustrates the network architecture using tunnels on the CE routers.

Figure 22 IPv6 Using Tunnels on the CE Routers



IPv6 Provider Edge Multipath

Internal and external BGP multipath for IPv6 allows the IPv6 router to load balance between several paths (for example, same neighboring autonomous system (AS) or sub-AS, or the same metric) to reach its destination. The 6PE multipath feature uses multiprotocol internal BGP (MP-IBGP) to distribute IPv6 routes over the MPLS IPv4 core network and to attach an MPLS label to each route.

When MP-IBGP multipath is enabled on the 6PE router, all labeled paths are installed in the forwarding table with MPLS information (label stack) when MPLS information is available. This functionality enables 6PE to perform load balancing.

How to Implement 6PE

This section includes the following implementation procedure:

- Configuring 6PE, page MPC-286

Configuring 6PE

This task describes how to configure 6PE on PE routers to transport the IPv6 prefixes across the IPv4 cloud.

Be sure to configure 6PE on PE routers participating in both the IPv4 cloud and IPv6 clouds.

Note To learn routes from both clouds, you can use all routing protocols supported on Cisco IOS XR software: BGP, OSPF, IS-IS, EIGRP, RIP, and Static.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family ipv6 labeled-unicast**
5. **exit**
6. **exit**
7. **address-family ipv6 unicast**
8. **allocate-label** [**all** | **route-policy** *policy_name*]
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 1`	Enters the number that identifies the autonomous system (AS) in which the router resides. Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 1.1.1.1`	Enters neighbor configuration mode for configuring Border Gateway Protocol (BGP) routing sessions.
Step 4	**address-family ipv6 labeled-unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 labeled-unicast`	Specifies IPv6 labeled-unicast address prefixes. **Note** This option is also available in IPv6 neighbor configuration mode and VRF neighbor configuration mode.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit`	Exits BGP address-family submode.

	Command or Action	Purpose
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# exit`	Exits BGP neighbor submode.
Step 7	**address-family ipv6 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# address-family ipv6 unicast`	Specifies IPv6 unicast address prefixes.
Step 8	**allocate-label** [**all** | **route-policy** *policy_name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all`	Allocates MPLS labels for specified IPv4 unicast routes. **Note** The **route-policy** keyword provides finer control to filter out certain routes from being advertised to the neighbor.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for 6PE

This section includes the following configuration example:

- Configuring 6PE on a PE Router: Example, page MPC-288

Configuring 6PE on a PE Router: Example

The following sample configuration shows the configuration of 6PE on a PE router:

```
interface POS0/3/0/0
 ipv6 address 2001::1/64
!
```

```
router isis ipv6-cloud
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface POS0/3/0/0
  address-family ipv6 unicast
  !
!
router bgp 55400
 bgp router-id 54.6.1.1
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
  network 55:5::/64
  redistribute connected
  redistribute isis ipv6-cloud
 !
 neighbor 34.4.3.3
  remote-as 55400
  address-family ipv4 unicast
  !
  address-family ipv6 labeled-unicast
```

Additional References

For additional information related to this feature, refer to the following references:

Related Document

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco CRS-1 router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
—	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software

Layer 2 Tunnel Protocol Version 3 (L2TPv3) is an Internet Engineering Task Force (IETF) working group draft that provides several enhancements to L2TP, including the ability to tunnel any Layer 2 (L2) payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network using L2 virtual private networks (VPNs).

For additional information about L2TPv3, see *Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software.*

Feature History for Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software

Release	Modification
Release 3.7.0	This feature was introduced on the Cisco XR 12000 Series Router.

Contents

- Prerequisites for Layer 2 Tunnel Protocol Version 3, page MPC-291
- Information About Layer 2 Tunnel Protocol Version 3, page MPC-292
- How to Implement Layer 2 Tunnel Protocol Version 3, page MPC-296
- Configuration Examples for Layer 2 Tunnel Protocol Version 3, page MPC-318
- Additional References, page MPC-320

Prerequisites for Layer 2 Tunnel Protocol Version 3

The following prerequisites are required to implement L2TPv3:

- You must enable Cisco Express Forwarding (CEF) before you configure an cross-connect attachment circuit (AC) for a customer edge (CE) device.
- You must configure a Loopback interface on the router for originating and terminating the L2TPv3 traffic. The Loopback interface must have an IP address that is reachable from the remote provider edge (PE) device at the other end of an L2TPv3 control-channel.

Note A cross-connection is expressed as *xconnect* in the CLI.

Information About Layer 2 Tunnel Protocol Version 3

To configure the L2TPv3 feature, you should understand the following concepts:

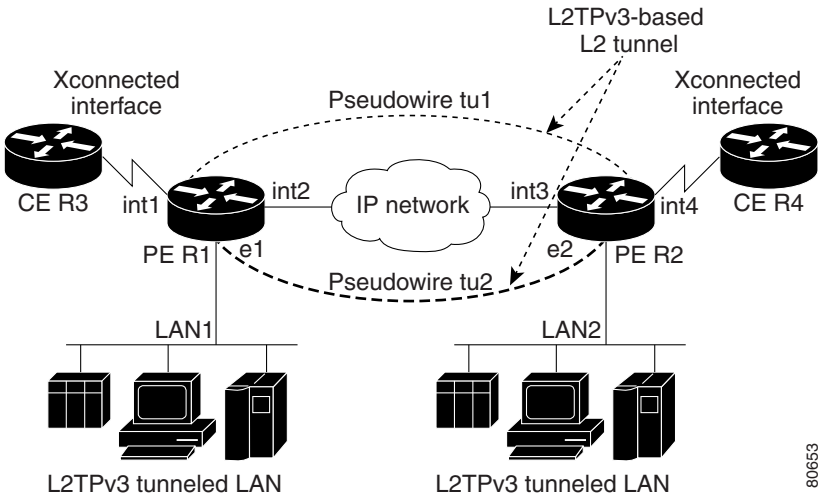
- L2TPv3 Operation, page MPC-292
- L2TPv3 Benefits, page MPC-293
- L2TPv3 Features, page MPC-293

L2TPv3 Operation

Figure 23 shows how the L2TPv3 feature is used to set up VPNs using Layer 2 tunneling over an IP network. All traffic between two customer network sites is encapsulated in IP packets carrying L2TP data messages and sent across an IP network. The backbone routers of the IP network treat the traffic as any other IP traffic and does not need to know anything about the customer networks.

Both PE routers R1 and R2 provide L2TPv3 services. The R1 and R2 routers communicate with each other using a pseudowire over the IP backbone network through a path comprising the interfaces *int1* and *int2*, the IP network, and interfaces *int3* and *int4*. The CE routers R3 and R4 communicate through a pair of cross-connected Ethernet or 802.1q VLAN interfaces using an L2TPv3 session. The L2TPv3 session tu1 is a pseudowire configured between interface int1 on R1 and interface int4 on R2. Any packet arriving on interface int1 on R1 is encapsulated and sent through the pseudowire (tu1) to R2. R2 decapsulates the packet and sends it on interface int4 to R4. When R4 needs to send a packet to R3, the packet follows the same path in reverse.

Figure 23 L2TPv3 Operation



L2TPv3 Benefits

L2TPv3 provides the following benefits:

- Simplifies deployment of VPNs—L2TPv3 is an industry-standard L2 tunneling protocol that ensures interoperability among vendors, increasing customer flexibility and service availability.
- Does not require MPLS—Service providers need not deploy MPLS in the core IP backbone to set up VPNs using L2TPv3 over the IP backbone; this will result in operational savings and increased revenue.
- Supports L2 tunneling over IP for any payload—L2TPv3 provides enhancements to L2TP to support L2 tunneling of any payload over an IP core network. L2TPv3 defines the base L2TP protocol as being separate from the L2 payload that is tunneled.

L2TPv3 Features

L2TPv3 provides cross-connect support for Ethernet, 802.1q (VLAN), and Frame Relay, using the sessions described in the following sections:

- Static L2TPv3 Sessions, page MPC-293
- Dynamic L2TPv3 Sessions, page MPC-294

L2TPv3 also supports:

- Sequencing, page MPC-294
- Local Switching, page MPC-294
- Local Switching: Quality of Service, page MPC-295
- L2TPv3 Pseudowire Manager, page MPC-295
- L2TPv3 Type of Service Marking, page MPC-296
- Keepalive, page MPC-296
- Maximum Transmission Unit Handling, page MPC-296
- Distributed switching
- L2TPv3 control message hashing
- L2TPv3 control message rate limiting
- L2TPv3 digest secret graceful switchover
- Manual clearing of L2TPv3 tunnels
- L2TPv3 tunnel management

Static L2TPv3 Sessions

Typically, the L2TP control plane is responsible for negotiating session parameters (such as the session ID or the cookie) to set up the session; however, some IP networks require sessions to be configured so that no signaling is required for session establishment. Therefore, you can set up static L2TPv3 sessions for a PE router by configuring fixed values for the fields in the L2TP data header. A static L2TPv3 session allows the PE to tunnel L2 traffic as soon as the AC to which the session is bound comes up.

Note In an L2TPv3 static session, you can still run the L2TP control-channel to perform peer authentication and dead-peer detection. If the L2TP control-channel cannot be established or is torn down because of a hello failure, the static session is also torn down.

When you use a static L2TPv3 session, you cannot perform circuit interworking (for example, LMI) because there is no facility to exchange control messages. To perform circuit interworking, you must use a dynamic session.

Dynamic L2TPv3 Sessions

A dynamic L2TP session is established through the exchange of control messages containing attribute-value pair (AVP). Each AVP contains information about the nature of the L2 link being forwarded: including the payload type, virtual circuit (VC) ID, and so on.

Multiple L2TP sessions can exist between a pair of PEs, and can be maintained by a single control-channel. Session IDs and cookies are dynamically generated and exchanged as part of a dynamic session setup. Sequencing configuration is also exchanged and circuit state changes are conveyed using the set link info (SLI) message.

Sequencing

Although the correct sequence of received L2 frames is guaranteed by some L2 technologies (by the nature of the link, such as a serial line) or the protocol itself, forwarded L2 frames may be lost, duplicated, or reordered when they traverse a network as IP packets. If the L2 protocol does not provide an explicit sequencing mechanism, you can configure L2TP to sequence its data packets according to the data channel sequencing mechanism described in the L2TPv3 IETF l2tpext working group draft.

A receiver of L2TP data packets mandates sequencing through the sequencing required AVP when the session is being negotiated. A sender that receives this AVP (or that is manually configured to send sequenced packets) uses the L2-specific pseudowire control encapsulation defined in L2TPv3.

Currently, you can configure L2TP only to drop out-of-order packets; you cannot configure L2TP to deliver the packets out-of-order. No reordering mechanism is available.

Local Switching

An AC to AC cross-connect, also called *local switching*, is a building block of L2VPN that allows frames to switch between two different ACs on the same PE (see Figure 24). Local switching is supported for both static and dynamic sessions.

You must configure separate IP addresses for each cross-connect statement.

Note The Cisco CRS-1 router plays the role of a PE router at the edge of a provider network, where CE devices are connected to Cisco CRS-1 PE routers using Layer 2 LAN services.

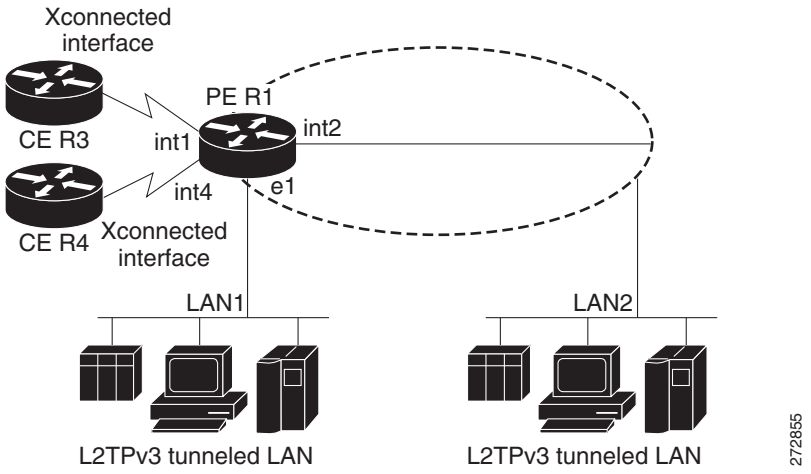
The following configurations are supported for local switching:

- Port-to-Port
- VLAN-to-VLAN

- Port-to-VLAN
- VLAN-to-Port

Note VLAN-to-VLAN options do not require interworking. Port-to-VLAN and VLAN-to-port do, and it is locally managed by the L2VPN application. If both interfaces are Ethernet VLAN, each reside on a single physical interface. By definition, local switching is not a pseudowire technology, because signaling protocols (such as LDP or L2TPv3) are not involved.

Figure 24 Local Switching Operation



Local Switching: Quality of Service

The following quality of service (QoS) requirements apply to local switching:

- QoS service policies can be applied to any L2 AC (port or VLAN, or both) and can be applied to any interworking mode (port-to-port, vlan-to-port, port-to-vlan, vlan-to-vlan). The AC can be cross-connected to a pseudowire (EoL2TPv3) or to another AC (local switching).
- QoS service policies can be attached directly to the AC.
- QoS service policies can be attached to the main interface using **match vlan** on L2 VLAN ACs.
- QoS service policies attached to the main interface can be inherited by all L2 VLANs.
- QoS service policies cannot be attached to a main interface when there are service policies already attached to its L3VLANs or L2VLAN ACs.
- QoS service policies already attached to the main interface are not permitted on L3 VLAN or L2 VLAN ACs.

L2TPv3 Pseudowire Manager

The pseudowire manager is a client library provided by the pseudowire signaling module that runs in the context of the L2VPN process. This client library implements interface to pseudo-wire signaling protocol for specific pseudowire type.

L2TPv3 Type of Service Marking

When L2 traffic is tunneled across an IP network, information contained in the type of service (ToS) bits may be transferred to the L2TP-encapsulated IP packets in one of the following ways:

- If the tunneled L2 frames encapsulate IP packets themselves, it may be desirable to simply copy the ToS bytes of the inner IP packets to the outer IP packet headers. This action is known as "ToS byte reflection."
- Static ToS byte configuration. You specify the ToS byte value used by all packets sent across the pseudowire.

Keepalive

The keepalive mechanism for L2TPv3 extends only to the endpoints of the tunneling protocol. L2TP has a reliable control message delivery mechanism that serves as the basis for the keepalive mechanism. The keepalive mechanism consists of an exchange of L2TP hello messages.

If a keepalive mechanism is required, the control plane is used, although it may not be used to bring up sessions. You can manually configure sessions.

In the case of static L2TPv3 sessions, a control channel between the two L2TP peers is negotiated through the exchange of start control channel request (SCCRQ), start control channel replay (SCCRP), and start control channel connected (SCCCN) control messages. The control channel is responsible only for maintaining the keepalive mechanism through the exchange of hello messages.

The interval between hello messages is configurable per control channel. If one peer detects that the other has gone down through the keepalive mechanism, it sends a StopCCN control message and then notifies all of the pseudowires to the peer about the event. This notification results in the teardown of both manually configured and dynamic sessions.

Maximum Transmission Unit Handling

It is important that you configure an maximum transmission unit (MTU) appropriate for each L2TPv3 tunneled link. The configured MTU size ensures the following:

- The lengths of the tunneled L2 frames fall below the MTU of the destination AC.
- The tunneled packets are not fragmented, which forces the receiving PE to reassemble them.

L2TPv3 handles the MTU as follows:

- The default behavior is to fragment packets that are larger than the session MTU.

How to Implement Layer 2 Tunnel Protocol Version 3

This section includes the tasks required to implement L2TPv3, as follows:

- Configuring a Pseudowire Class, page MPC-297 (required)
- Configuring L2TP Control-Channel Parameters, page MPC-298 (required)
- Configuring L2TPv3 Pseudowires, page MPC-309 (required)
- Configuring the Cross-connect Attachment Circuit, page MPC-315 (required)

Configuring a Pseudowire Class

Perform this task to configure a pseudowire class or template.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class name*
4. **encapsulation** {**mpls** | **l2tpv3**}
5. **sequencing** {**both**}
6. **protocol l2tpv3** [**class** *class name*]
7. **ipv4 source** *ip-address*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configure submode.
Step 3	**pw-class** *class name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class wkg`	Enters a pseudowire-class name.
Step 4	**encapsulation** {**l2tpv3** | **mpls**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`	Configures pseudowire encapsulation. The options are: • L2TPv3—Sets pseudowire encapsulation to L2TPV3. • MPLS—Sets pseudowire encapsulation to MPLS.
Step 5	**sequencing** {**both**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# sequencing both`	Configures pseudowire class sequencing.

	Command or Action	Purpose
Step 6	**protocol l2tpv3** [**class** *class name*] **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3) # protocol l2tpv3`	Configures the dynamic pseudowire signaling protocol. If the class is not specified, the default class is used.
Step 7	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3) # ipv4 source 127.0.0.1`	Configures the local source IPv4 address.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3) # end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3) # commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring L2TP Control-Channel Parameters

This section describes the tasks to create a template of L2TP control-channel parameters that can be inherited by different pseudowire classes. The three main parameters described are:

- Timing parameters
- Authentication parameters
- Maintenance parameters

L2TP control-channel parameters are used in control-channel authentication, keepalive messages, and control-channel negotiation. In an L2TPv3 session, the same L2TP class must be specified in the pseudowire configured on the PE router at each end of the control-channel.

Note

- The L2TP class must be configured before it is associated with a pseudowire class (see Configuring a Pseudowire Class, page MPC-297).
- These tasks are supported only on the Cisco XR 12000 Series Router.

The three main groups of L2TP control-channel parameters that you can configure in an L2TP class are described in the following subsections:

- Configuring L2TP Control-Channel Timing Parameters, page MPC-299
- Configuring L2TPv3 Control-Channel Authentication Parameters, page MPC-300
- Configuring L2TP Control-Channel Maintenance Parameters, page MPC-308

Note When you enter L2TP class configuration mode, you can configure L2TP control-channel parameters in any order. If you have multiple authentication requirements you can configure multiple sets of L2TP class control-channel parameters with different L2TP class names. However, only one set of L2TP class control-channel parameters can be applied to a connection between any pair of IP addresses.

Configuring L2TP Control-Channel Timing Parameters

The following L2TP control-channel timing parameters can be configured in L2TP class configuration mode:

- Packet size of the receive window used for the control-channel.
- Retransmission parameters used for control messages.
- Timeout parameters used for the control-channel.

Note This task configures a set of timing control-channel parameters in an L2TP class. All timing control-channel parameter configurations can be configured in any order. If not configured, the default values are applied.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *l2tp-class-name*
3. **receive-window** *size*
4. **retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*}
5. **timeout setup** *seconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *l2tp-class-name* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class cisco`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 3	**receive-window** *size* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# receive-window 30`	Configures the number of packets that can be received by the remote peer before backoff queueing occurs. • The valid values range from 1 to the upper limit the peer has for receiving packets. The default value is 512.
Step 4	**retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*} **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# retransmit retries 10`	Configures parameters that affect the retransmission of control packets. • **initial retries**—Specifies how many SCCRQs are re-sent before giving up on the session. Range is 1 to 1000. The default is 2. • **retries**—Specifies how many retransmission cycles occur before determining that the peer PE router does not respond. Range is 1 to 1000. The default is 15. • **timeout** {**max** | **min**}—Specifies maximum and minimum retransmission intervals (in seconds) for resending control packets. Range is 1 to 8. The default maximum interval is 8; the default minimum interval is 1.
Step 5	**timeout setup** *seconds* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# timeout setup 400`	Configures the amount of time, in seconds, allowed to set up a control-channel. • Range is 60 to 6000. Default value is 300.

Configuring L2TPv3 Control-Channel Authentication Parameters

Two methods of control-channel message authentication are available:

- L2TP Control-Channel (see Configuring Authentication for the L2TP Control-Channel, page MPC-301)
- L2TPv3 Control Message Hashing (see Configuring L2TPv3 Control Message Hashing, page MPC-302)

You can enable both methods of authentication to ensure interoperability with peers that support only one of these methods of authentication, but this configuration will yield control of which authentication method is used to the peer PE router. Enabling both methods of authentication should be considered an interim solution to solve backward-compatibility issues during software upgrades.

The principal difference between the L2TPv3 Control Message Hashing feature and CHAP-style L2TP control-channel authentication is that, instead of computing the hash over selected contents of a received control message, the L2TPv3 Control Message Hashing feature uses the entire message in the hash. In addition, instead of including the hash digest in only the SCCRP and SCCCN messages, it includes it in all messages.

This section also describes how to configure L2TPv3 digest secret graceful switchover (see Configuring L2TPv3 Digest Secret Graceful Switchover, page MPC-304,) which lets you make the transition from an old L2TPv3 control-channel authentication password to a new L2TPv3 control-channel authentication password without disrupting established L2TPv3 tunnels.

Note Support for L2TP control-channel authentication is maintained for backward compatibility. Either or both authentication methods can be enabled to allow interoperability with peers supporting only one of the authentication methods.

Configuring Authentication for the L2TP Control-Channel

The L2TP control-channel method of authentication is the older, CHAP-like authentication system inherited from L2TPv2.

The following L2TP control-channel authentication parameters can be configured in L2TP class configuration mode:

- Authentication for the L2TP control-channel
- Password used for L2TP control-channel authentication
- Local hostname used for authenticating the control-channel

This task configures a set of authentication control-channel parameters in an L2TP class. All of the authentication control-channel parameter configurations may be configured in any order. If these parameters are not configured, the default values are applied.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **authentication**
4. **password** {**0** | **7**} *password*
5. **hostname** *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 3	**authentication** **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# authentication`	Enables authentication for the control-channel between PE routers.
Step 4	**password** {**0** | **7**} *password* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# password 7 cisco`	Configures the password used for control-channel authentication. • [**0** | **7**]—Specifies the input format of the shared secret. The default value is **0**. – **0**—Specifies an encrypted password will follow. – **7**—Specifies an unencrypted password will follow. • *password*—Defines the shared password between peer routers.
Step 5	**hostname** *name* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# hostname yb2`	Specifies a hostname used to identify the router during L2TP control-channel authentication. • If you do not use this command, the default hostname of the router is used.

Configuring L2TPv3 Control Message Hashing

Perform this task to configure L2TPv3 Control Message Hashing feature for an L2TP class.

L2TPv3 control message hashing incorporates authentication or integrity check for all control messages. This per-message authentication is designed to guard against control message spoofing and replay attacks that would otherwise be trivial to mount against the network.

Enabling the L2TPv3Control Message Hashing feature will impact performance during control-channel and session establishment because additional digest calculation of the full message content is required for each sent and received control message. This is an expected trade-off for the additional security afforded by this feature. In addition, network congestion may occur if the receive window size is too small. If the L2TPv3 Control Message Hashing feature is enabled, message digest validation must be enabled. Message digest validation deactivates the data path received sequence number update and restricts the minimum local receive window size to 35.

You can configure control-channel authentication or control message integrity checking; however, control-channel authentication requires participation by both peers, and a shared secret must be configured on both routers. Control message integrity check is unidirectional, and requires configuration on only one of the peers.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*]
4. **hidden**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.

	Command or Action	Purpose
Step 3	**digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*] **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# digest secret cisco hash sha`	Enables L2TPv3 control-channel authentication or integrity checking. • **secret**—Enables L2TPv3 control-channel authentication. **Note** If the **digest** command is issued without the **secret** keyword option, L2TPv3 integrity checking is enabled. • {**0** | **7**}—Specifies the input format of the shared secret. The default value is **0**. – **0**—Specifies that a plain-text secret is entered. – **7**—Specifies that an encrypted secret is entered. • *password*—Defines the shared secret between peer routers. The value entered for the *password* argument must be in the format that matches the input format specified by the {**0** | **7**} keyword option. • **hash** {**MD5** | **SHA1**}—Specifies the hash function to be used in per-message digest calculations. – **MD5**—Specifies HMAC-MD5 hashing (default value). – **SHA1**—Specifies HMAC-SHA-1 hashing.
Step 4	**hidden** **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# hidden`	Enables AVP hiding when sending control messages to an L2TPv3 peer.

Configuring L2TPv3 Digest Secret Graceful Switchover

Perform this task to make the transition from an old L2TPv3 control-channel authentication password to a new L2TPv3 control-channel authentication password without disrupting established L2TPv3 tunnels.

Note This task is not compatible with authentication passwords configured with the older, CHAP-like control-channel authentication system.

L2TPv3 control-channel authentication occurs using a password that is configured on all participating peer PE routers. The L2TPv3 Digest Secret Graceful Switchover feature allows a transition from an old control-channel authentication password to a new control-channel authentication password without disrupting established L2TPv3 tunnels.

Before performing this task, you must enable control-channel authentication (see Configuring L2TPv3 Control Message Hashing, page MPC-302).

Note During the period when both a new and an old password are configured, authentication can occur only with the new password if the attempt to authenticate using the old password fails.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*]
4. **end**
   or
   commit
5. **show l2tp tunnel** all
6. **configure**
7. **l2tp-class** *word*
8. **no digest** [**secret** [**0** | **7**] *password*] [**hash** {**md5** | **sha**}]
9. **end**
   or
   commit
10. **show l2tp tunnel** all

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.

	Command or Action	Purpose
Step 3	**digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*] **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# digest secret cisco hash sha`	Enables L2TPv3 control-channel authentication or integrity checking. • **secret**—Enables L2TPv3 control-channel authentication. **Note** If the **digest** command is issued without the **secret** keyword option, L2TPv3 integrity checking is enabled. • {**0** | **7**}—Specifies the input format of the shared secret. The default value is **0**. – **0**—Specifies that a plain-text secret is entered. – **7**—Specifies that an encrypted secret is entered. • *password*—Defines the shared secret between peer routers. The value entered for the *password* argument must be in the format that matches the input format specified by the {**0** | **7**} keyword option. • **hash** {**MD5** | **SHA1**}—Specifies the hash function to be used in per-message digest calculations. – **MD5**—Specifies HMAC-MD5 hashing (default value). – **SHA1**—Specifies HMAC-SHA-1 hashing.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2tp-class)# end`
	or
	`RP/0/0/CPU0:router(config-l2tp-class)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

	Command or Action	Purpose
Step 5	**show l2tp tunnel all** **Example:** `RP/0/0/CPU0:router# show l2tun tunnel all`	Displays the current state of L2 tunnels and information about configured tunnels, including local and remote L2 Tunneling Protocol (L2TP) hostnames, aggregate packet counts, and control-channel information. **Note** Use this command to determine if any tunnels are not using the new password for control-channel authentication. The output displayed for each tunnel in the specified L2TP class should show that two secrets are configured.
Step 6	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 7	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 8	**no digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*] **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# no digest secret cisco hash sha1`	Disables L2TPv3 control-channel authentication or integrity checking.

	Command or Action	Purpose

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2tp-class)# end`
	or
	`RP/0/0/CPU0:router(config-l2tp-class)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show l2tp tunnel all** **Example:** `RP/0/0/CPU0:router# show l2tun tunnel all`	Displays the current state of L2 tunnels and information about configured tunnels, including local and remote L2 Tunneling Protocol (L2TP) hostnames, aggregate packet counts, and control-channel information. • Tunnels should no longer be using the old control-channel authentication password. If a tunnel does not update to show that only one secret is configured after several minutes have passed, that tunnel can be manually cleared and a defect report should be filed with TAC. **Note** Issue this command to ensure that all tunnels are using only the new password for control-channel authentication. The output displayed for each tunnel in the specified L2TP class should show that one secret is configured.

Configuring L2TP Control-Channel Maintenance Parameters

Perform this task to configure the interval used for hello messages in an L2TP class.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **hello-interval** *interval*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 3	**hello-interval** *interval* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# hello-interval 100`	Specifies the exchange interval (in seconds) used between L2TP hello packets. • Valid values for the *interval* argument range from 0 to 1000. The default value is 60.

Configuring L2TPv3 Pseudowires

Perform the following tasks to configure static and dynamic L2TPv3 pseudowires:

- Configuring a Dynamic L2TPv3 Pseudowire, page MPC-309
- Configuring a Static L2TPv3 Pseudowire, page MPC-312

Configuring a Dynamic L2TPv3 Pseudowire

Perform this task to configure a dynamic L2TPv3 pseudowire.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *name*
4. **p2p** *name*
5. **neighbor** *ip-address* **pw-id** *number*
6. **pw-class** *pw-class-name*
7. **end**
   or
   commit

8. **pw-class** *pw-class-name*
9. **encapsulation l2tpv3**
10. **ipv4 source** *ip-address*
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enter L2VPN configure submode.
Step 3	**xconnect group** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp_01`	Enter a name for the cross-connect group.
Step 4	**p2p** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1`	Enters p2p configuration submode to configure point-to-point cross-connects.
Step 5	**neighbor** *ip-address* **pw-id** *number* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 665`	Configures a pseudowire for a cross-connect.
Step 6	**pw-class** *pw-class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class class100`	Enters pseudowire class submode to define a name for the cross-connect.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**pw-class** *pw-class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class class100`	Enters pseudowire class submode to define a pseudowire class template.
Step 9	**encapsulation l2tpv3** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`	Configures L2TPv3 pseudowire encapsulation.

	Command or Action	Purpose
Step 10	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# ipv4 source 127.0.0.1`	Configures the local source IPv4 address.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# commit`
		Saves configuration changes.
		• When you enter the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Configuring a Static L2TPv3 Pseudowire

Perform this task to configure a static L2TPv3 pseudowire.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *name*
4. **p2p** *name*
5. **neighbor** *ip-address* **pw-id** *number*
6. **l2tp static local session** {*session-id*}
7. **l2tp static local cookie size** {**0** | **4** | **8**} [**value** {*low-value*} [{*high-value*}]]
8. **l2tp static remote session** {*session-id*}
9. **l2tp static remote cookie size** {**0** | **4** | **8**} [**value** {*low-value*} [{*high-value*}]]
10. **pw-class** *name*
11. **end**
    or
    commit

12. **pw-class** *name*
13. **encapsulation l2tpv3**
14. **ipv4 source** *ip-address*
15. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configure submode.
Step 3	**xconnect group** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group customer_X`	Enters a name for the cross-connect group.
Step 4	**p2p** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1`	Enters p2p configuration submode to configure point-to-point cross-connects.
Step 5	**neighbor** *ip-address* **pw-id** *number* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 666`	Configures a pseudowire for a cross-connect.
Step 6	**l2tp static local session** {*session-id*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static local session 147`	Configures a L2TP pseudowire static session ID.
Step 7	***l2tp static local cookie size*** *{***0** | **4** | **8***} [**value** {low-value} [{high-value}]]* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static local cookie size 4 value 0xbeef`	Configures a L2TP pseudowire static session cookie.

	Command or Action	Purpose
Step 8	**l2tp static remote session** {*session-id*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static remote session 123`	Configures a L2TP pseudowire remote session ID.
Step 9	**l2tp static remote cookie size** {*0* | *4* | *8*} [**value** {*low-value*} [{*high-value*}]] **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static remote cookie size 8 value 0x456 0xFFB`	Configures a L2TP pseudowire remote session cookie.
Step 10	**pw-class** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class class100`	Enters pseudowire class submode to define a pseudowire class template.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 12	**pw-class** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class class100`	Enters pseudowire class submode to define a pseudowire class template.
Step 13	**encapsulation l2tpv3** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`	Configures L2TPv3 pseudowire encapsulation.

	Command or Action	Purpose
Step 14	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# ipv4 source 127.0.0.1`	Configures the local source IPv4 address.

Step 15	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Cross-connect Attachment Circuit

This configuration procedure binds an Ethernet 802.1q VLAN, or Frame Relay AC to an L2TPv3 pseudowire for cross-connect service. The virtual circuit identifier that you configure creates the binding between a pseudowire configured on a PE router and an AC in a CE device. The virtual circuit identifier configured on the PE router at one end of the L2TPv3 control-channel must also be configured on the peer PE router at the other end.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *free_format_string*
4. **p2p** *name*
5. **interface** *interface_name*
6. **neighbor** *ip-address* **pw-id** *number*
7. **pw-class** *name*

8. **end**
   or
   commit
9. **pw-class** *name*
10. **encapsulation l2tpv3**
11. **ipv4 source** *ip-address*
12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enter L2VPN configure submode.
Step 3	**xconnect group** *free_format_string* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group customer_X`	Configures a cross-connect group.
Step 4	**p2p** *xconnect_id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1`	Enters p2p configuration submode to configure point-to-point cross-connects.
Step 5	**interface** *interface_name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface pos 1/1/1/1`	Enters interface configuration mode.

Step 6	**neighbor** *ip-address* **pw-id** *number*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 665`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)`
		Configures a pseudowire for a cross-connect.

	Command or Action	Purpose
Step 7	**pw-class** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw) pw-class l2tpv3-encap`	Enters pseudowire class submode to define a pseudowire class template.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**pw-class** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class l2tpv3-encap`	Enters pseudowire class submode to define a pseudowire class template.
Step 10	**encapsulation l2tpv3** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`	Configures L2TPv3 pseudowire encapsulation.
Step 11	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3) # ipv4 source 127.0.0.1`	Configures the local source IPv4 address.

	Command or Action	Purpose

Step 12	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Layer 2 Tunnel Protocol Version 3

This section provides the following configuration examples:

- Configuring an L2TP Class for L2TPv3-based L2VPN PE Routers: Example, page MPC-318
- Configuring a Pseudowire Class: Example, page MPC-319
- Configuring L2TPv3 Control Channel Parameters: Example, page MPC-319
- Configuring the Cross-Connect Group: Example, page MPC-319
- Configuring an Interface for Layer 2 Transport Mode: Example, page MPC-319

Configuring an L2TP Class for L2TPv3-based L2VPN PE Routers: Example

The following example shows how to configure a L2TP class with L2TPv3 based L2VPN for a PE router.

```
configure
  l2tp-class l2tptest
    receive-window 256
    retransmit retries 8
    retransmit initial retries 10
    retransmit initial timeout max 4
    retransmit initial timeout min 2
    timeout setup 90
    hostname PE1
    hello-interval 100
    digest secret cisco hash MD5
  end
```

Configuring a Pseudowire Class: Example

The following example shows a pseudowire class configuration on a PE router:

```
configure
 l2vpn
  pw-class FR1
   encapsulation l2tpv3
    protocol l2tpv3 class FR-l2tp
    tos value 100 reflect
    ttl 50
    ipv4 source 127.0.0.1
    cookie size 4
    sequencing both resync 150
```

Configuring L2TPv3 Control Channel Parameters: Example

The following example shows a typical L2TPv3 control-channel configuration:

```
configure
 l2tp-class FR-l2tp
  authentication
  hostname R2-PE1
  password 7 121A0C041104
  hello-interval 10
  digest secret 7 02050D480809
```

Configuring the Cross-Connect Group: Example

The following example shows how to group all cross -connects for FR1:

```
configure
 l2vpn
  xconnect group FR1
   p2p FR1
    interface Serial0/3/3/0/3/1:0.101
    neighbor 10.1.1.1 pw-id 2001
     pw-class FR1
```

Configuring an Interface for Layer 2 Transport Mode: Example

The following example shows how to configure an interface to operate in Layer 2 transport mode:

```
configure
 interface Serial0/3/3/0/3/1:0
  encapsulation frame-relay
  frame-relay lmi-type ansi
  exit
 interface Serial0/3/3/0/3/1:0.101 l2transport
  pvc 101
```

Additional References

The following sections provide additional information related to L2TPv3.

Related Documents

Related Topic	Document Title
MPLS VPN-related commands	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS Layer 3 VPNs	*Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
Getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
draft-ietf-l2tpext-l2tp-base-03.txt	Layer Two Tunneling Protocol (Version 3) L2TPv3

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1321	*The MD5 Message Digest Algorithm*
RFC 2104	*HMAC-Keyed Hashing for Message Authentication*

RFCs	Title
RFC 2661	*Layer Two Tunneling Protocol "L2TP"*
RFC 3931	*Layer Two Tunneling Protocol Version 3 "L2TPv3*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software

The MPLS VPNs over IP Tunnels feature lets you deploy Layer 3 Virtual Private Network (L3VPN) services, over an IP core network, using L2TPv3 multipoint tunneling instead of MPLS. This allows L2TPv3 tunnels to be configured as multipoint tunnels to transport IP VPN services across the core IP network.

Note This feature is available on the Cisco XR 12000 Series Router only.

Feature History for Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software

Release	Modification
Release 3.5.0	This feature was introduced on the Cisco XR 12000 Series Router.
Release 3.6.0	No modification.
Release 3.7.0	No modification.

Contents

- Prerequisites for Configuring MPLS VPNs over IP Tunnels, page MPC-324
- Restrictions for Configuring MPLS VPNs over IP Tunnels, page MPC-324
- Information About MPLS VPNs over IP Tunnels, page MPC-324
- How to Configure MPLS VPNs over IP Tunnels, page MPC-327
- Configuration Examples for MPLS VPNs over IP Tunnels, page MPC-342
- Additional References, page MPC-344

Prerequisites for Configuring MPLS VPNs over IP Tunnels

The following prerequisites are required to implement MPLS VPNs over IP Tunnels:

- You must be in a user group associated with a task group that includes the proper task IDs for
  - BGP commands
  - MPLS commands (generally)
  - MPLS Layer 3 VPN commands

For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*.

Restrictions for Configuring MPLS VPNs over IP Tunnels

The following restrictions apply when you configure MPLS VPNs over IP tunnels:

- MPLS forwarding cannot be enabled on a provider edge (PE) router.

Information About MPLS VPNs over IP Tunnels

To implement MPLS VPNs over IP Tunnels, you must understand the following concepts:

- Overview: MPLS VPNs over IP Tunnels, page MPC-324
- Advertising Tunnel Type and Tunnel Capabilities Between PE Routers—BGP, page MPC-325
- PE Routers and Address Space, page MPC-325
- Packet Validation Mechanism, page MPC-326
- Quality of Service Using the Modular QoS CLI, page MPC-326
- BGP Multipath Load Sharing for MPLS VPNs over IP Tunnels, page MPC-326
- Inter-AS and CSC Support over IP Tunnels, page MPC-327

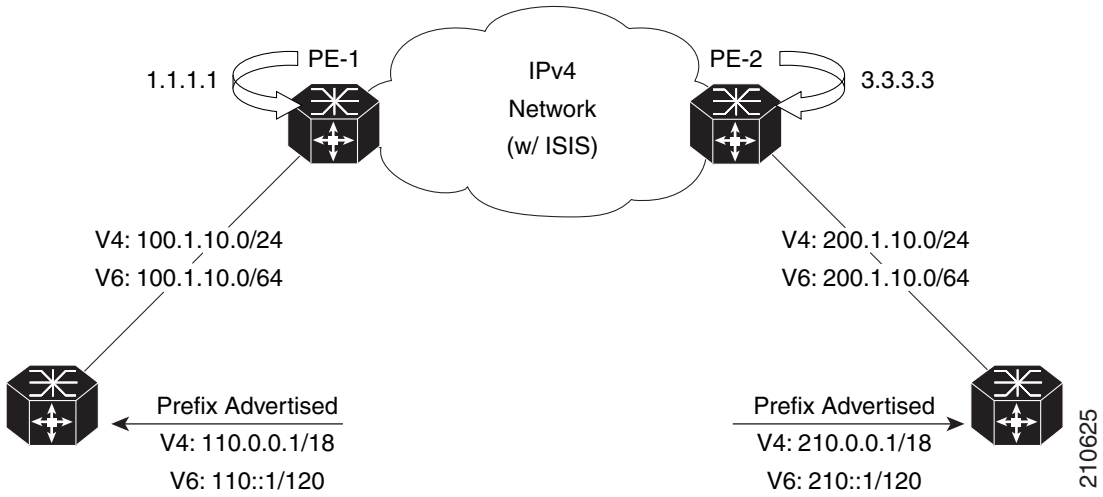
Overview: MPLS VPNs over IP Tunnels

Traditionally, VPN services are deployed over IP core networks using MPLS, *or* L2TPv3 tunnels using point-to-point links. However, an L2TPv3 multipoint tunnel network allows L3VPN services to be carried through the core without the configuration of MPLS.

L2TPv3 multipoint tunneling supports multiple tunnel endpoints, which creates a full-mesh topology that requires only one tunnel to be configured on each PE router. This permits VPN traffic to be carried from enterprise networks across cooperating service provider core networks to remote sites.

Figure 25 illustrates the topology used for the configuration steps.

Figure 25 Basic MPLS VPN over IP Topology



Advertising Tunnel Type and Tunnel Capabilities Between PE Routers—BGP

Border Gateway Protocol (BGP) is used to advertise the tunnel endpoints and the subaddress family identifier (SAFI) specific attributes (which contains the tunnel type, and tunnel capabilities). This feature introduces the tunnel SAFI and the BGP SAFI-Specific Attribute (SSA) attribute.

These attributes allow BGP to distribute tunnel encapsulation information between PE routers. VPNv4 traffic is routed through these tunnels. The next hop, advertised in BGP VPNv4 updates, determines which tunnel to use for routing tunnel traffic.

SAFI

The tunnel SAFI defines the tunnel endpoint and carries the endpoint IPv4 address and next hop. It is identified by the SAFI number 64.

BGP SSA

The BGP SSA carries the BGP preference and BGP flags. It also carries the tunnel cookie, tunnel cookie length, and session ID. It is identified by attribute number 19.

PE Routers and Address Space

One multipoint L2TPv3 tunnel must be configured on each PE router. To create the VPN, you must configure a unique Virtual Routing and Forwarding (VRF) instance. The tunnel that transports the VPN traffic across the core network resides in its own address space. A special purpose VRF called a *Resolve in VRF* (RiV) is created to manage the tunnel address space. You also configure the address space under the RiV that is associated with the tunnel and a static route in the RiV to route outgoing traffic through the tunnel.

Packet Validation Mechanism

The MPLS VPNs over IP Tunnels feature provides a simple mechanism to validate received packets from appropriate peers. The multipoint L2TPv3 tunnel header is automatically configured with a 64-bit cookie and L2TPv3 session ID. This packet validation mechanism protects the VPN from illegitimate traffic sources. The cookie and session ID are not user-configurable, but they are visible in the packet as it is routed between the two tunnel endpoints. Note that this packet validation mechanism does not protect the VPN from hackers who are able to monitor legitimate traffic between PE routers.

Quality of Service Using the Modular QoS CLI

To configure the bandwidth on the encapsulation and decapsulation interfaces, use the modular QoS CLI (MQC).

Note This task is optional.

Use the MQC to configure the IP precedence or Differentiated Services Code Point (DSCP) value set in the IP carrier header during packet encapsulation. To set these values, enter a standalone **set** command or a **police** command using the keyword **tunnel**. In the input policy on the encapsulation interface, you can set the precedence or DSCP value in the IP payload header by using MQC commands without the keyword **tunnel**.

Note You must attach a QoS policy to the physical interface—*not* to the tunnel interface.

If Modified Deficit Round Robin (MDRR)/Weighted Random Early Detection (WRED) is configured for the encapsulation interface in the input direction, the final value of the precedence or DSCP field in the IP carrier header is used to determine the precedence class for which the MDRR/WRED policy is applied. On the decapsulation interface in the input direction, you can configure a QoS policy based on the precedence or DSCP value in the IP carrier header of the received packet. In this case, an MQC policy with a class to match on precedence or DSCP value will match the precedence or DSCP value in the received IP carrier header. Similarly, the precedence class for which the MDRR/WRED policy is applied on the decapsulation input direction is also determined by precedence or DSCP value in the IP carrier header.

BGP Multipath Load Sharing for MPLS VPNs over IP Tunnels

BGP Multipath Load Sharing for EBGP and IBGP lets you configure multipath load balancing with both external BGP and internal BGP paths in BGP networks that are configured to use MPLS VPNs. (When faced with multiple routes to the same destination, BGP chooses the best route for routing traffic toward the destination so that no individual router is overburdened.)

BGP Multipath Load Sharing is useful for multihomed autonomous systems and PE routers that import both EBGP and IBGP paths from multihomed and stub networks.

Inter-AS and CSC Support over IP Tunnels

The L3VPN Inter-AS feature provides a method of interconnecting VPNs between different VPN service providers. Inter-AS supports connecting different VPN service providers to provide native IP L3VPN services. For more information about Inter-AS, see Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software.

Carrier Supporting Carrier (CSC) is implemented in circumstances in which one service provider needs to use the transport services provided by another service provider. The service provider that provides the transport is called the backbone carrier. The service provider, which uses the services provided by the backbone carrier, is called a customer carrier. Backbone carriers with CSC, bridge two or more customer carrier sites through an MPLS VPN/MPLS VPN over IP tunnels backbone. For more information about CSC, see Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software.

How to Configure MPLS VPNs over IP Tunnels

The following procedures are required to configure MPLS VPN over IP:

- Configuring the Global VRF Definition, page MPC-327 (required)
- Configuring a Route-Policy Definition, page MPC-329 (required)
- Configuring a Static Route, page MPC-330 (required)
- Configuring an IPv4 Loopback Interface, page MPC-331 (required)
- Configuring a CFI VRF Interface, page MPC-333 (required)
- Configuring the Core Network, page MPC-334 (required)
- Configuring Inter-AS and CSC Support over IP Tunnels, page MPC-335
- Verifying MPLS VPN over IP, page MPC-342 (optional)

Note All procedures occur on the local PE (PE1). Corresponding procedures must be configured on the remote PE (PE2).

Configuring the Global VRF Definition

Perform this task to configure the global VRF definition.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*]
5. **export route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*]
6. **exit**
7. **address-family ipv6 unicast**
8. **import route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*]

9. **export route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*]
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config)# vrf vrf-name`	Specifies a name assigned to a VRF.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Specifies an IPv4 address-family address.
Step 4	**import route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/0/CPU0:router(config-vrf-af)# import route-target 500:99`	Configures a VPN routing and forwarding (VRF) import route-target extended community.
Step 5	**export route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/0/CPU0:router(config-vrf-af)# export route-target 700:44`	Configures a VPN routing and forwarding (VRF) export route-target extended community.
Step 6	**exit** **Example:** `RP/0/0/CPU0:router(config-vrf-af)# exit`	Exits interface configuration mode.
Step 7	**address-family ipv6 unicast** **Example:** `RP/0/0/CPU0:router(config-vrf)# address-family ipv6 unicast`	Specifies an IPv6 address-family address.

	Command or Action	Purpose
Step 8	**import route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/0/CPU0:router(config-vrf-af)# import route-target 500:99`	Configures a VPN routing and forwarding (VRF) import route-target extended community.
Step 9	**export route-target** [*0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/0/CPU0:router(config-vrf-af)# import route-target 700:88`	Configures a VPN routing and forwarding (VRF) export route-target extended community.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# end`
	or
	`RP/0/0/CPU0:router(config-vrf-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Route-Policy Definition

Perform this task to configure a route-policy definition for CE-PE EBGP.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name* **pass**
3. **end policy**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy** *name* **pass** **Example:** `RP/0/0/CPU0:router(config)# route-policy ottawa_admin pass`	Defines and passes a route policy.
Step 3	**end policy** **Example:** `RP/0/0/CPU0:router(config-rpl)# end policy`	End of route-policy definition.

Configuring a Static Route

Perform this task to add more than 4K static routes (Global/VRF).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **maximum path ipv4** *1-140000*
4. **maximum path ipv6** *1-140000*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router static** **Example:** `RP/0/0/CPU0:router(config)# router static`	Enters static route configuration subcommands.

	Command or Action	Purpose
Step 3	**maximum path ipv4** *1-140000* **Example:** `RP/0/0/CPU0:router (config-static)# maximum path ipv4 1-140000`	Enters the maximum number of static ipv4 paths that can be configured.
Step 4	**maximum path ipv6** *1-140000* **Example:** `RP/0/0/CPU0:router(config-static)# maximum path ipv6 1-140000`	Enters the maximum number of static ipv6 paths that can be configured.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-static)# end`
	or
	`RP/0/0/CPU0:router(config-static)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an IPv4 Loopback Interface

The following task describes how to configure an IPv4 Loopback interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id*
3. **ipv4 address** *ipv4-address*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-id* **Example:** `RP/0/0/CPU0:router(config)# interface Loopback0`	Enters interface configuration mode and enables a Loopback interface.
Step 3	**ipv4 address** *ipv4-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 1.1.1.1 255.255.255.255`	Enters an IPv4 address and mask for the associated IP subnet. The network mask can be specified in either of two ways: • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are the network address.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a CFI VRF Interface

Perform this task to associate a VPN routing and forwarding (VRF) instance with an interface or a subinterface on the PE routers.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address*
5. **ipv6 address** *ipv6-address*
6. **dot1q vlan** *vlan-id*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-id* **Example:** `RP/0/0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1`	Enters interface configuration mode and enables a GigabitEthernet interface.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-if)# vrf v1`	Specifies a VRF name.
Step 4	**ipv4 address** *ipv4-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 100.1.10.2 255.255.255.0`	Enters an IPv4 address and mask for the associated IP subnet. The network mask can be specified in either of two ways: • The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address. • The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address.

	Command or Action	Purpose
Step 5	**ipv6 address** *ipv6-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv6 100::1:10:2/64`	Enters an IPv6 address. This argument must be in the form documented in RFC 2373, where the address is specified in hexadecimal using 16-bit values between colons, as follows: • IPv6 name or address: Hostname or X:X::X%zone • IPv6 prefix: X:X::X%zone/<0-128>
Step 6	**dot1q native vlan** *vlan-id* **Example:** `RP/0/0/CPU0:router(config-if)# dot1q native vlan 665`	Enters the trunk interface ID. Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Core Network

To configure the core network, refer to the procedures documented in *Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software.*

The tasks are presented as follows:

- Assessing the needs of MPLS VPN customers
- Configuring routing protocols in the core
- Configuring MPLS in the core
- Enabling FIB in the core
- Configuring BGP on the PE routers and route reflectors

Configuring Inter-AS and CSC Support over IP Tunnels

These tasks describe how to configure Inter-AS and CSC support over IP tunnels:

- Configuring the ASBRs to Exchange VPN-IPv4 Addresses for IP Tunnels, page MPC-335 (required)
- Configuring the Backbone Carrier Core for IP Tunnels, page MPC-338
- Configuring CSC-PE Routers for IP Tunnels, page MPC-338

Configuring the ASBRs to Exchange VPN-IPv4 Addresses for IP Tunnels

Perform this task to configure an external Border Gateway Protocol (eBGP) autonomous system boundary router (ASBR) to exchange VPN-IPv4 routes with another autonomous system for IP tunnels

Note This procedure is supported on the Cisco XR 12000 Series Router.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 tunnel**}
4. **address-family** {**vpnv4 unicast**}
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **address-family** {**vpnv4 unicast**}
8. **route-policy** *route-policy-name* {**in**}
9. **route-policy** *route-policy-name* {**out**}
10. **neighbor** *ip-address*
11. **remote-as** *autonomous-system-number*
12. **update-source** *interface-type interface-number*
13. **address-family** {**ipv4 tunnel**}
14. **address-family** {**vpnv4 unicast**}
15. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**ipv4 tunnel**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family ipv4 tunnel`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures IPv4 tunnel address family.

Step 4	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/0/CPU0:router(cconfig-bgp-af)#`
	`address-family vpnv4 unicast`
		Configures VPNv4 address family.

Step 5	**neighbor** *ip-address*
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

Step 7	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
	`address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 8	**route-policy** *route-policy-name* {**in**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
	`route-policy pass-all in`
		Applies a routing policy to updates that are received from a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **in** keyword to define the policy for inbound routes.

	Command or Action	Purpose
Step 9	**route-policy** *route-policy-name* {**out**} **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies a routing policy to updates that are sent from a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **out** keyword to define the policy for outbound routes.

Step 10	**neighbor** *ip-address*
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# neighbor 175.40.25.2`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 175.40.25.2 as an VPNv4 iBGP peer.

Step 11	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 12	**update-source** *interface-type interface-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# update-source loopback0`	Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 13	**address-family** {**ipv4 tunnel**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 tunnel`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures IPv4 tunnel address family.

	Command or Action	Purpose
Step 14	**address-family** {**vpnv4 unicast**} **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# address-family vpnv4 unicast`	Configures VPNv4 address family.

Step 15	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Backbone Carrier Core for IP Tunnels

Configuring the backbone carrier core requires setting up connectivity and routing functions for the CSC core and the CSC-PE routers. To do so, you must complete the following high-level tasks:

- Verify IP connectivity in the CSC core.
- Configure IP tunnels in the core.
- Configure VRFs for CSC-PE routers.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

Configuring CSC-PE Routers for IP Tunnels

Perform this task to configure a CSC-PE for IP tunnels.

For information on how to configure CSC-CE routers, see the Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software module.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** {**vpnv4 unicast**}

4. **address-family** {**ipv4 tunnel**}
5. **neighbor** *A.B.C.D*
6. **remote-as** *as-number*
7. **update-source** *interface-type interface-number*
8. **address-family** {**vpnv4 unicast**}
9. **address-family** {**ipv4 tunnel**}
10. **vrf** *vrf-name*
11. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}
12. **address-family** {**ipv4 unicast**}
13. **allocate-label all**
14. **neighbor** *A.B.C.D*
15. **remote-as** *as-number*
16. **address-family** {**ipv4 labeled-unicast**}
17. **route-policy** *route-policy-name* **in**
18. **route-policy** *route-policy-name* **out**
19. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *as-number*
	
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 2`
	`RP/0/0/CPU0:router(config-bgp)#`
		Configures a BGP routing process and enters router configuration mode.
		• Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

Step 4	**address-family** {**ipv4 tunnel**} **Example:** `RP/0/0/CPU0:router(config-bgp-af)# address-family ipv4 tunnel`	Configures IPv4 tunnel address family.

	Command or Action	Purpose

Step 5	**neighbor** *A.B.C.D*
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.10.10.0`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Configures the IP address for the BGP neighbor.

Step 6	**remote-as** *as-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 888`	Configures the AS number for the BGP neighbor.
Step 7	**update-source** *interface-type interface-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# update-source loopback0`	Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 8	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 unicast address family.

Step 9	**address-family** {**ipv4 tunnel**} **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# address-family ipv4 tunnel`	Configures IPv4 tunnel address family.

Step 10	**vrf** *vrf-name*
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf 9999`
	`RP/0/0/CPU0:router(config-bgp-vrf)#`
		Configures a VRF instance.

Step 11	**rd** {*as-number:nn* | *ip-address:nn* | **auto**} **Example:** `RP/0/0/CPU0:router(config-bgp-vrf)# rd auto`	Configures a route distinguisher. **Note** Use the **auto** keyword to automatically assign a unique route distinguisher.

Step 12	**address-family** {**ipv4 unicast**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-vrf-af)#`
		Configures IPv4 unicast address family.

Step 13	**allocate-label all** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-af)# allocate-label all`	Allocate labels for all local prefixes and prefixes received with labels.

	Command or Action	Purpose

Step 14	**neighbor** *A.B.C.D*
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-af)# neighbor 10.10.10.0`
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)#`
		Configures the IP address for the BGP neighbor.

Step 15	**remote-as** *as-number* **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr)# remote-as 888`	Enables the exchange of information with a neighboring BGP router.

Step 16	**address-family** {**ipv4 labeled-unicast**}
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)#`
		Configures IPv4 labeled-unicast address family.

Step 17	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all in`	Applies the pass-all policy to all inbound routes.
Step 18	**route-policy** *route-policy-name* **out** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out`	Applies the pass-all policy to all outbound routes.

Step 19	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying MPLS VPN over IP

To verify the configuration of end-end (PE-PE) MPLS VPN over IP provisioning, use the following **show** commands:

- **show cef recursive-nexthop**
- **show bgp ipv4 tunnel**
- **show bgp vpnv4 unicast summary**
- **show bgp vrf v1 ipv4 unicast summary**
- **show bgp vrf v1 ipv4 unicast** *prefix*
- **show cef vrf v1 ipv4** *prefix*
- **show cef ipv6 recursive-nexthop**
- **show bgp vpnv6 unicast summary**
- **show bgp vrf v1 ipv6 unicast summary**
- **show bgp vrf v1 ipv6 unicast** *prefix*
- **show cef vrf v1 ipv6** *prefix*

Configuration Examples for MPLS VPNs over IP Tunnels

This section provides the following examples:

- Configuring an L2TPv3 Tunnel: Example, page MPC-342
- Configuring the Global VRF Definition: Example, page MPC-342
- Configuring a Route-Policy Definition: Example, page MPC-343
- Configuring a Static Route: Example, page MPC-343
- Configuring an IPv4 Loopback Interface: Example, page MPC-343
- Configuring a CFI VRF Interface: Example, page MPC-343

Configuring an L2TPv3 Tunnel: Example

The following example shows how to configure an L2TPv3 tunnel:

```
tunnel-template t1
 encapsulation l2tp
 !
 source Loopback0
!
```

Configuring the Global VRF Definition: Example

The following example shows how to configure an L2TPv3 tunnel:

```
vrf v1
 address-family ipv4 unicast
 import route-target
   1:1
```

```
  !
  export route-target
   1:1
  !
 address-family ipv6 unicast
  import route-target
   1:1
!
  export route-target
   1:1
  !
```

Configuring a Route-Policy Definition: Example

The following example shows how to configure a route-policy definition:

```
configure
  route-policy pass-all
  pass
end-policy
!
```

Configuring a Static Route: Example

The following example shows how to configure a static route:

```
configure
  router static
  maximum path ipv4 <1-140000>
  maximum path ipv6 <1-140000>
end-policy
!
```

Configuring an IPv4 Loopback Interface: Example

The following example shows how to configure an IPv4 Loopback Interface:

```
configure
 interface Loopback0
 ipv4 address 1.1.1.1 255.255.255.255
!
```

Configuring a CFI VRF Interface: Example

The following example shows how to configure an L2TPv3 tunnel:

```
configure
  interface GigabitEthernet0/0/0/1.1
  vrf v1
   ipv4 address 100.1.10.2 255.255.255.0
   ipv6 address 100::1:10:2/64
   dot1q vlan 101
!
```

Additional References

For additional information related to this feature, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software*
Layer 2 Tunnel Protocol Version 3	*Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software*
Routing (BGP, EIGRP, OSPF, and RIP) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*
Routing (BGP, EIGRP, OSPF, and RIP) configuration	*Cisco IOS XR Routing Configuration Guide*
MPLS LDP configuration: configuration concepts, task, and examples	*Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software*
MPLS Traffic Engineering Resource Reservation Protocol configuration: configuration concepts, task, and examples	*Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software*
Cisco CRS-1 router getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 2547	*BGP/MPLS VPNs*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software

A Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of an MPLS provider core network. At each customer site, one or more customer edge (CE) routers attach to one or more provider edge (PE) routers.

This module provides the conceptual and configuration information for MPLS Layer 3 VPNs on Cisco IOS XR software.

Note In Release 3.5, you must acquire an evaluation or permanent license in order to use MPLS Layer 3 VPN functionality. However, if you are upgrading to Release 3.5 from a previous version of the software, MPLS Layer 3 VPN functionality will continue to work using an implicit license for 90 days (during which time, you can purchase a permanent license). For more information about licenses, see the *Software Entitlement on Cisco IOS XR Software* module in *Cisco IOS XR System Management Configuration Guide.*

For a description of the commands listed in this module search online in the Cisco IOS XR software master command index.

Feature History for Implementing MPLS Layer 3 VPN on Cisco IOS XR Configuration Module

Release	Modification
Release 3.3.0	This feature was introduced on the Cisco CRS-1 and Cisco XR 12000 Series Router.
Release 3.4.0	Support was added for MPLS L3VPN Carrier Supporting Carrier (CSC) functionality, including conceptual information and configuration tasks.
Release 3.4.1	No modification.
Release 3.5.0	Support was added for 6VPE. MPLS L3VPN Carrier Supporting Carrier (CSC) information was upgraded.
Release 3.6.0	Support was added for Inter-AS and CSC over IP Tunnels.
Release 3.7.0	Support was added for: • IPv6 VPN Provider Edge (6VPE) on the Cisco CRS-1. • Inter-AS support for 6VPE.

Contents

- MPLS L3VPN Prerequisites, page MPC-348
- Information About MPLS Layer 3 VPNs on Cisco IOS XR Software, page MPC-349
- Inter-AS Support for L3VPN, page MPC-354
- Carrier Supporting Carrier Support for L3VPN, page MPC-366
- IPv6 VPN Provider Edge (6VPE) Support, page MPC-369
- How to Implement MPLS Layer 3 VPNs on Cisco IOS XR Software, page MPC-371
- Configuring 6VPE Support, page MPC-424
- Configuration Examples for Implementing MPLS Layer 3 VPNs, page MPC-430
- Additional References, page MPC-440

MPLS L3VPN Prerequisites

The following prerequisites are required to configure MPLS Layer 3 VPN:

- You must be in a user group associated with a task group that includes the proper task IDs for
  - BGP commands
  - MPLS commands (generally)
  - MPLS Layer 3 VPN commands
- For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*.

The following prerequisites are required for configuring MPLS VPN Inter-AS with autonomous system boundary routers (ASBRs) exchanging VPN-IPV4 addresses or IPv4 routes and MPLS labels:

- Before configuring external Border Gateway Protocol (eBGP) routing between autonomous systems or subautonomous systems in an MPLS VPN, ensure that all MPLS VPN routing instances and sessions are properly configured (see How to Implement MPLS Layer 3 VPNs on Cisco IOS XR Software, page MPC-371 for procedures).
- The following tasks must be performed:
  - Define VPN routing instances
  - Configure BGP routing sessions in the MPLS core
  - Configure PE-to-PE routing sessions in the MPLS core
  - Configure BGP PE-to-CE routing sessions
  - Configure a VPN-IPv4 eBGP session between directly connected ASBRs

To configure MPLS Layer 3 VPNs, routers must support MPLS forwarding and Forwarding Information Base (FIB).

Information About MPLS Layer 3 VPNs on Cisco IOS XR Software

To implement MPLS Layer 3 VPNs, you need to understand the following concepts:

- MPLS L3VPN Overview, page MPC-349
- MPLS L3VPN Benefits, page MPC-350
- MPLS L3VPN Restrictions, page MPC-350
- How MPLS L3VPN Works, page MPC-351
- MPLS L3VPN Major Components, page MPC-353

MPLS L3VPN Overview

Before defining an MPLS VPN, VPN in general must be defined. A VPN is:

- An IP-based network delivering private network services over a public infrastructure
- A set of sites that are allowed to communicate with each other privately over the Internet or other public or private networks

Conventional VPNs are created by configuring a full mesh of tunnels or permanent virtual circuits (PVCs) to all sites in a VPN. This type of VPN is not easy to maintain or expand, as adding a new site requires changing each edge device in the VPN.

MPLS-based VPNs are created in Layer 3 and are based on the peer model. The peer model enables the service provider and the customer to exchange Layer 3 routing information. The service provider relays the data between the customer sites without customer involvement.

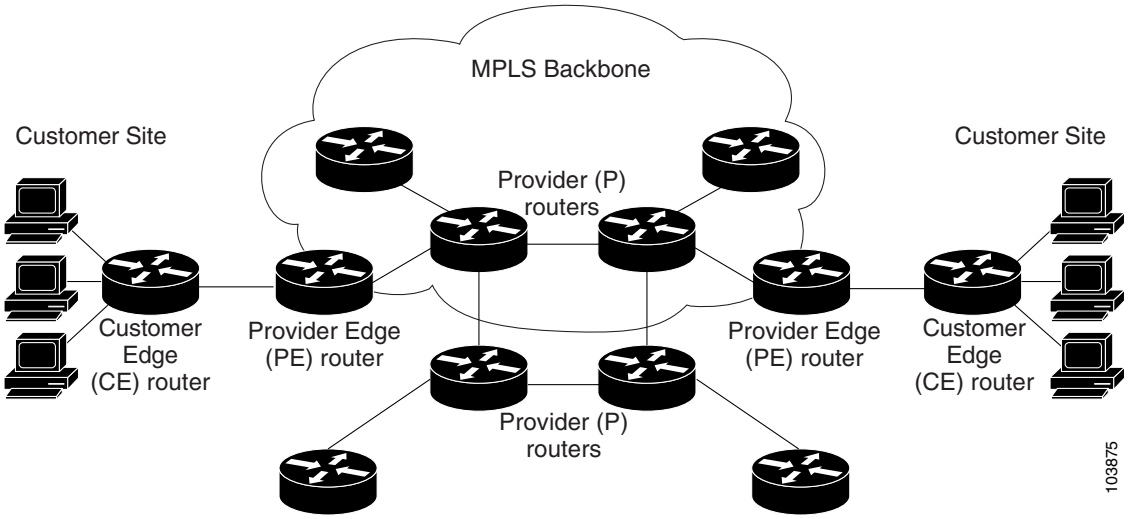
MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the edge router of the service provider that provides services to the customer site needs to be updated.

The components of the MPLS VPN are described as follows:

- Provider (P) router—Router in the core of the provider network. PE routers run MPLS switching and do not attach VPN labels to routed packets. VPN labels are used to direct data packets to the correct egress router.
- PE router—Router that attaches the VPN label to incoming packets based on the interface or subinterface on which they are received, and also attaches the MPLS core labels. A PE router attaches directly to a CE router.
- Customer (C) router—Router in the Internet service provider (ISP) or enterprise network.
- Customer edge (CE) router—Edge router on the network of the ISP that connects to the PE router on the network. A CE router must interface with a PE router.

Figure 26 shows a basic MPLS VPN topology.

Figure 26 Basic MPLS VPN Topology



MPLS L3VPN Benefits

MPLS L3VPN provides the following benefits:

- Service providers can deploy scalable VPNs and deliver value-added services.
- Connectionless service guarantees that no prior action is necessary to establish communication between hosts.
- Centralized Service: Building VPNs in Layer 3 permits delivery of targeted services to a group of users represented by a VPN.
- Scalability: Create scalable VPNs using connection-oriented, point-to-point overlays, Frame Relay, or ATM virtual connections.
- Security: Security is provided at the edge of a provider network (ensuring that packets received from a customer are placed on the correct VPN) and in the backbone.
- Integrated Quality of Service (QoS) support: QoS provides the ability to address predictable performance and policy implementation and support for multiple levels of service in an MPLS VPN.
- Straightforward Migration: Service providers can deploy VPN services using a straightforward migration path.
- Migration for the end customer is simplified. There is no requirement to support MPLS on the CE router and no modifications are required for a customer intranet.

MPLS L3VPN Restrictions

The following are restrictions for implementing MPLS Layer 3 VPNs:

- Multihop VPN-IPv4 eBGP is not supported for configuring eBGP routing between autonomous systems or subautonomous systems in an MPLS VPN.
- MPLS VPN supports only IPv4 address families.

The following restrictions apply when configuring MPLS VPN Inter-AS with ASBRs exchanging IPv4 routes and MPLS labels:

- For networks configured with eBGP multihop, a label switched path (LSP) must be configured between nonadjacent routers.
- Inter-AS supports IPv4 routes only. IPv6 is not supported.

Note The physical interfaces that connect the BGP speakers must support FIB and MPLS.

The following restrictions apply to routing protocols OSPF and RIP:

- IPv6 is not supported on OSPF and RIP.

How MPLS L3VPN Works

MPLS VPN functionality is enabled at the edge of an MPLS network. The PE router performs the following tasks:

- Exchanges routing updates with the CE router
- Translates the CE routing information into VPN version 4 (VPNv4) and VPN version 6 (VPNv6) routes
- Exchanges VPNv4 and VPNv6 routes with other PE routers through the Multiprotocol Border Gateway Protocol (MP-BGP)

Virtual Routing and Forwarding Tables

Each VPN is associated with one or more VPN routing and forwarding (VRF) instances. A VRF defines the VPN membership of a customer site attached to a PE router. A VRF consists of the following components:

- An IP version 4 (IPv4) unicast routing table
- A derived FIB table
- A set of interfaces that use the forwarding table
- A set of rules and routing protocol parameters that control the information that is included in the routing table

These components are collectively called a VRF instance.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A site can be a member of multiple VPNs. However, a site can associate with only one VRF. A VRF contains all the routes available to the site from the VPNs of which it is a member.

Packet forwarding information is stored in the IP routing table and the FIB table for each VRF. A separate set of routing and FIB tables is maintained for each VRF. These tables prevent information from being forwarded outside a VPN and also prevent packets that are outside a VPN from being forwarded to a router within the VPN.

VPN Routing Information: Distribution

The distribution of VPN routing information is controlled through the use of VPN route target communities, implemented by BGP extended communities. VPN routing information is distributed as follows:

- When a VPN route that is learned from a CE router is injected into a BGP, a list of VPN route target extended community attributes is associated with it. Typically, the list of route target community extended values is set from an export list of route targets associated with the VRF from which the route was learned.
- An import list of route target extended communities is associated with each VRF. The import list defines route target extended community attributes that a route must have for the route to be imported into the VRF. For example, if the import list for a particular VRF includes route target extended communities A, B, and C, then any VPN route that carries any of those route target extended communities—A, B, or C—is imported into the VRF.

BGP Distribution of VPN Routing Information

A PE router can learn an IP prefix from the following sources:

- A CE router by static configuration
- An eBGP session with the CE router
- A Routing Information Protocol (RIP) exchange with the CE router
- Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and RIP as Interior Gateway Protocols (IGPs)

The IP prefix is a member of the IPv4 address family. After the PE router learns the IP prefix, the PE converts it into the VPN-IPv4 prefix by combining it with a 64-bit route distinguisher. The generated prefix is a member of the VPN-IPv4 address family. It uniquely identifies the customer address, even if the customer site is using globally nonunique (unregistered private) IP addresses. The route distinguisher used to generate the VPN-IPv4 prefix is specified by the **rd** command associated with the VRF on the PE router.

BGP distributes reachability information for VPN-IPv4 prefixes for each VPN. BGP communication takes place at two levels:

- Within the IP domain, known as an autonomous system.
- Between autonomous systems.

PE to PE or PE to route reflector (RR) sessions are iBGP sessions, and PE to CE sessions are eBGP sessions. PE to CE eBGP sessions can be directly or indirectly connected (eBGP multihop).

BGP propagates reachability information for VPN-IPv4 prefixes among PE routers by the BGP protocol extensions (see RFC 2283, Multiprotocol Extensions for BGP-4), which define support for address families other than IPv4. Using the extensions ensures that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.

MPLS Forwarding

Based on routing information stored in the VRF IP routing table and the VRF FIB table, packets are forwarded to their destination using MPLS.

A PE router binds a label to each customer prefix learned from a CE router and includes the label in the network reachability information for the prefix that it advertises to other PE routers. When a PE router forwards a packet received from a CE router across the provider network, it labels the packet with the label learned from the destination PE router. When the destination PE router receives the labeled packet, it pops the label and uses it to direct the packet to the correct CE router. Label forwarding across the provider backbone is based on either dynamic label switching or traffic engineered paths. A customer data packet carries two levels of labels when traversing the backbone:

- The top label directs the packet to the correct PE router.
- The second label indicates how that PE router should forward the packet to the CE router.

More labels can be stacked if other features are enabled. For example, if traffic engineering (TE) tunnels with fast reroute (FRR) are enabled, the total number of labels imposed in the PE is four (Layer 3 VPN, Label Distribution Protocol (LDP), TE, and FRR).

Automatic Route Distinguisher Assignment

To take advantage of iBGP load balancing, every network VRF must be assigned a unique route distinguisher. VRFs require a route distinguisher for BGP to distinguish between potentially identical prefixes received from different VPNs.

With thousands of routers in a network each supporting multiple VRFs, configuration and management of route distinguishers across the network can present a problem. Cisco IOS XR simplifies this process by assigning unique route distinguisher to VRFs using the **rd auto** command.

To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across the VRFs.

Finally, route distinguisher values are checkpointed so that route distinguisher assignment to VRF is persistent across failover or process restart. If an route distinguisher is explicitely configured for a VRF, this value is not overridden by the autoroute distinguisher.

MPLS L3VPN Major Components

An MPLS-based VPN network has three major components:

- VPN route target communities—A VPN route target community is a list of all members of a VPN community. VPN route targets need to be configured for each VPN community member.
- Multiprotocol BGP (MP-BGP) peering of the VPN community PE routers—MP-BGP propagates VRF reachability information to all members of a VPN community. MP-BGP peering needs to be configured in all PE routers within a VPN community.
- MPLS forwarding—MPLS transports all traffic between all VPN community members across a VPN service-provider network.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A given site can be a member of multiple VPNs. However, a site can associate with only one VRF. A customer-site VRF contains all the routes available to the site from the VPNs of which it is a member.

Inter-AS Support for L3VPN

This section contains the following topics:

- Inter-AS Restrictions, page MPC-354
- Inter-AS Support: Overview, page MPC-354
- Inter-AS and ASBRs, page MPC-355
- Transmitting Information Between Autonomous Systems, page MPC-356
- Exchanging VPN Routing Information, page MPC-357
- Packet Forwarding, page MPC-359
- Confederations, page MPC-362
- MPLS VPN Inter-AS BGP Label Distribution, page MPC-364
- Exchanging IPv4 Routes with MPLS labels, page MPC-364

Inter-AS Restrictions

Inter-AS functionality is available using VPNv4 only. VPNv6 is not currently supported.

Inter-AS Support: Overview

An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and uses a single, clearly defined routing protocol.

As VPNs grow, their requirements expand. In some cases, VPNs need to reside on different autonomous systems in different geographic areas. In addition, some VPNs need to extend across multiple service providers (overlapping VPNs). Regardless of the complexity and location of the VPNs, the connection between autonomous systems must be seamless.

An MPLS VPN Inter-AS provides the following benefits:

- Allows a VPN to cross more than one service provider backbone.

  Service providers, running separate autonomous systems, can jointly offer MPLS VPN services to the same end customer. A VPN can begin at one customer site and traverse different VPN service provider backbones before arriving at another site of the same customer. Previously, MPLS VPN could traverse only a single BGP autonomous system service provider backbone. This feature lets multiple autonomous systems form a continuous, seamless network between customer sites of a service provider.

- Allows a VPN to exist in different areas.

  A service provider can create a VPN in different geographic areas. Having all VPN traffic flow through one point (between the areas) allows for better rate control of network traffic between the areas.

- Allows confederations to optimize iBGP meshing.

  Internal Border Gateway Protocol (iBGP) meshing in an autonomous system is more organized and manageable. You can divide an autonomous system into multiple, separate subautonomous systems and then classify them into a single confederation. This capability lets a service provider offer

MPLS VPNs across the confederation, as it supports the exchange of labeled VPN-IPv4 Network Layer Reachability Information (NLRI) between the subautonomous systems that form the confederation.

Inter-AS and ASBRs

Separate autonomous systems from different service providers can communicate by exchanging IPv4 NLRI in the form of VPN-IPv4 addresses. The ASBRs use eBGP to exchange that information. Then an Interior Gateway Protocol (IGP) distributes the network layer information for VPN-IPV4 prefixes throughout each VPN and each autonomous system. The following protocols are used for sharing routing information:

- Within an autonomous system, routing information is shared using an IGP.
- Between autonomous systems, routing information is shared using an eBGP. An eBGP lets service providers set up an interdomain routing system that guarantees the loop-free exchange of routing information between separate autonomous systems.

The primary function of an eBGP is to exchange network reachability information between autonomous systems, including information about the list of autonomous system routes. The autonomous systems use EBGP border edge routers to distribute the routes, which include label switching information. Each border edge router rewrites the next-hop and MPLS labels.

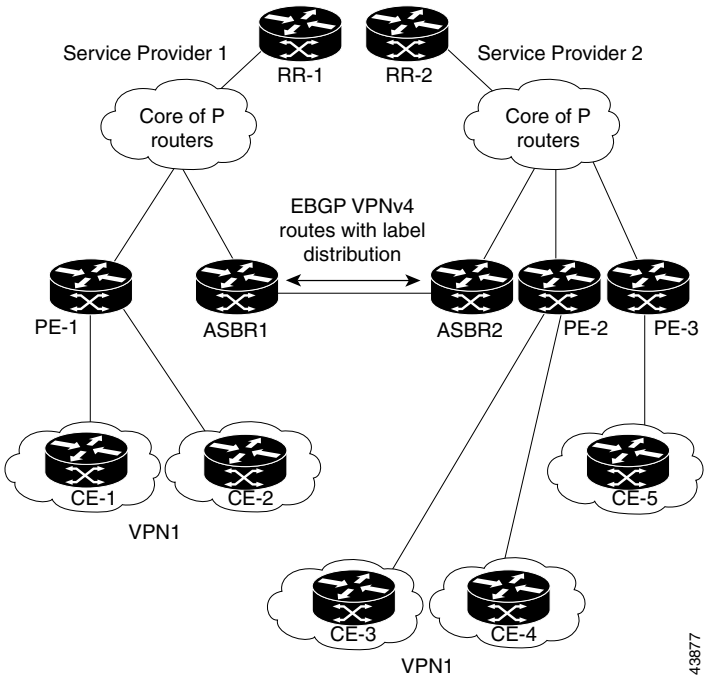
Inter-AS configurations supported in an MPLS VPN can include:

- Interprovider VPN—MPLS VPNs that include two or more autonomous systems, connected by separate border edge routers. The autonomous systems exchange routes using eBGP. No IGP or routing information is exchanged between the autonomous systems.
- BGP Confederations—MPLS VPNs that divide a single autonomous system into multiple subautonomous systems and classify them as a single, designated confederation. The network recognizes the confederation as a single autonomous system. The peers in the different autonomous systems communicate over eBGP sessions; however, they can exchange route information as if they were iBGP peers.

Transmitting Information Between Autonomous Systems

Figure 27 illustrates one MPLS VPN consisting of two separate autonomous systems. Each autonomous system operates under different administrative control and runs a different IGP. Service providers exchange routing information through eBGP border edge routers (ABSR1 and ASBR2).

Figure 27 eBGP Connection Between Two MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Addresses



This configuration uses the following process to transmit information:

Step 1 The provider edge router (PE-1) assigns a label for a route before distributing that route. The PE router uses the multiprotocol extensions of BGP to transmit label mapping information. The PE router distributes the route as a VPN-IPv4 address. The address label and the VPN identifier are encoded as part of the NLRI.

Step 2 The two route reflectors (RR-1 and RR-2) reflect VPN-IPv4 internal routes within the autonomous system. The border edge routers of the autonomous system (ASBR1 and ASBR2) advertise the VPN-IPv4 external routes.

Step 3 The eBGP border edge router (ASBR1) redistributes the route to the next autonomous system (ASBR2). ASBR1 specifies its own address as the value of the eBGP next-hop attribute and assigns a new label. The address ensures:

- That the next-hop router is always reachable in the service provider (P) backbone network.
- That the label assigned by the distributing router is properly interpreted. (The label associated with a route must be assigned by the corresponding next-hop router.)

Step 4 The eBGP border edge router (ASBR2) redistributes the route in one of the following ways, depending on the configuration:

- If the iBGP neighbors are configured with the **next-hop-self** command, ASBR2 changes the next-hop address of updates received from the eBGP peer, then forwards it.

- If the iBGP neighbors are not configured with the **next-hop-self** command, the next-hop address does not get changed. ASBR2 must propagate a host route for the eBGP peer through the IGP. To propagate the eBGP VPN-IPv4 neighbor host route, use the **redistribute** command with the **connected** keyword. The eBGP VPN-IPv4 neighbor host route is automatically installed in the routing table when the neighbor comes up. This automatic installation is essential to establish the label switched path between PE routers in different autonomous systems. You need to manually configure the static route to the next hop and redistribute it to IGP, to let other PE routers use the /32 host prefix label to forward traffic for an Inter-AS VPN redistribute connected option.

Note This option is not supported for Inter-AS over IP tunnels.

Exchanging VPN Routing Information

Autonomous systems exchange VPN routing information (routes and labels) to establish connections. To control connections between autonomous systems, the PE routers and eBGP border edge routers maintain a label forwarding information base (LFIB). The LFIB manages the labels and routes that the PE routers and eBGP border edge routers receive during the exchange of VPN information.

The autonomous systems use the following guidelines to exchange VPN routing information:

- Routing information includes:
  - The destination network (N)
  - The next-hop field associated with the distributing router
  - A local MPLS label (L)
- A route distinguisher (RD1). A route distinguisher is part of a destination network address. It makes the VPN-IPv4 route globally unique in the VPN service provider environment.
- The ASBRs are configured to change the next-hop when sending VPN-IPv4 NLRIs to the iBGP neighbors. Therefore, the ASBRs must allocate a new label when they forward the NLRI to the iBGP neighbors.

Figure 28 Exchanging Routes and Labels Between MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Address

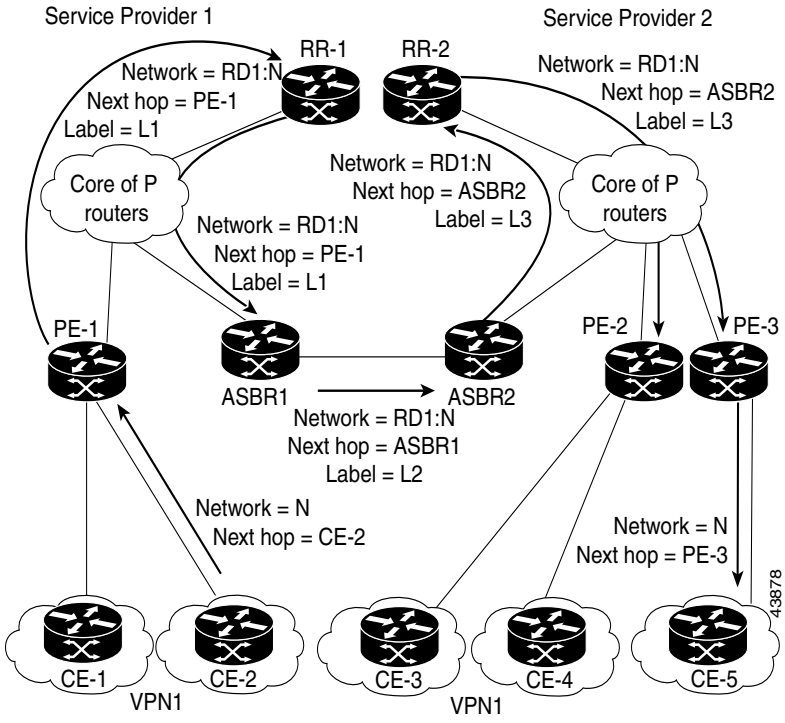
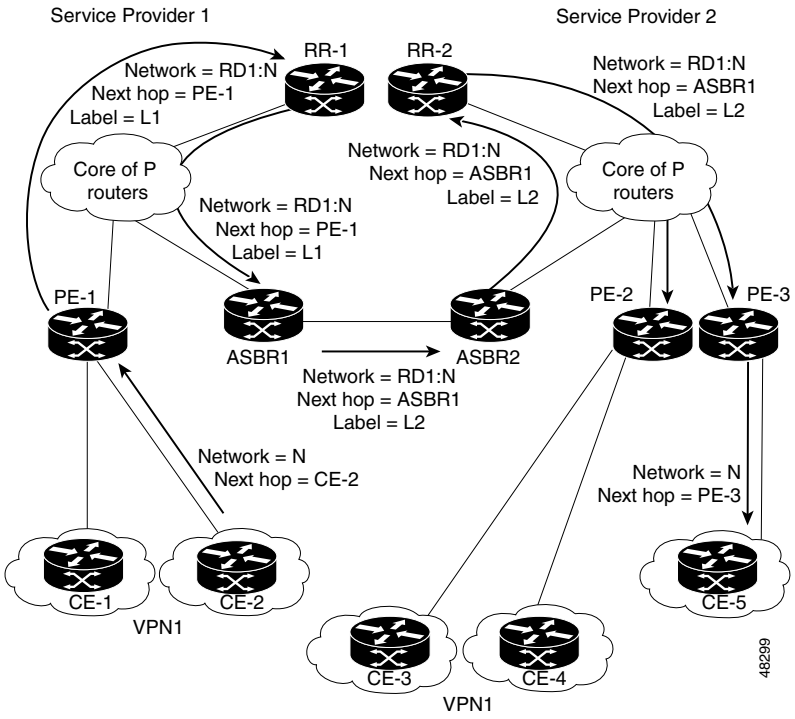


Figure 29 illustrates the exchange of VPN route and label information between autonomous systems. The only difference is that ASBR2 is configured with the **redistribute** command with the **connected** keyword, which propagates the host routes to all PEs. The command is necessary as ASBR2 is not configured to change the next-hop address.

Note Figure 29 is not applicable to Inter-AS over IP tunnels.

Figure 29 Exchanging Routes and Labels with the redistributed Command in an MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses



Packet Forwarding

Note This section is not applicable to Inter-AS over IP tunnels.

Figure 30 illustrates how packets are forwarded between autonomous systems in an interprovider network using the following packet method.

Packets are forwarded to their destination by means of MPLS. Packets use the routing information stored in the LFIB of each PE router and eBGP border edge router.

The service provider VPN backbone uses dynamic label switching to forward labels.

Each autonomous system uses standard multilevel labeling to forward packets between the edges of the autonomous system routers (for example, from CE-5 to PE-3). Between autonomous systems, only a single level of labeling is used, corresponding to the advertised route.

A data packet carries two levels of labels when traversing the VPN backbone:

- The first label (IGP route label) directs the packet to the correct PE router on the eBGP border edge router. (For example, the IGP label of ASBR2 points to the ASBR2 border edge router.)
- The second label (VPN route label) directs the packet to the appropriate PE router or eBGP border edge router.

Figure 30 Forwarding Packets Between MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Addresses

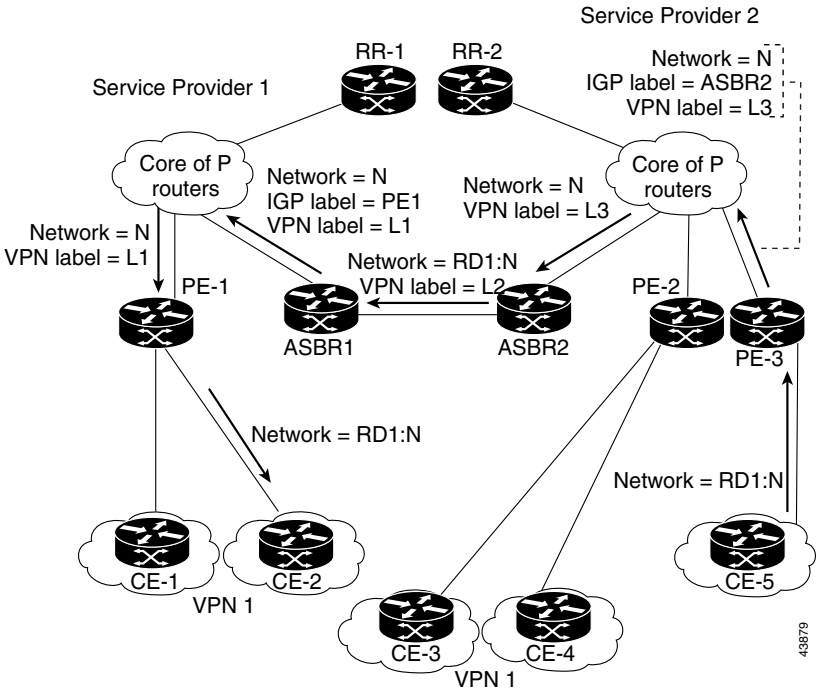


Figure 31 shows the same packet forwarding method, except the eBGP router (ASBR1) forwards the packet without reassigning a new label to it.

Figure 31 Forwarding Packets Without a New Label Assignment Between MPLS VPN Inter-AS System with ASBRs Exchanging VPN-IPv4 Addresses

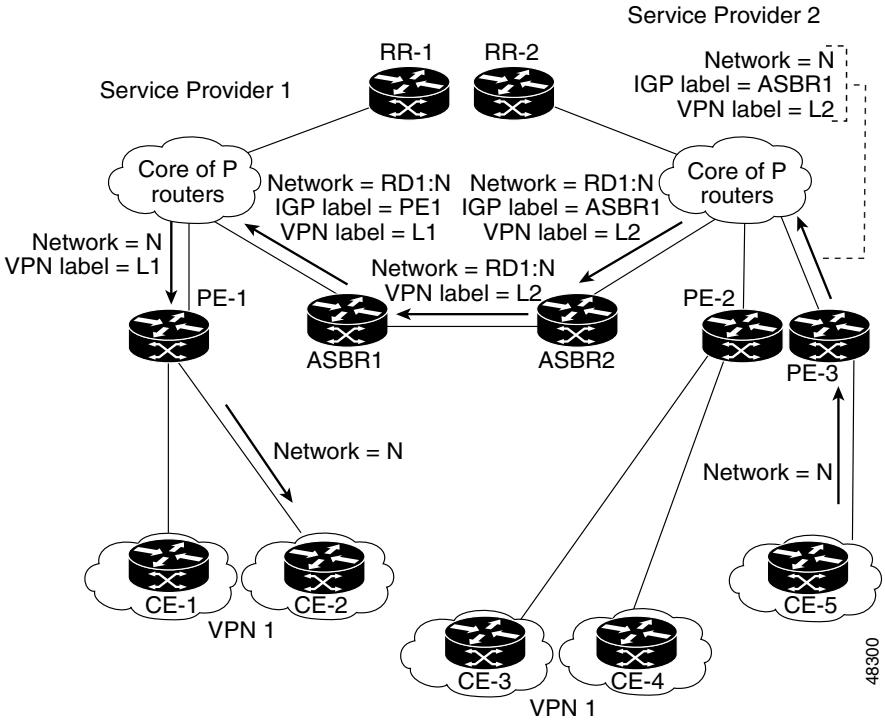
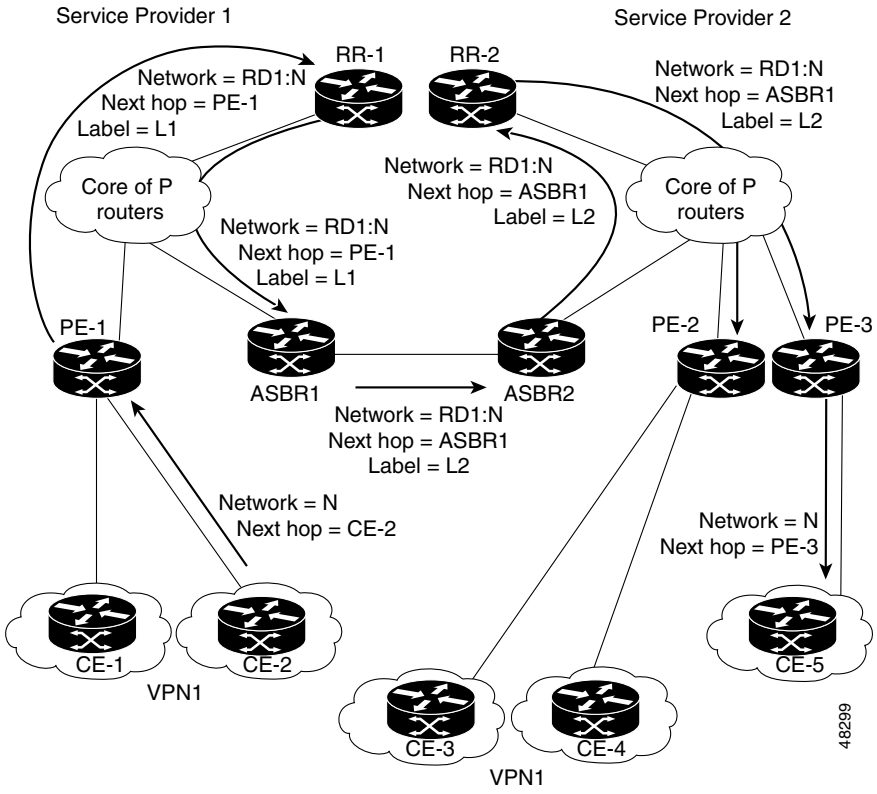


Figure 32 illustrates the exchange of VPN route and label information between autonomous systems.

Figure 32 Exchanging Routes and Labels in an MPLS VPN Inter-AS with ASBRs



Confederations

A confederation is multiple subautonomous systems grouped together. A confederation reduces the total number of peer devices in an autonomous system. A confederation divides an autonomous system into subautonomous systems and assigns a confederation identifier to the autonomous systems. A VPN can span service providers running in separate autonomous systems or multiple subautonomous systems that form a confederation.

In a confederation, each subautonomous system is fully meshed with other subautonomous systems. The subautonomous systems communicate using an IGP, such as Open Shortest Path First (OSPF) or Intermediate System-to-Intermediate System (IS-IS). Each subautonomous system also has an eBGP connection to the other subautonomous systems. The confederation eBGP (CEBGP) border edge routers forward next-hop-self addresses between the specified subautonomous systems. The next-hop-self address forces the BGP to use a specified address as the next hop rather than letting the protocol choose the next hop.

You can configure a confederation with separate subautonomous systems two ways:

- Configure a router to forward next-hop-self addresses between only the CEBGP border edge routers (both directions). The subautonomous systems (iBGP peers) at the subautonomous system border do not forward the next-hop-self address. Each subautonomous system runs as a single IGP domain. However, the CEBGP border edge router addresses are known in the IGP domains.

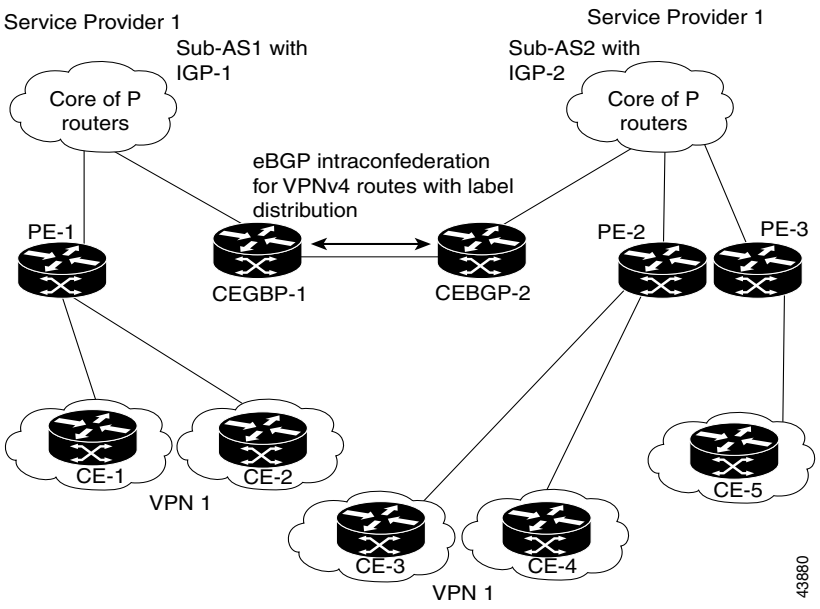
- Configure a router to forward next-hop-self addresses between the CEBGP border edge routers (both directions) and within the iBGP peers at the subautonomous system border. Each subautonomous system runs as a single IGP domain but also forwards next-hop-self addresses between the PE routers in the domain. The CEBGP border edge router addresses are known in the IGP domains.

Note Figure 27 illustrates how two autonomous systems exchange routes and forward packets. Subautonomous systems in a confederation use a similar method of exchanging routes and forwarding packets.

Figure 33 illustrates a typical MPLS VPN confederation configuration. In this configuration:

- The two CEBGP border edge routers exchange VPN-IPv4 addresses with labels between the two autonomous systems.
- The distributing router changes the next-hop addresses and labels and uses a next-hop-self address.
- IGP-1 and IGP-2 know the addresses of CEBGP-1 and CEBGP-2.

Figure 33 eBGP Connection Between Two Subautonomous Systems in a Confederation



In this confederation configuration:

- CEBGP border edge routers function as neighboring peers between the subautonomous systems. The subautonomous systems use eBGP to exchange route information.
- Each CEBGP border edge router (CEBGP-1 and CEBGP-2) assigns a label for the router before distributing the route to the next subautonomous system. The CEBGP border edge router distributes the route as a VPN-IPv4 address by using the multiprotocol extensions of BGP. The label and the VPN identifier are encoded as part of the NLRI.

- Each PE and CEBGP border edge router assigns its own label to each VPN-IPv4 address prefix before redistributing the routes. The CEBGP border edge routers exchange IPV-IPv4 addresses with the labels. The next-hop-self address is included in the label (as the value of the eBGP next-hop attribute). Within the subautonomous systems, the CEBGP border edge router address is distributed throughout the iBGP neighbors, and the two CEBGP border edge routers are known to both confederations.

For more information about how to configure confederations, see "Configuring MPLS Forwarding for ASBR Confederations" section on page MPC-409.

MPLS VPN Inter-AS BGP Label Distribution

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up the MPLS VPN Inter-AS network so that the ASBRs exchange IPv4 routes with MPLS labels of the provider edge (PE) routers. Route reflectors (RRs) exchange VPN-IPv4 routes by using multihop, multiprotocol external Border Gateway Protocol (eBGP). This method of configuring the Inter-AS system is often called MPLS VPN Inter-AS BGP Label Distribution.

Configuring the Inter-AS system so that the ASBRs exchange the IPv4 routes and MPLS labels has the following benefits:

- Saves the ASBRs from having to store all the VPN-IPv4 routes. Using the route reflectors to store the VPN-IPv4 routes and forward them to the PE routers results in improved scalability compared with configurations in which the ASBR holds all the VPN-IPv4 routes and forwards the routes based on VPN-IPv4 labels.
- Having the route reflectors hold the VPN-IPv4 routes also simplifies the configuration at the border of the network.
- Enables a non-VPN core network to act as a transit network for VPN traffic. You can transport IPv4 routes with MPLS labels over a non-MPLS VPN service provider.
- Eliminates the need for any other label distribution protocol between adjacent label switch routers (LSRs). If two adjacent LSRs are also BGP peers, BGP can handle the distribution of the MPLS labels. No other label distribution protocol is needed between the two LSRs.

Exchanging IPv4 Routes with MPLS labels

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up a VPN service provider network to exchange IPv4 routes with MPLS labels. You can configure the VPN service provider network as follows:

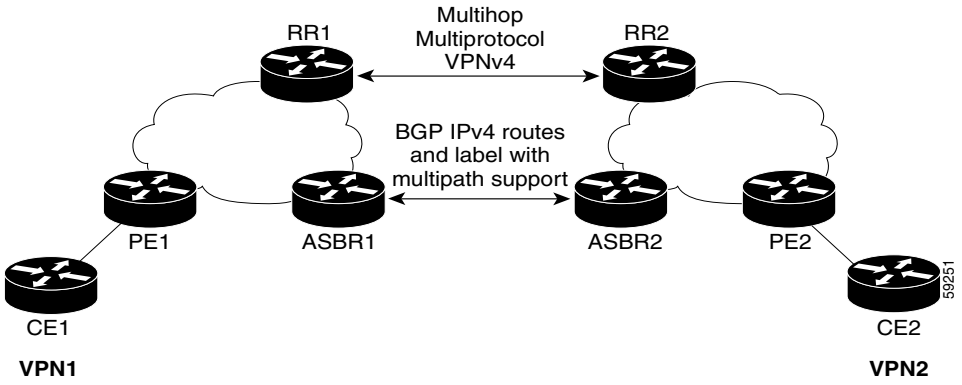
- Route reflectors exchange VPN-IPv4 routes by using multihop, multiprotocol eBGP. This configuration also preserves the next-hop information and the VPN labels across the autonomous systems.
- A local PE router (for example, PE1 in Figure 34) needs to know the routes and label information for the remote PE router (PE2).

This information can be exchanged between the PE routers and ASBRs in one of two ways:

- Internal Gateway Protocol (IGP) and Label Distribution Protocol (LDP): The ASBR can redistribute the IPv4 routes and MPLS labels it learned from eBGP into IGP and LDP and from IGP and LDP into eBGP.
- Internal Border Gateway Protocol (iBGP) IPv4 label distribution: The ASBR and PE router can use direct iBGP sessions to exchange VPN-IPv4 and IPv4 routes and MPLS labels.

  Alternatively, the route reflector can reflect the IPv4 routes and MPLS labels learned from the ASBR to the PE routers in the VPN. This reflecting of learned IPv4 routes and MPLS labels is accomplished by enabling the ASBR to exchange IPv4 routes and MPLS labels with the route reflector. The route reflector also reflects the VPN-IPv4 routes to the PE routers in the VPN. For example, in VPN1, RR1 reflects to PE1 the VPN-IPv4 routes it learned and IPv4 routes and MPLS labels learned from ASBR1. Using the route reflectors to store the VPN-IPv4 routes and forward them through the PE routers and ASBRs allows for a scalable configuration.

Figure 34 VPNs Using eBGP and iBGP to Distribute Routes and MPLS Labels



BGP Routing Information

BGP routing information includes the following items:

- Network number (prefix), which is the IP address of the destination.
- Autonomous system (AS) path, which is a list of the other ASs through which a route passes on the way to the local router. The first AS in the list is closest to the local router; the last AS in the list is farthest from the local router and usually the AS where the route began.
- Path attributes, which provide other information about the AS path, for example, the next hop.

BGP Messages and MPLS Labels

MPLS labels are included in the update messages that a router sends. Routers exchange the following types of BGP messages:

- Open messages—After a router establishes a TCP connection with a neighboring router, the routers exchange open messages. This message contains the number of the autonomous system to which the router belongs and the IP address of the router that sent the message.
- Update messages—When a router has a new, changed, or broken route, it sends an update message to the neighboring router. This message contains the NLRI, which lists the IP addresses of the usable routes. The update message includes any routes that are no longer usable. The update message also includes path attributes and the lengths of both the usable and unusable paths. Labels for VPN-IPv4 routes are encoded in the update message, as specified in RFC 2858. The labels for the IPv4 routes are encoded in the update message, as specified in RFC 3107.
- Keepalive messages—Routers exchange keepalive messages to determine if a neighboring router is still available to exchange routing information. The router sends these messages at regular intervals. (Sixty seconds is the default for Cisco routers.) The keepalive message does not contain routing data; it contains only a message header.
- Notification messages—When a router detects an error, it sends a notification message.

Sending MPLS Labels with Routes

When BGP (eBGP and iBGP) distributes a route, it can also distribute an MPLS label that is mapped to that route. The MPLS label mapping information for the route is carried in the BGP update message that contains the information about the route. If the next hop is not changed, the label is preserved.

When you issue the **show bgp neighbors** *ip-address* command on both BGP routers, the routers advertise to each other that they can then send MPLS labels with the routes. If the routers successfully negotiate their ability to send MPLS labels, the routers add MPLS labels to all outgoing BGP updates.

Carrier Supporting Carrier Support for L3VPN

This section provides conceptual information about MPLS VPN Carrier Supporting Carrier (CSC) functionality and includes the following topics:

- CSC Prerequisites, page MPC-367
- CSC Benefits, page MPC-367
- Configuration Options for the Backbone and Customer Carriers, page MPC-367

Throughout this document, the following terminology is used in the context of CSC:

backbone carrier—Service provider that provides the segment of the backbone network to the other provider. A backbone carrier offers BGP and MPLS VPN services.

customer carrier—Service provider that uses the segment of the backbone network. The customer carrier may be an Internet service provider (ISP) or a BGP/MPLS VPN service provider.

CE router—A customer edge router is part of a customer network and interfaces to a provider edge (PE) router. In this document, the CE router sits on the edge of the customer carrier network.

PE router—A provider edge router is part of a service provider's network connected to a customer edge (CE) router. In this document, the PE router sits on the edge of the backbone carrier network

ASBR—An autonomous system boundary router connects one autonomous system to another.

CSC Prerequisites

- You must be able to configure MPLS VPNs with end-to-end (CE-to-CE router) pings working.
- You must be able to configure Interior Gateway Protocols (IGPs), MPLS Label Distribution Protocol (LDP), and Multiprotocol Border Gateway Protocol (MP-BGP).
- You must ensure that CSC-PE and CSC-CE routers support BGP label distribution.

Note BGP is the only supported label distribution protocol on the link between CE and PE.

CSC Benefits

This section describes the benefits of CSC to the backbone carrier and customer carriers.

Benefits to the Backbone Carrier

- The backbone carrier can accommodate many customer carriers and give them access to its backbone.
- The MPLS VPN carrier supporting carrier feature is scalable.
- The MPLS VPN carrier supporting carrier feature is a flexible solution.

Benefits to the Customer Carriers

- The MPLS VPN carrier supporting carrier feature removes from the customer carrier the burden of configuring, operating, and maintaining its own backbone.
- Customer carriers who use the VPN services provided by the backbone carrier receive the same level of security that Frame Relay or ATM-based VPNs provide.
- Customer carriers can use any link layer technology to connect the CE routers to the PE routers and the PE routers to the P routers.
- The customer carrier can use any addressing scheme and still be supported by a backbone carrier.

Benefits of Implementing MPLS VPN CSC Using BGP

The benefits of using BGP to distribute IPv4 routes and MPLS label routes are:

- BGP takes the place of an IGP and LDP in a VPN forwarding and routing instance (VRF) table.
- BGP is the preferred routing protocol for connecting two ISPs,

Configuration Options for the Backbone and Customer Carriers

To enable CSC, the backbone and customer carriers must be configured accordingly:

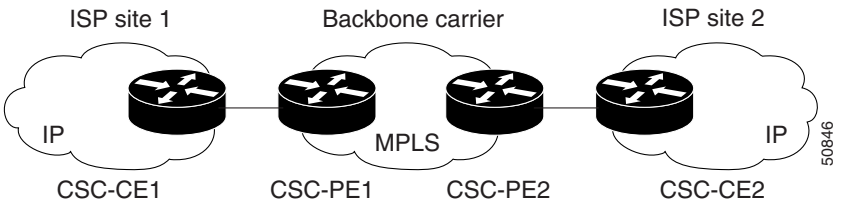
- The backbone carrier must offer BGP and MPLS VPN services.
- The customer carrier can take several networking forms. The customer carrier can be:
  - An ISP with an IP core (see the "Customer Carrier: ISP with IP Core" section on page MPC-368).
  - An MPLS service provider with or without VPN services (see "Customer Carrier: MPLS Service Provider" section on page MPC-368).

Note An IGP in the customer carrier network is used to distribute next hops and loopbacks to the CSC-CE. IBGP with label sessions are used in the customer carrier network to distribute next hops and loopbacks to the CSC-CE.

Customer Carrier: ISP with IP Core

Figure 35 shows a network configuration where the customer carrier is an ISP. The customer carrier has two sites, each of which is a point of presence (POP). The customer carrier connects these sites using a VPN service provided by the backbone carrier. The backbone carrier uses MPLS or IP tunnels to provide VPN services. The ISP sites use IP.

Figure 35 Network: Customer Carrier Is an ISP

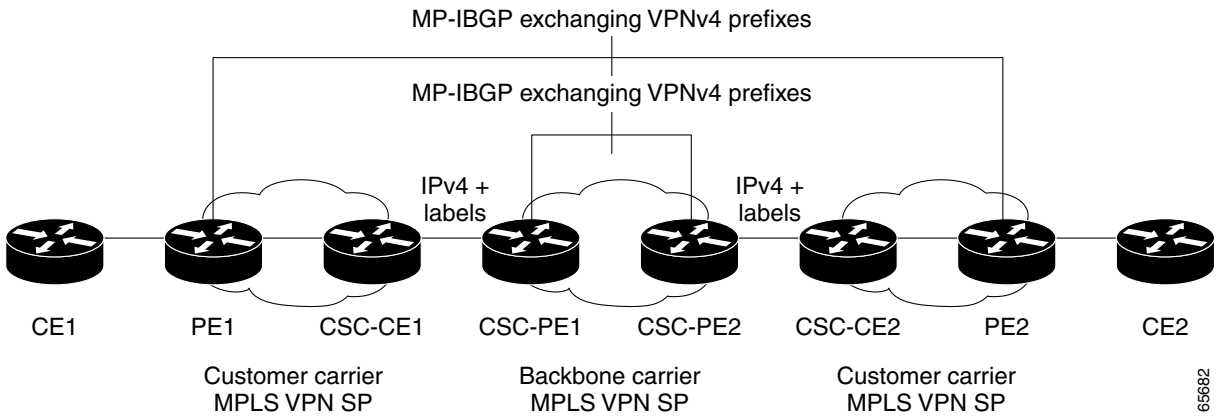


The links between the CE and PE routers use EBGP to distribute IPv4 routes and MPLS labels. Between the links, the PE routers use multiprotocol IBGP to distribute VPNv4 routes.

Customer Carrier: MPLS Service Provider

Figure 36 shows a network configuration where the backbone carrier and the customer carrier are BGP/MPLS VPN service providers. The customer carrier has two sites. The customer carrier uses MPLS in it's network while the backbone carrier may use MPLS or IP tunnels in it's network

Figure 36 Network: Customer Carrier Is an MPLS VPN Service Provider



In this configuration (Figure 36), the customer carrier can configure its network in one of these ways:

- The customer carrier can run an IGP and LDP in its core network. In this case, the CSC-CE1 router in the customer carrier redistributes the EBGP routes it learns from the CSC-PE1 router of the backbone carrier to an IGP.
- The CSC-CE1 router of the customer carrier system can run an IPv4 and labels IBGP session with the PE1 router.

IPv6 VPN Provider Edge (6VPE) Support

6VPE uses the existing MPLS IPv4 core infrastructure for IPv6 transports to enable IPv6 sites to communicate over an MPLS IPv4 core network using MPLS label switch paths (LSPs). 6VPE relies on multiprotocol BGP extensions in the IPv4 network configuration on the provider edge (PE) router to exchange IPv6 reachability information. Edge routers are then configured to be dual stacks running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange (see "Dual Stack" section on page MPC-370).

This section includes the follow subsections:

- 6PVE Benefits, page MPC-369
- 6VPE Network Architecture, page MPC-369
- Dual Stack, page MPC-370
- 6VPE Operation, page MPC-370

6PVE Benefits

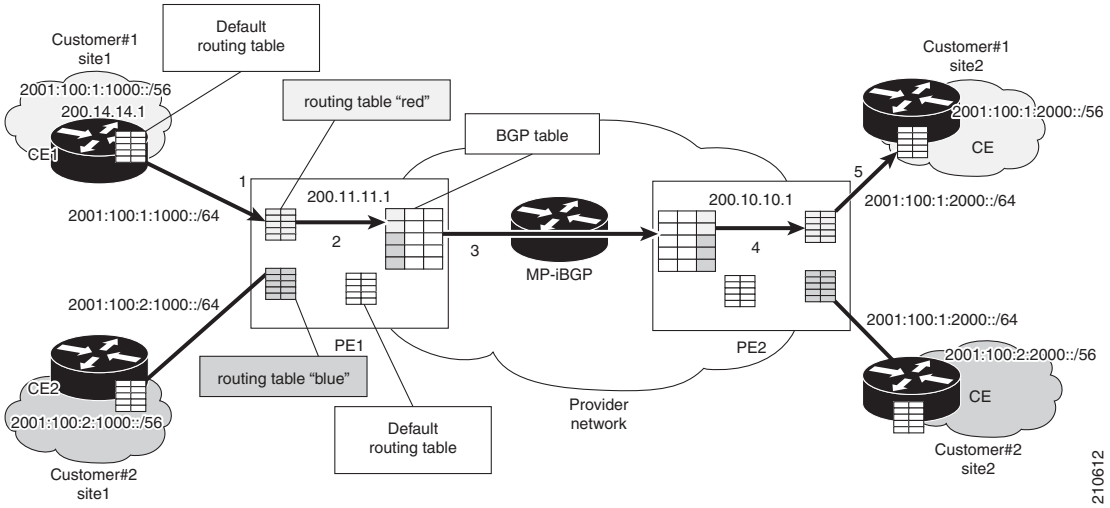
6VPE provides the following benefits to service providers:

- Support for IPv6 without changing the IPv4 MPLS backbone.
- No requirement for a separate signaling plane.
- Leverages operational IPv4 MPLS backbones.
- Cost savings from operating expenses.
- Addresses the security limitations of 6PE.
- Provides logically-separate routing table entries for VPN member devices.
- Provides support for Inter-AS and CSC scenarios. Inter-AS support for 6VPE requires support of Border Gateway Protocol (BGP) to enable the address families and to allocate and distribute the PE and ASBR labels.

6VPE Network Architecture

Figure 37 illustrates the 6VPE network architecture and control plane protocols when two IPv6 sites communicate through an MPLSv4 backbone.

Figure 37 6VPE Network Architecture



Dual Stack

Dual stack is a technique that lets IPv4 and IPv6 coexist on the same interfaces. Coexistence of IPv4 and IPv6 is a requirement for initial deployment. With regard to supporting IPv6 on a MPLS network, two important aspects of the network should be reviewed:

- *Core*: The 6VPE technique carries IPv6 in a VPN fashion over a non-IPv6-aware MPLS core, and enables IPv4 or IPv6 communities to communicate with each other over an IPv4 MPLS backbone without modifying the core infrastructure. By avoiding dual stacking on the core routers, the resources can be dedicated to their primary function to avoid any complexity on the operational side. The transition and integration with respect to the current state of networks is also transparent.
- *Access*: To support native IPv6, the access that connects to IPv4 and IPv6 domains must be IPv6-aware. Service provider edge elements can exchange routing information with end users; therefore, dual stacking is a mandatory requirement on the access layer.

6VPE Operation

When IPv6 is enabled on the subinterface that is participating in a VPN, it becomes an IPv6 VPN. The customer edge-provider edge link is running IPv6 or IPv4 natively. The addition of IPv6 on a provider edge router turns the provider edge into 6VPE, thereby enabling service providers to support IPv6 over the MPLS network.

Provider edge routers use VRF tables to maintain the segregated reachability and forwarding information of each IPv6 VPN. MPBGP with its IPv6 extensions distributes the routes from 6VPE to other 6VPEs through a direct IBGP session or through VPNv6 route reflectors. The next hop of the advertising provider edge router still remains the IPv4 address (normally it is a loopback interface), but with the addition of IPv6, a value of ::FFFF: is prepended to the IPv4 next hop.

Note The Cisco CRS-1 router does not support multiple VRFs on the same physical or logical interface. Only one VRF, which is used for both IPv4 and IPv6 address families, is supported.

The technique can be best described as automatic tunneling of the IPv6 packets through the IPv4 backbone. The MP-BGP relationships remain the same as they are for VPNv4 traffic, with an additional capability of VPNv6. Where both IPv4 and IPv6 are supported, the same set of MPBGP peering relationships is used.

To summarize, from the control plane perspective, the prefixes are signaled across the backbone in the same way as regular MPLS and VPN prefix advertisements. The top label represents the IGP information that remains the same as for IPv4 MPLS. The bottom label represents the VPN information that the packet belongs to. As described earlier, additionally the MPBGP next hop is updated to make it IPv6-compliant. The forwarding or data plane function remains the same as it is deployed for the IPv4 MPLS VPN. The packet forwarding of IPv4 on the current MPLS VPN remains intact.

For detailed information on commands used to configure 6VPE over MPLS, see *Cisco IOS XR MPLS Configuration Guide*.

How to Implement MPLS Layer 3 VPNs on Cisco IOS XR Software

This section contains instructions for the following tasks:

- Configuring the Core Network, page MPC-371 (required)
- Connecting MPLS VPN Customers, page MPC-374 (required)
- Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels, page MPC-394 (optional)
- Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses, page MPC-403 (optional)
- Configuring Carrier Supporting Carrier, page MPC-412(optional)
- Verifying the MPLS Layer 3 VPN Configuration, page MPC-421

Configuring the Core Network

Configuring the core network includes the following tasks:

- Assessing the Needs of MPLS VPN Customers, page MPC-371 (required)
- Configuring Routing Protocols in the Core, page MPC-372 (required)
- Configuring MPLS in the Core, page MPC-372 (required)
- Determining if FIB Is Enabled in the Core, page MPC-372 (required)
- Configuring Multiprotocol BGP on the PE Routers and Route Reflectors, page MPC-373 (required)

Assessing the Needs of MPLS VPN Customers

Before configuring an MPLS VPN, the core network topology must be identified so that it can best serve MPLS VPN customers. Perform this task to identify the core network topology.

SUMMARY STEPS

1. Identify the size of the network.
2. Identify the routing protocols in the core.
3. Determine if MPLS High Availability support is required.
4. Determine if BGP load sharing and redundant paths are required.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Identify the size of the network.	Identify the following to determine the number of routers and ports required: • How many customers will be supported? • How many VPNs are required for each customer? • How many virtual routing and forwarding (VRF) instances are there for each VPN?
Step 2	Identify the routing protocols in the core.	Determine which routing protocols are required in the core network.
Step 3	Determine if MPLS High Availability support is required.	MPLS VPN nonstop forwarding and graceful restart are supported on select routers and Cisco IOS XR software releases.
Step 4	Determine if BGP load sharing and redundant paths are required.	Determine if BGP load sharing and redundant paths in the MPLS VPN core are required.

Configuring Routing Protocols in the Core

To configure a routing protocol, see *Cisco IOS XR Routing Configuration Guide*.

Configuring MPLS in the Core

To enable MPLS on all routers in the core, you must configure a Label Distribution Protocol (LDP). You can use either of the following as an LDP:

- MPLS LDP. See *Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software* for configuration information.
- MPLS Traffic Engineering Resource Reservation Protocol (RSVP). See *Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software* for configuration information.

Determining if FIB Is Enabled in the Core

Forwarding Information Base (FIB) must be enabled on all routers in the core, including the provider edge (PE) routers. For information on how to determine if FIB is enabled, see *Implementing Cisco Express Forwarding on Cisco IOS XR Software* module in *Cisco IOS XR IP Addresses and Services Configuration Guide*.

Configuring Multiprotocol BGP on the PE Routers and Route Reflectors

Perform this task to configure multiprotocol BGP (MP-BGP) connectivity on the PE routers and route reflectors.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family vpnv4 unicast**
   or
   address-family vpnv6 unicast
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family vpnv4 unicast**
   or
   address-family vpnv6 unicast
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 3	**address-family vpnv4 unicast** or **address-family vpnv6 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24 remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

	Command or Action	Purpose
Step 5	**address-family vpnv4 unicast** or **address-family vpnv6 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`	Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Connecting MPLS VPN Customers

To connect MPLS VPN customers to the VPN, perform the following tasks:

- Defining VRFs on the PE Routers to Enable Customer Connectivity, page MPC-375 (required)
- Configuring VRF Interfaces on PE Routers for Each VPN Customer, page MPC-377 (required)
- Configuring BGP as the Routing Protocol Between the PE and CE Routers, page MPC-379 (optional)
- Configuring RIPv2 as the Routing Protocol Between the PE and CE Routers, page MPC-383 (optional)
- Configuring Static Routes Between the PE and CE Routers, page MPC-385 (optional)
- Configuring OSPF as the Routing Protocol Between the PE and CE Routers, page MPC-387 (optional)
- Configuring EIGRP as the Routing Protocol Between the PE and CE Routers, page MPC-390 (optional)
- Configuring EIGRP Redistribution in the MPLS VPN, page MPC-392 (optional)

Defining VRFs on the PE Routers to Enable Customer Connectivity

Perform this task to define VPN routing and forwarding (VRF) instances.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-policy** *policy-name*
5. **import route-target** [*as-number:nn* | *ip-address:nn*]
6. **export route-policy** *policy-name*
7. **export route-target** [*as-number:nn* | *ip-address:nn*]
8. **exit**
9. **exit**
10. **router bgp** *autonomous-system-number*
11. **vrf** *vrf-name*
12. **rd** {*as-number* | *ip-address* | **auto**}
13. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config)# vrf vrf_1`	Configures a VRF instance and enters VRF configuration mode.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Enters VRF address family configuration mode for the IPv4 address family.
Step 4	**import route-policy** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# import route-policy policy_A`	Specifies a route policy that can be imported into the local VPN.

	Command or Action	Purpose
Step 5	**import route-target** [*as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 120:1`	Allows exported VPN routes to be imported into the VPN if one of the route targets of the exported route matches one of the local VPN import route targets.
Step 6	**export route-policy** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# export route-policy policy_B`	Specifies a route policy that can be exported from the local VPN.
Step 7	**export route-target** [*as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 120:2`	Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# exit`	Exits VRF address family configuration mode and returns the router to VRF configuration mode.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-vrf)# exit`	Exits VRF configuration mode and returns the router to global configuration mode.
Step 10	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 11	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1`	Configures a VRF instance and enters VRF configuration mode for BGP routing.

	Command or Action	Purpose
Step 12	**rd** {*as-number* | *ip-address* | **auto**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# rd auto`	Automatically assigns a unique route distinguisher (RD) to vrf_1.

Step 13	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-vrf)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-vrf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring VRF Interfaces on PE Routers for Each VPN Customer

Perform this task to associate a VPN routing and forwarding (VRF) instance with an interface or a subinterface on the PE routers.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **interface** *type instance*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address mask*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config)# interface pos 0/3/0/0`	Enters interface configuration mode.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# vrf vrf_A`	Configures a VRF instance and enters VRF configuration mode.
Step 4	**ipv4 address** *ipv4-address mask* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0`	Configures a primary IPv4 address for the specified interface.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure PE-to-CE routing sessions using BGP.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp router-id** {*ip-address*}
4. **vrf** *vrf-name*
5. **label-allocation-mode** *per-ce*
6. **address-family ipv4 unicast**
7. **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute isis *process-id* [**level** {**1** | **1-inter-area** | **2**}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute ospfv3 *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute static [**metric** *metric-value*] [**route-policy** *route-policy-name*]
8. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
9. **network** {*ip-address/prefix-length* | *ip-address mask*} [**route-policy** *route-policy-name*]
10. **exit**
11. **neighbor** *ip-address*
12. **remote-as** *autonomous-system-number*
13. **password** {**clear** | **encrypted**} *password*
14. **ebgp-multihop** [*ttl-value*]
15. **address-family ipv4 unicast**
16. **allowas-in** [*as-occurrence-number*]
17. **route-policy** *route-policy-name* **in**
18. **route-policy** *route-policy-name* **out**
19. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.
Step 3	**bgp router-id** {*ip-address*} **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24`	Configures the local router with a router id of 192.168.70.24.
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for BGP routing.
Step 5	**label-allocation-mode per-ce** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce`	Sets the MPLS VPN label allocation mode for each customer edge (CE) label mode allowing the provider edge (PE) router to allocate one label for every immediate next-hop.
Step 6	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast`	Enters VRF address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 7	**redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute isis** *process-id* [**level** {**1** | **1-inter-area** | **2**}] [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** route-policy-name] or **redistribute ospfv3** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# redistribute connected`	Causes routes to be redistributed into BGP. The routes that can be redistributed into BGP are: • connected • Intermediate System-to-Intermediate System (IS-IS) • Open Shortest Path First (OSPF) • OSPFv3 • static
Step 8	**aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/8 as-set`	Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized. • The **as-set** keyword generates autonomous system set path information and community information from contributing paths. • The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths. • The **summary-only** keyword filters all more specific routes from updates. • The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.
Step 9	**network** {*ip-address/prefix-length* | *ip-address mask*} [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# network 172.20.0.0/16`	Configures the local router to originate and advertise the specified network.
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit`	Exits VRF address family configuration mode and returns the router to VRF configuration mode for BGP routing.

	Command or Action	Purpose
Step 11	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24`	Places the router in VRF neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.
Step 12	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 13	**password** {**clear** | **encrypted**} *password* **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# password clear pswd123`	Configures neighbor 172.168.40.24 to use MD5 authentication with the password pswd123.
Step 14	**ebgp-multihop** [*ttl-value*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop`	Allows a BGP connection to neighbor 172.168.40.24.
Step 15	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast`	Enters VRF neighbor address family configuration mode for BGP routing.
Step 16	**allowas-in** [*as-occurrence-number*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 3`	Replaces the neighbor autonomous system number (ASN) with the PE ASN in the AS path three times.
Step 17	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in`	Applies the In-Ipv4 policy to inbound IPv4 unicast routes.

	Command or Action	Purpose
Step 18	**route-policy** *route-policy-name* **out** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in`	Applies the In-Ipv4 policy to outbound IPv4 unicast routes.

Step 19	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring RIPv2 as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions using Routing Information Protocol version 2 (RIPv2).

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **vrf** *vrf-name*
4. **interface** *type instance*
5. **site-of-origin** {*as-number*:*number* | *ip-address*:*number*}
6. **exit**
7. **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
   or
   redistribute connected [**route-policy** *name*]
   or
   redistribute isis *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
   or
   redistribute eigrp *as-number* [**route-policy** *name*]
   or

redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**route-policy** *name*]
or
redistribute static [**route-policy** *name*]

8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Enters the Routing Information Protocol (RIP) configuration mode allowing you to configure the RIP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-rip)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for RIP routing.
Step 4	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-rip-vrf)# interface pos 0/3/0/0`	Enters VRF interface configuration mode.
Step 5	**site-of-origin** {*as-number:number* | *ip-address:number*} **Example:** `RP/0/RP0/CPU0:router(config-rip-vrf-if)# site-of-origin 200:1`	Identifies routes that have originated from a site so that the re-advertisement of that prefix back to the source site can be prevented. Uniquely identifies the site from which a PE router has learned a route.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-rip-vrf-if)# exit`	Exits VRF interface configuration mode, and returns the router to VRF configuration mode for RIP routing.

	Command or Action	Purpose
Step 7	**redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*] or **redistribute connected** [**route-policy** *name*] or **redistribute eigrp** *as-number* [**route-policy** *name*] or **redistribute isis** *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*] or **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**route-policy** *name*] or **redistribute static** [**route-policy** *name*] **Example:** `RP/0/RP0/CPU0:router(config-rip-vrf)# redistribute connected`	Causes routes to be redistributed into RIP. The routes that can be redistributed into RIP are: • Border Gateway Protocol (BGP) • connected • Enhanced Interior Gateway Routing Protocol (EIGRP) • Open Shortest Path First (OSPF) • static

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-rip-vrf)# end`
	or
	`RP/0/RP0/CPU0:router(config-rip-vrf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Static Routes Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use static routes.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf** *vrf-name*
4. **address-family ipv4 unicast**
5. *prefix/mask* [**vrf** *vrf-name*] {*ip-address* | *interface-type interface-instance*}
6. *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router static** **Example:** `RP/0/RP0/CPU0:router(config)# router static`	Enters static routing configuration mode allowing you to configure the static routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-static)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for static routing.

Step 4	**address-family ipv4 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-static-vrf)#`
	`address-family ipv4 unicast`
		Enters VRF address family configuration mode for the IPv4 address family.

Step 5	*prefix/mask* [**vrf** *vrf-name*] {*ip-address* |
	interface-type interface-instance}
	
	Example:
	`RP/0/RP0/CPU0:router(config-static-vrf-afi)#`
	`172.168.40.24/24 vrf vrf_1 10.1.1.1`
		Assigns the static route to vrf_1.

	Command or Action	Purpose
Step 6	*prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 bfd fast-detect`	Enables bidirectional forwarding detection (BFD) to detect failures in the path between adjacent forwarding engines. This option is available is when the forwarding router address is specified in Step 5.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-static-vrf-afi)# end`
	or
	`RP/0/RP0/CPU0:router(config-static-vrf-afi)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring OSPF as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Open Shortest Path First (OSPF).

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **router-id** {*router-id* | *interface-type interface-instance*}
5. **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   or
   redistribute connected [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   or
   redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   or
   redistribute static [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

or
redistribute eigrp *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
or
redistribute rip [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

6. **area** *area-id*
7. **interface** *type instance*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 109`	Enters OSPF configuration mode allowing you to configure the OSPF routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.
Step 4	**router-id** {*router-id* | *interface-type interface-instance*} **Example:** `RP/0/RP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10`	Configures the router ID for the OSPF routing process.

	Command or Action	Purpose
Step 5	**redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute eigrp** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}][**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] **Example:** `RP/0/RP0/CPU0:router(config-ospf-vrf)# redistribute connected`	Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are: • Border Gateway Protocol (BGP) • connected • Enhanced Interior Gateway Routing Protocol (EIGRP) • OSPF • static • Routing Information Protocol (RIP)
Step 6	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0`	Configures the OSPF area as area 0.

	Command or Action	Purpose
Step 7	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-vrf-ar)# interface pos 0/3/0/0`	Associates interface POS 0/3/0/0 with area 0.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-vrf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-vrf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring EIGRP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Enhanced Interior Gateway Routing Protocol (EIGRP).

Using EIGRP between the PE and CE routers allows you to transparently connect EIGRP customer networks through an MPLS-enable Border Gateway Protocol (BGP) core network so that EIGRP routes are redistributed through the VPN across the BGP network as internal BGP (iBGP) routes.

Prerequisites

BGP must configured in the network. See *Implementing BGP on Cisco IOS XR Software* module in *Cisco IOS XR Routing Configuration Guide*.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*

3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **router-id** *router-id*
6. **autonomous-system** *as-number*
7. **default-metric** *bandwidth delay reliability loading mtu*
8. **redistribute** {{**bgp** | **connected** | **isis** | **ospf**| **rip** | **static**} [*as-number* | *instance-name*]} [**route-policy** *name*]
9. **interface** *type instance*
10. **site-of-origin** {*as-number:number* | *ip-address:number*}
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router eigrp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router eigrp 24`	Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for EIGRP routing.
Step 4	**address-family ipv4** **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf)# address family ipv4`	Enters VRF address family configuration mode for the IPv4 address family.
Step 5	**router-id** *router-id* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# router-id 172.20.0.0`	Configures the router ID for the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process.
Step 6	**autonomous-system** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 6`	Configures the EIGRP routing process to run within a VRF.

	Command or Action	Purpose
Step 7	**default-metric** *bandwidth delay reliability loading mtu* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# default-metric 100000 4000 200 45 4470`	Sets the metrics for an EIGRP.
Step 8	**redistribute** {{**bgp** | **connected** | **isis** | **ospf**| **rip** | **static**} [*as-number* | *instance-name*]} [**route-policy** *name*] **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute connected`	Causes connected routes to be redistributed into EIGRP.
Step 9	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# interface pos 0/3/0/0`	Associates interface POS 0/3/0/0 with the EIGRP routing process.
Step 10	**site-of-origin** {*as-number:number* | *ip-address:number*} **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 201:1`	Configures site of origin (SoO) on interface POS 0/3/0/0.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring EIGRP Redistribution in the MPLS VPN

Perform this task for every provider edge (PE) router that provides VPN services to enable Enhanced Interior Gateway Routing Protocol (EIGRP) redistribution in the MPLS VPN.

Prerequisites

The metric can be configured in the route-policy configuring using the **redistribute** command (or configured with the **default-metric** command). If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not installed in the EIGRP database. If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not advertised to the CE router. See *Implementing EIGRP on Cisco IOS XR Software* module in *Cisco IOS XR Routing Configuration Guide*.

Restrictions

Redistribution between native EIGRP VPN routing and forwarding (VRF) instances is not supported. This behavior is designed.

SUMMARY STEPS

1. **configure**
2. **router eigrp** as-number
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router eigrp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router eigrp 24`	Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1`	Configures a VRF instance and enters VRF configuration mode for EIGRP routing.
Step 4	**address-family ipv4** **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf)# address family ipv4`	Enters VRF address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 5	**redistribute bgp** [*as-number*] [**route-policy** *policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 24 route-policy policy_A`	Causes Border Gateway Protocol (BGP) routes to be redistributed into EIGRP.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels

Note
- This section is not applicable to Inter-AS over IP tunnels.
- This task is supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

This section contains instructions for the following tasks:

- Configuring the ASBRs to Exchange IPv4 Routes and MPLS Labels (required)
- Configuring the Route Reflectors to Exchange VPN-IPv4 Routes (required)
- Configuring the Route Reflector to Reflect Remote Routes in Its AS (required)

Configuring the ASBRs to Exchange IPv4 Routes and MPLS Labels

Note

- This task is not applicable to Inter-AS over IP tunnels.
- This task is supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

Perform this task to configure the autonomous system boundary routers (ASBRs) to exchange IPv4 routes and MPLS labels.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast**}
4. **allocate-label** {**all**}
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **address-family** {**ipv4 labeled-unicast**}
8. **route-policy** *route-policy-name* {**in**}
9. **route-policy** *route-policy-name* {**out**}
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
	`RP/0/RP0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**ipv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)#`
	`address-family ipv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-af)#`
		Enters global address family configuration mode for the IPv4 unicast address family.

	Command or Action	Purpose
Step 4	**allocate-label** {**all**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all`	Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 6	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

Step 7	**address-family** {**ipv4 labeled-unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)`
		Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 8	**route-policy** *route-policy-name* {**in**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`	Applies a routing policy to updates that are received from a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **in** keyword to define the policy for inbound routes.

	Command or Action	Purpose
Step 9	**route-policy** *route-policy-name* {**out**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies a routing policy to updates that are sent to a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **out** keyword to define the policy for outbound routes.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Route Reflectors to Exchange VPN-IPv4 Routes

Perform this task to enable the route reflectors to exchange VPN-IPv4 routes by using multihop. This task specifies that the next-hop information and the VPN label are to be preserved across the autonomous system.

Note
- This task is not applicable to Inter-AS over IP tunnels.
- This task is supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **ebgp-multihop** [*ttl-value*]

6. **update-source** *interface-type interface-number*
7. **address-family** {**vpnv4 unicast**}
8. **route-policy** *route-policy-name* {**in**}
9. **route-policy** *route-policy-name* {**out**}
10. **next-hop-unchanged**
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
	`RP/0/RP0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 5	**ebgp-multihop** [*ttl-value*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# ebgp-multihop`	Enables multihop peerings with external BGP neighbors.
Step 6	**update-source** *interface-type interface-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0`	Allows BGP sessions to use the primary IP address from a particular interface as the local address.

	Command or Action	Purpose

Step 7	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 8	**route-policy** *route-policy-name* {**in**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`	Applies a routing policy to updates that are received from a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **in** keyword to define the policy for inbound routes.
Step 9	**route-policy** *route-policy-name* {**out**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies a routing policy to updates that are sent to a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **out** keyword to define the policy for outbound routes.
Step 10	**next-hop-unchanged** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-unchanged`	Disables overwriting of the next hop before advertising to external Border Gateway Protocol (eBGP) peers.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Route Reflector to Reflect Remote Routes in Its AS

Perform this task to enable the route reflector (RR) to reflect the IPv4 routes and labels learned by the autonomous system boundary router (ASBR) to the provider edge (PE) routers in the autonomous system. This task is accomplished by making the ASBR and PE route reflector clients of the RR.

Note

- This task is not applicable to Inter-AS over IP tunnels.
- This task is supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast**}
4. **allocate-label** {**all**}
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **update-source** *interface-type interface-number*
8. **address-family** {**ipv4 labeled-unicast**}
9. **route-reflector-client**
10. **neighbor** *ip-address*
11. **remote-as** *autonomous-system-number*
12. **update-source** *interface-type interface-number*
13. **address-family** {**ipv4 labeled-unicast**}
14. **route-reflector-client**
15. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

	Command or Action	Purpose

Step 3	**address-family** {**ipv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-af)#`
		Enters global address family configuration mode for the IPv4 unicast address family.

Step 4	**allocate-label** {**all**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all`	Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 7	**update-source** *interface-type interface-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0`	Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 8	**address-family** {**ipv4 labeled-unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 9	**route-reflector-client** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures the router as a BGP route reflector and neighbor 172.168.40.24 as its client.

Step 10	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# neighbor 175.40.25.2`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 175.40.25.2 as an VPNv4 iBGP peer.

	Command or Action	Purpose
Step 11	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 12	**update-source** *interface-type interface-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0`	Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 13	**address-family** {**ipv4 labeled-unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 14	**route-reflector-client** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures the neighbor as a route reflector client.

Step 15	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses

This section contains instructions for the following tasks:

- Configuring the ASBRs to Exchange VPN-IPv4 Addresses, page MPC-403 (required)
- Configuring EBGP Routing to Exchange VPN Routes Between Subautonomous Systems in a Confederation, page MPC-406 (required)
- Configuring MPLS Forwarding for ASBR Confederations, page MPC-409
- Configuring a Static Route to an ASBR Confederation Peer, page MPC-411

Note These procedures are supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

Configuring the ASBRs to Exchange VPN-IPv4 Addresses

Perform this task to configure an external Border Gateway Protocol (eBGP) autonomous system boundary router (ASBR) to exchange VPN-IPv4 routes with another autonomous system.

Note This procedure is supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**vpnv4 unicast**}
4. **neighbor** *ip-address*
5. **remote-as** *autonomous-system-number*
6. **address-family** {**vpnv4 unicast**}
7. **route-policy** *route-policy-name* {**in**}
8. **route-policy** *route-policy-name* {**out**}
9. **neighbor** *ip-address*
10. **remote-as** *autonomous-system-number*
11. **update-source** *interface-type interface-number*
12. **address-family** {**vpnv4 unicast**}
13. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
	`RP/0/RP0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)#`
	`address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

Step 4	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 5	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

Step 6	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
	`address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 7	**route-policy** *route-policy-name* {**in**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
	`route-policy pass-all in`
		Applies a routing policy to updates that are received from a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **in** keyword to define the policy for inbound routes.

	Command or Action	Purpose
Step 8	**route-policy** *route-policy-name* {**out**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies a routing policy to updates that are sent from a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **out** keyword to define the policy for outbound routes.

Step 9	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# neighbor 175.40.25.2`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 175.40.25.2 as an VPNv4 iBGP peer.

Step 10	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 11	**update-source** *interface-type interface-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0`	Allows BGP sessions to use the primary IP address from a particular interface as the local address.

	Command or Action	Purpose

Step 12	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 13	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring EBGP Routing to Exchange VPN Routes Between Subautonomous Systems in a Confederation

Perform this task to configure external Border Gateway Protocol (eBGP) routing to exchange VPN routes between subautonomous systems in a confederation.

Note This procedure is supported on the Cisco CRS-1 and Cisco XR 12000 Series Router.

Note To ensure that host routes for VPN-IPv4 eBGP neighbors are propagated (by means of the Interior Gateway Protocol [IGP]) to other routers and PE routers, specify the **redistribute connected** command in the IGP configuration portion of the confederation eBGP (CEBGP) router. If you are using Open Shortest Path First (OSPF), make sure that the OSPF process is not enabled on the CEBGP interface in which the "redistribute connected" subnet exists.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp confederation peers** *peer autonomous-system-number*

4. **bgp confederation identifier** *autonomous-system-number*
5. **address-family** {**vpnv4 unicast**}
6. **neighbor** *ip-address*
7. **remote-as** *autonomous-system-number*
8. **address-family** {**vpnv4 unicast**}
9. **route-policy** *route-policy-name* **in**
10. **route-policy** *route-policy-name* **out**
11. **next-hop-self**
12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
	`RP/0/RP0/CPU0:router(config-bgp)#`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**bgp confederation peers** *peer autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 8`	Configures the peer autonomous system number that belongs to the confederation.
Step 4	**bgp confederation identifier** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5`	Specifies the autonomous system number for the confederation ID.

Step 5	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

	Command or Action	Purpose

Step 6	**neighbor** *ip-address*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 7	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

Step 8	**address-family** {**vpnv4 unicast**}
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 9	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in`	Applies a routing policy to updates received from a BGP neighbor.
Step 10	**route-policy** *route-policy-name* **out** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy Out-Ipv4 out`	Applies a routing policy to updates advertised to a BGP neighbor.

	Command or Action	Purpose
Step 11	**next-hop-self** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	Disables next-hop calculation and let you insert your own address in the next-hop field of BGP updates.

Step 12	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring MPLS Forwarding for ASBR Confederations

Perform this task to configure MPLS forwarding for autonomous system boundary router (ASBR) confederations (in BGP) on a specified interface.

Note This configuration adds the implicit NULL rewrite corresponding to the peer associated with the interface, which is required to prevent BGP from automatically installing rewrites by LDP (in multihop instances).

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **mpls activate**
4. **interface** *type interface-id*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *as-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
	`RP/0/RP0/CPU0:router(config-bgp)`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**mpls activate**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# mpls activate`
	`RP/0/RP0/CPU0:router(config-bgp-mpls)#`
		Enters BGP MPLS activate configuration mode.

Step 4	**interface** *type interface-id* **Example:** `RP/0/RP0/CPU0:router(config-bgp-mpls)# interface pos 0/3/0/0`	Enables MPLS on the interface.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-mpls)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-mpls)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Static Route to an ASBR Confederation Peer

Perform this task to configure a static route to an Inter-AS confederation peer. For more detailed information, see "Configuring a Static Route to a Peer" section on page MPC-419

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router static**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router static`
	`RP/0/RP0/CPU0:router(config-static)#`
		Enters router static configuration mode.

Step 3	**address-family ipv4 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-static)#`
	`address-family ipv4 unicast`
	`RP/0/RP0/CPU0:router(config-static-afi)#`
		Enables an IPv4 address family.

	Command or Action	Purpose
Step 4	***A.B.C.D/length*** *next-hop* **Example:** `RP/0/RP0/CPU0:router(config-static-afi)# 10.10.10.10/32 9.9.9.9`	Enters the address of the destination router (including IPv4 subnet mask).

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-static-afi)# end`
	or
	`RP/0/RP0/CPU0:router(config-static-afi)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Carrier Supporting Carrier

Perform the tasks in this section to configure CSC:

- Identifying the Carrier Supporting Carrier Topology, page MPC-412 (required)
- Configuring the Backbone Carrier Core, page MPC-413 (required)
- Configuring the CSC-PE and CSC-CE Routers, page MPC-413 (required)
- Configuring a Static Route to a Peer, page MPC-419 (required)

Identifying the Carrier Supporting Carrier Topology

Before you configure the MPLS VPN CSC with BGP, you must identify both the backbone and customer carrier topology.

Note You can connect multiple CSC-CE routers to the same PE, or you can connect a single CSC-CE router to multiple CSC-PEs using more than one CSC-CE interface to provide redundancy and multiple path support in a CSC topology.

Perform this task to identify the carrier supporting carrier topology.

SUMMARY STEPS

1. Identify the type of customer carrier, ISP, or MPLS VPN service provider**.**
2. Identify the CE routers.
3. Identify the customer carrier core router configuration.
4. Identify the customer carrier edge (CSC-CE) routers.
5. Identify the backbone carrier router configuration.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Identify the type of customer carrier, ISP, or MPLS VPN service provider.	Sets up requirements for configuration of carrier supporting carrier network.
Step 2	Identify the CE routers.	Sets up requirements for configuration of CE to PE connections.
Step 3	Identify the customer carrier core router configuration.	Sets up requirements for configuration between core (P) routers and between P routers and edge routers (PE and CSC-CE routers).
Step 4	Identify the customer carrier edge (CSC-CE) routers.	Sets up requirements for configuration of CSC-CE to CSC-PE connections.
Step 5	Identify the backbone carrier router configuration.	Sets up requirements for configuration between CSC core routers and between CSC core routers and edge routers (CSC-CE and CSC-PE routers).

Configuring the Backbone Carrier Core

Configuring the backbone carrier core requires setting up connectivity and routing functions for the CSC core and the CSC-PE routers. To do so, you must complete the following high-level tasks:

- Verify IP connectivity in the CSC core.
- Verify LDP configuration in the CSC core.

Note This task is not applicable to CSC over IP tunnels.

- Configure VRFs for CSC-PE routers.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

Configuring the CSC-PE and CSC-CE Routers

Perform the following tasks to configure links between a CSC-PE router and the carrier CSC-CE router for an MPLS VPN CSC network that uses BGP to distribute routes and MPLS labels:

- Configuring a CSC-PE (required)
- Configuring a CSC-CE (required)

Figure 38 shows the configuration for the peering with directly connected interfaces between CSC-PE and CSC-CE routers. This configuration is used as the example in the tasks that follow.

Figure 38 Configuration for Peering with Directly Connected Interfaces Between CSC-PE and CSC-CE Routers

Configuring a CSC-PE

Perform this task to configure a CSC-PE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family vpnv4 unicast**
4. **neighbor** *A.B.C.D*
5. **remote-as** *as-number*
6. **update-source** *interface-type interface-number*
7. **address-family vpnv4 unicast**
8. **vrf** *vrf-name*
9. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}
10. **address-family ipv4 unicast**
11. **allocate-label all**
12. **neighbor** *A.B.C.D*
13. **remote-as** *as-number*
14. **address-family ipv4 labeled-unicast**
15. **route-policy** *route-policy-name* **in**
16. **route-policy** *route-policy-name* **out**
17. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *as-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 2`
	`RP/0/RP0/CPU0:router(config-bgp)#`
		Configures a BGP routing process and enters router configuration mode.
		• Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3	**address-family vpnv4 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)#`
	`address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

Step 4	**neighbor** *A.B.C.D*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 10.10.10.0`
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
		Configures the IP address for the BGP neighbor.

Step 5	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 888`	Configures the AS number for the BGP neighbor.

Step 6	**update-source** *interface-type interface-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
	`update-source loopback0`
		Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 7	**address-family vpnv4 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)#`
	`address-family vpnv4 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 unicast address family.

Step 8	**vrf** *vrf-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# vrf 9999`
	`RP/0/RP0/CPU0:router(config-bgp-vrf)#`
		Configures a VRF instance.

	Command or Action	Purpose
Step 9	**rd** {*as-number:nn* | *ip-address:nn* | **auto**} **Example:** `RP/0/RP0/CPU0:router(onfig-bgp-vrf)# rd auto`	Configures a route distinguisher. **Note** Use the **auto** keyword to automatically assign a unique route distinguisher.
Step 10	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast RP/0/RP0/CPU0:router(config-bgp-vrf-af)#`	Configures IPv4 unicast address family.
Step 11	**allocate-label all** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# allocate-label all`	Allocate labels for all local prefixes and prefixes received with labels.
Step 12	**neighbor** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# neighbor 10.10.10.0 RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)#`	Configures the IP address for the BGP neighbor.
Step 13	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 888`	Enables the exchange of information with a neighboring BGP router.
Step 14	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 labeled-unicast RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)#`	Configures IPv4 labeled-unicast address family.
Step 15	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all in`	Applies the pass-all policy to all inbound routes.

	Command or Action	Purpose
Step 16	**route-policy** *route-policy-name* **out** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out`	Applies the pass-all policy to all outbound routes.

Step 17	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(cconfig-bgp-vrf-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a CSC-CE

Perform this task to configure a CSC-CE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv4 unicast**
4. **redistribute ospf** *instance-number*
5. **allocate-label route-policy** *route-policy-name*
6. **exit**
7. **neighbor** *A.B.C.D*
8. **remote-as** *as-number*
9. **address-family ipv4 labeled-unicast**
10. **route-policy** *route-policy-name* **in**
11. **route-policy** *route-policy-name* **out**
12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 1`	Configures a BGP routing process and enters router configuration mode. • Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures IPv4 unicast address-family.
Step 4	**redistribute ospf** *instance-number* **Example:** `RP/0/RP0/CPU0:router(config-router-af)# redistribute ospf 1`	Redistributes OSPF routes into BGP.
Step 5	**allocate-label route-policy** *route-policy-name* **Example:** `RP/0/RP0/CPU0:router(config-router-af)# allocate-label route-policy internal-routes`	Allocates labels for those routes that match the route policy. These labeled routes are advertised to neighbors configured with address-family ipv4 labeled-unicast.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 7	**neighbor** *A.B.C.D* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 51.0.0.1`	Configures the IP address for the BGP neighbor.
Step 8	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# remote-as 1`	Enables the exchange of information with a neighboring BGP router.
Step 9	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp)#`	Configures IPv4 labeled-unicast address family.

	Command or Action	Purpose
Step 10	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RP0/CPU0:router(config-bgp)#`	Applies the route-policy to all inbound routes.
Step 11	**route-policy** *route-policy-name* **out** **Example:** `RP/0/RP0/CPU0:router(config-bgp)#`	Applies the route-policy to all outbound routes.

Step 12	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Static Route to a Peer

Perform this task to configure a static route to an Inter-AS or CSC-CE peer.

When you configure an Inter-AS or CSC peer, BGP allocates a label for a /32 route to that peer and performs a NULL label rewrite. When forwarding a labeled packet to the peer, the router removes the top label from the label stack; however, in such an instance, BGP expects a /32 route to the peer. This task ensures that there is, in fact, a /32 route to the peer.

Please be aware of the following facts before performing this task:

- A /32 route is not required to establish BGP peering. A route using a shorter prefix length will also work.
- A shorter prefix length route is not associated with the allocated label; even though the BGP session comes up between the peers, without the static route, forwarding will not work.

Note To configure a static route on a CSC-PE, you must configure the router under the VRF (as noted in the detailed steps).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router(config)# configure`	Enters global configuration mode.
Step 2	**router static** **Example:** `RP/0/RP0/CPU0:router(config)# router static`	Enters router static configuration mode.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast`	Enables an IPv4 address family. **Note** To configure a static route on a CSC-PE, you must first configure the VRF using the **vrf** command before **address-family**.

	Command or Action	Purpose
Step 4	***A.B.C.D/length*** *next-hop* **Example:** `RP/0/RP0/CPU0:router(config-static-afi)# 10.10.10.10/32 9.9.9.9`	Enters the address of the destination router (including IPv4 subnet mask).

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-static-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-static-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying the MPLS Layer 3 VPN Configuration

Perform this task to verify the MPLS Layer 3 VPN configuration.

SUMMARY STEPS

1. **show running-config router bgp** *as-number* **vrf** *vrf-name*
2. **show running-config routes**
3. **show ospf vrf** *vrf-name* **database**
4. **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*
5. **show bgp vrf** *vrf-name* **summary**
6. **show bgp vrf** *vrf-name* **neighbors** *ip-address*
7. **show bgp vrf** *vrf-name*
8. **show route vrf** *vrf-name ip-address*
9. **show bgp vpn unicast summary**
10. **show running-config router isis**
11. **show running-config mpls**
12. **show isis adjacency**

13. **show mpls ldp forwarding**
14. **show bgp vpnv4 unicast**
    or
    show bgp vpnv6 unicast
15. **show bgp vrf** *vrf-name*
16. **show bgp vrf** *vrf-name* **imported-routes**
17. **show route vrf** *vrf-name ip-address*
18. **show cef vrf** *vrf-name ip-address*
19. **show cef vrf** *vrf-name ip-address* **location** *node-id*
20. **show bgp vrf** *vrf-name ip-address*
21. **show ospf vrf** *vrf-name* **database**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show running-config router bgp** *as-number* **vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A`	Displays the specified VPN routing and forwarding (VRF) content of the currently running configuration.
Step 2	**show running-config routes** **Example:** `RP/0/RP0/CPU0:router# show running-config routes`	Displays the Open Shortest Path First (OSPF) routes table in the currently running configuration.
Step 3	**show ospf vrf** *vrf-name* **database** **Example:** `RP/0/RP0/CPU0:router# show ospf vrf vrf_A database`	Displays lists of information related to the OSPF database for a specified VRF.
Step 4	**show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A neighbor 172.168.40.24`	Displays the Border Gateway Protocol (BGP) VRF neighbor content of the currently running configuration.
Step 5	**show bgp vrf** *vrf-name* **summary** **Example:** `RP/0/RP0/CPU0:router# show bgp vrf vrf_A summary`	Displays the status of the specified BGP VRF connections.

	Command or Action	Purpose
Step 6	**show bgp vrf** *vrf-name* **neighbors** *ip-address* **Example:** `RP/0/RP0/CPU0:router# show bgp vrf vrf_A neighbors 172.168.40.24`	Displays information about BGP VRF connections to the specified neighbors.
Step 7	**show bgp vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router# show bgp vrf vrf_A`	Displays information about a specified BGP VRF.
Step 8	**show route vrf** *vrf-name ip-address* **Example:** `RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0`	Displays the current routes in the Routing Information Base (RIB) for a specified VRF.
Step 9	**show bgp vpn unicast summary** **Example:** `RP/0/RP0/CPU0:router# show bgp vpn unicast summary`	Displays the status of all BGP VPN unicast connections.
Step 10	**show running-config router isis** **Example:** `RP/0/RP0/CPU0:router# show running-config router isis`	Displays the Intermediate System-to-Intermediate System (IS-IS) content of the currently running configuration.
Step 11	**show running-config mpls** **Example:** `RP/0/RP0/CPU0:router# show running-config mpls`	Displays the MPLS content of the currently running-configuration.
Step 12	**show isis adjacency** **Example:** `RP/0/RP0/CPU0:router# show isis adjacency`	Displays IS-IS adjacency information.
Step 13	**show mpls ldp forwarding** **Example:** `RP/0/RP0/CPU0:router# show mpls ldp forwarding`	Displays the Label Distribution Protocol (LDP) forwarding state installed in MPLS forwarding.
Step 14	**show bgp vpnv4 unicast** or **show bgp vpnv6 unicast** **Example:** `RP/0/RP0/CPU0:router# show bgp vpnv4 unicast`	Displays entries in the BGP routing table for VPNv4 or VPNv6 unicast addresses.
Step 15	**show bgp vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router# show bgp vrf vrf_A`	Displays entries in the BGP routing table for VRF vrf_A.

	Command or Action	Purpose
Step 16	**show bgp vrf** *vrf-name* **imported-routes** **Example:** `RP/0/RP0/CPU0:router# show bgp vrf vrf_A imported-routes`	Displays BGP information for routes imported into specified VRF instances.
Step 17	**show route vrf** *vrf-name ip-address* **Example:** `RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0`	Displays the current specified VRF routes in the RIB.
Step 18	**show cef vrf** *vrf-name ip-address* **Example:** `RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1`	Displays the IPv4 Cisco Express Forwarding (CEF) table for a specified VRF.
Step 19	**show cef vrf** *vrf-name ip-address* **location** *node-id* **Example:** `RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1 location 0/1/cpu0`	Displays the IPv4 CEF table for a specified VRF and location.
Step 20	**show bgp vrf** *vrf-name ip-address* **Example:** `RP/0/RP0/CPU0:router# show bgp vrf vrf_A 10.0.0.0`	Displays entries in the BGP routing table for VRF vrf_A.
Step 21	**show ospf vrf** *vrf-name* **database** **Example:** `RP/0/RP0/CPU0:router# show ospf vrf vrf_A database`	Displays lists of information related to the OSPF database for a specified VRF.

Configuring 6VPE Support

The following tasks are required to configure 6VPE support:

- Configuring an IPv6 Address Family Under VRF, page MPC-425
- Configuring BGP Route Distinguisher and Core-facing Sessions, page MPC-426
- Configuring a PE-CE Protocol, page MPC-428

Configuring an IPv6 Address Family Under VRF

Perform this task to configure an IPv6 address-family under the VRF for 6VPE support.

Note You can also configure a maximum-routes limit for the VRF, export, and import policies.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf_name*
3. **address-family ipv6 unicast**
4. **import route-target** [*as-number:nn* | *ip-address:nn*]
5. **export route-target** [*as-number:nn* | *ip-address:nn*]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config)# vrf vrf_1`	Configures a VRF instance and enters VRF configuration mode.
Step 3	**address-family ipv6 unicast** **Example:** `RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Enters VRF address family configuration mode for the IPv6 address family.
Step 4	**import route-target** [*as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 120:1`	Configures a VPN routing and forwarding (VRF) import route-target extended community.

	Command or Action	Purpose
Step 5	**export route-target** [*as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 120:2`	Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-vrf-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-vrf-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP Route Distinguisher and Core-facing Sessions

Perform this task to configure VRF route distinguisher values and core-facing neighbors under BGP.

Note Before you perform this task, you must first configure a VRF and map the VRF to an interface. For more information, see *Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software*.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family vpnv6 unicast**
4. **vrf** *vrf-name*
5. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}
6. **address-family ipv6 unicast**
7. **exit**
8. **neighbor** *ip-address* **remote-as** *as-number*

9. **address-family ipv6 unicast**
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *as-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 100`
	`RP/0/RP0/CPU0:router(config-bgp)#`
		Enters router BGP configuration mode.

Step 3	**address-family vpnv6 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)#`
	`address-family vpnv6 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-af)`
		Enters address family configuration mode for the VPNv6 address family.

Step 4	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# vrf red`	Configures a VPN VRF instance and enters VRF configuration mode.
Step 5	**rd** {*as-number:nn* | *ip-address:nn* | **auto**} **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# router bgp 100`	Configures a route distinguisher.

Step 6	**address-family ipv6 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-vrf)#`
	`address-family ipv6 unicast`
		Enters IPv6 address family configuration mode.

Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit`	Exits the current configuration mode.
Step 8	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24 remote-as 2002f`	Creates a neighbor and assigns it a remote autonomous system number.

	Command or Action	Purpose
Step 9	**address-family ipv6 unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast`	Enters IPv6 address family configuration mode.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-vrf-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-vrf-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a PE-CE Protocol

Perform this task to configure a PE-CE protocol for 6VPE.

Note eBGP, iBGP and eiBGP load-balancing configuration options are also supported for 6VPE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv6 unicast**
5. **exit**
6. **exit**
7. **neighbor** *ip-address*
8. **remote-as** *as-number*
9. **address-family vpnv6 unicast**

10. **route-policy** *route-policy-name* **in**
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *as-number*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
	`RP/0/RP0/CPU0:router(config-bgp)`
		Enters router BGP configuration mode.

Step 3	**vrf** *vrf-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# vrf red`
	`RP/0/RP0/CPU0:router(config-bgp-vrf)`
		Configures a VPN VRF instance and enters VRF configuration mode.

Step 4	**address-family ipv6 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-vrf)`
	`address-family ipv6 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-vrf-af)`
		Enters IPv6 address family configuration mode.

Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit`	Exits the current configuration mode.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-vrf)# exit`	Exits the current configuration mode.
Step 7	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 10,10.10,10`	Creates a neighbor and assigns it a remote autonomous system number of 2002.
Step 8	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1000`	Creates a BGP neighbor and begin the exchange of routing information.

	Command or Action	Purpose

Step 9	**address-family vpnv6 unicast**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast`
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)`
		Enters address family configuration mode for the VPNv6 address family.

Step 10	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in`	Applies a routing policy to updates advertised to or received from a BGP neighbor.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing MPLS Layer 3 VPNs

The following section provides sample configurations for MPLS L3VPN features, including:

- Configuring an MPLS VPN Using BGP: Example, page MPC-431
- Configuring the Routing Information Protocol on the PE Router: Example, page MPC-432
- Configuring the PE Router Using EIGRP: Example, page MPC-432
- Configuration Examples for MPLS VPN CSC, page MPC-432
- Configuration Examples for 6VPE, page MPC-434

Configuring an MPLS VPN Using BGP: Example

The following example shows the configuration for an MPLS VPN using BGP on "vrf vpn1":

```
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!
interface Loopback0
  ipv4 address 10.0.0.1 255.255.255.255
!
interface gigabitEthernet 0/1/0/0
  vrf vpn1
  ipv4 address 34.0.0.2 255.0.0.0
!
interface gigabitEthernet 0/1/0/1
  ipv4 address 30.0.0.1 255.0.0.0
!
router ospf 100
  area 100
    interface loopback0
    interface gigabitEthernet 0/1/0/1
  !
!
router bgp 100
  address-family vpnv4 unicast
  neighbor 10.0.0.3
    remote-as 100
    update-source Loopback0
    address-family vpnv4 unicast
  !
  vrf vpn1
    rd 100:1
    address-family ipv4 unicast
      redistribute connected
    !
    neighbor 34.0.0.1
      remote-as 200
      address-family ipv4 unicast
        as-override
        route-policy pass-all in
        route-policy pass-all out
      !
      advertisement-interval 5
    !
  !
!
mpls ldp
  route-id looback0
  interface gigabitEthernet 0/1/0/1
!
```

Configuring the Routing Information Protocol on the PE Router: Example

The following example shows the configuration for the RIP on the PE router:

```
vrf vpn1
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!

interface gigabitEthernet 0/1/0/0
  vrf vpn1
  ipv4 address 34.0.0.2 255.0.0.0
!

router rip
 vrf vpn1
  interface GigabitEthernet0/1/0/0
  !
  timers basic 30 90 90 120
  redistribute bgp 100
  default-metric 3
  route-policy pass-all in
 !
```

Configuring the PE Router Using EIGRP: Example

The following example shows the configuration for the Enhanced Interior Gateway Routing Protocol (EIGRP) on the PE router:

```
Router eigrp 10
 vrf VRF1
  address-family ipv4
   router-id 40.1.1.2
   default-metric 100000 2000 255 1 1500
   as 62
   redistribute bgp 2000
   interface Loopback0
   !
   interface GigabitEthernet0/6/0/0
```

Configuration Examples for MPLS VPN CSC

Configuration examples for the MPLS VPN CSC include:

- Configuring the Backbone Carrier Core: Examples, page MPC-433
- Configuring the Links Between CSC-PE and CSC-CE Routers: Examples, page MPC-433
- Configuring a Static Route to a Peer: Example, page MPC-434

Configuring the Backbone Carrier Core: Examples

Configuration examples for the backbone carrier core included in this section are as follows:

- Configuring VRFs for CSC-PE Routers: Example, page MPC-433
- Configuring the Links Between CSC-PE and CSC-CE Routers: Examples, page MPC-433

Configuring VRFs for CSC-PE Routers: Example

The following example shows how to configure a VPN routing and forwarding instance (VRF) for a CSC-PE router:

```
config
  vrf vpn1
   address-family ipv4 unicast
    import route-target 100:1
    export route-target 100:1
   end
```

Configuring the Links Between CSC-PE and CSC-CE Routers: Examples

This section contains the following examples:

- Configuring a CSC-PE: Example, page MPC-433
- Configuring a CSC-CE: Example, page MPC-433

Configuring a CSC-PE: Example

In this example, a CSC-PE router peers with a PE router, 60.0.0.2, in its own AS. It also has a labeled unicast peering with a CSC-CE router, 52.0.0.1.

```
config
    router bgp 2
        address-family vpnv4 unicast
        neighbor 60.0.0.2
            remote-as 2
            update-source loopback0
            address-family vpnv4 unicast
        vrf customer-carrier
            rd 1:100
            address-family ipv4 unicast
                allocate-label all
                redistribute static
        neighbor 52.0.0.1
            remote-as 1
            address-family ipv4 labeled-unicast
                route-policy pass-all in
                route-policy pass-all out
                as-override
end
```

Configuring a CSC-CE: Example

The following example shows how to configure a CSC-CE router. In this example, the CSC-CE router peers CSC-PE router 52.0.0.2 in AS 2.

```
config
    router bgp 1
        address-family ipv4 unicast
```

```
            redistribute ospf 200
            allocate-label all
        neighbor 52.0.0.2
            remote-as 2
            address-family ipv4 labeled-unicast
            route-policy pass-all in
            route-policy pass-all out
end
```

Configuring a Static Route to a Peer: Example

The following example show how to configure a static route to an Inter-AS or CSC-CE peer:

```
config
 router static
  address-family ipv4 unicast
  50.0.0.2/32 40.1.1.1
end
```

Configuration Examples for 6VPE

Configuration examples for the MPLS VPN CSC include:

- Configuring an IPv6 Address Family Under VRF: Example, page MPC-434
- Configuring BGP for the Address Family VPNv6: Example, page MPC-434
- Configuring a PE-CE Protocol: Example, page MPC-435
- Configuring an Entire 6VPE Configuration: Example, page MPC-435

Configuring an IPv6 Address Family Under VRF: Example

The following example shows a standard configuration of an IPv6 address family under VRF:

```
configure
 vrf red
  address-family ipv6 unicast
   import route-target
    500:1
   !
   export route-target
    500:1
   !
  !
```

Configuring BGP for the Address Family VPNv6: Example

The following example shows the configuration for the address family VPNv6 under the PE peer:

```
configure
 router bgp 3
  address-family vpnv6 unicast
  !
   neighbor 192.168.254.3
   remote-as 3
   update-source Loopback0
   address-family ipv4 unicast
   !
   address-family vpnv44 unicast
```

```
   !
   address-family ipv6 labeled-unicast
   !
   address-family vpnv6 unicast
   !
  !
```

Configuring the Address Family IPv6 for the VRF Configuration Under BGP: Example

The following example shows the configuration for the address family IPv6 for the VRF configuration under BGP:

```
!
 vrf red
  address-family ipv6 unicast
  redistribute connected
  !
```

Configuring a PE-CE Protocol: Example

The following example shows the eBGP configuration of a PE-CE protocol:

```
  !
  neighbor 2001:db80:cafe:1::2
   remote-as 100
   address-family ipv6 unicast
   route-policy pass in
   route-policy pass out
```

Configuring an Entire 6VPE Configuration: Example

Two VPNs, which are named red and blue, are created across router2 and router4. The VRF red is for the user running IPv6 addressing in the network. The VRF blue is for the user running IPv4 addressing. 6VPE is implemented to carry the VPNv6 prefixes across to the other PE.

The following example shows the entire 6VPE configuration that includes the interface and VRF configurations of both PE routers across the route reflectors:

```
router2 (PE router)
interface GigabitEthernet0/0/1/3.1
 vrf red
 ipv4 address 192.3.1.1 255.255.255.0
 ipv6 address 2001:db80:cafe:1::1/64
 dot1q vlan 2
!

show run interface gigabitEthernet 0/0/1/3.2
interface GigabitEthernet0/0/1/3.2
 vrf blue
 ipv4 address 192.3.2.1 255.255.255.0
 dot1q vlan 3
!


vrf red
 address-family ipv4 unicast
  import route-target
   500:1
```

```
  !
  export route-target
   500:1
  !
 !
 address-family ipv6 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
!
vrf blue
 address-family ipv4 unicast
  import route-target
   600:1
  !
  export route-target
   600:1
  !
 !


router bgp 3
 address-family ipv4 unicast
  network 3.3.3.3/32
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
  network 2001:db82:cafe:1::/64
  allocate-label all
 !
 address-family vpnv6 unicast
 !
 neighbor 192.168.253.4
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 vrf red
  rd 500:1
  address-family ipv4 unicast
```

```
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
  neighbor 2001:db80:cafe:1::2
   remote-as 100
   address-family ipv6 unicast
    route-policy pass in
    route-policy pass out
   !
  !
 !
 vrf blue
  rd 600:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
!


router3 (RR)

router bgp 3
 bgp router-id 192.168.253.4
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
 !
 address-family vpnv6 unicast
 !
 neighbor-group all
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
   route-reflector-client
  !
  address-family vpnv4 unicast
   route-reflector-client
  !
  address-family ipv6 labeled-unicast
   route-reflector-client
  !
  address-family vpnv6 unicast
   route-reflector-client
  !
 !
 neighbor 192.168.253.1
  use neighbor-group all
 !
 neighbor 192.168.253.2
  use neighbor-group all
 !
 neighbor 192.168.253.3
  use neighbor-group all
 !
 neighbor 192.168.253.5
  use neighbor-group all
 !
 neighbor 192.168.253.6
  use neighbor-group all
```

```
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!

router4(PE router)

vrf red
 address-family ipv4 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
 address-family ipv6 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
!
vrf blue
 address-family ipv4 unicast
  import route-target
   600:1
  !
  export route-target
   600:1
  !
 !
!


router bgp 3
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
  network 2001:db84:beef:1::/64
  allocate-label all
 !
 address-family vpnv6 unicast
 !
 neighbor 192.168.253.4
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
```

```
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
 !
 vrf red
  rd 500:1
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
 !
 vrf blue
  rd 600:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
!
```

The following example displays the sample output for the entire 6VPE configuration:

```
show route vrf red ipv6

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

C    2001:db80:beef:1::/64 is directly connected,
      19:09:50, GigabitEthernet0/0/1/3.1
L    2001:db80:beef:1::1/128 is directly connected,
      19:09:50, GigabitEthernet0/0/1/3.1
B    2001:db80:cafe:1::/64
      [200/0] via ::ffff:192.168.253.3 (nexthop in vrf default), 07:03:40


show route vrf red ipv6

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

B    2001:db80:beef:1::/64
      [200/0] via ::ffff:192.168.253.6 (nexthop in vrf default), 07:04:14
C    2001:db80:cafe:1::/64 is directly connected,
```

```
      08:28:12, GigabitEthernet0/0/1/3.1
L    2001:db80:cafe:1::1/128 is directly connected,
      08:28:12, GigabitEthernet0/0/1/3.1
```

Additional References

For additional information, refer to the following documents:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software module in Cisco IOS XR MPLS Configuration Guide*
Routing (BGP, EIGRP, OSPF, and RIP) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*
Routing (BGP, EIGRP, OSPF, and RIP) configuration	*Cisco IOS XR Routing Configuration Guide*
MPLS LDP configuration: configuration concepts, task, and examples	*Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software*
MPLS Traffic Engineering Resource Reservation Protocol configuration: configuration concepts, task, and examples	*Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco CRS-1 router getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1700	*Assigned Numbers*
RFC 1918	*Address Allocation for Private Internets*
RFC 1966	*BGP Route Reflectors: An Alternative to Full Mesh iBGP*
RFC 2283	*Multiprotocol Extensions for BGP-4*
RFC 2547	*BGP/MPLS VPNs*
RFC 2842	*Capabilities Advertisement with BGP-4*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 3107	*Carrying Label Information in BGP-4*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

INDEX

HC	Cisco IOS XR Interface and Hardware Component Configuration Guide
IC	Cisco IOS XR IP Addresses and Services Configuration Guide
MCC	Cisco IOS XR Multicast Configuration Guide
MNC	Cisco IOS XR System Monitoring Configuration Guide
MPC	Cisco IOS XR MPLS Configuration Guide
QC	Cisco IOS XR Modular Quality of Service Configuration Guide
RC	Cisco IOS XR Routing Configuration Guide
SBC	Cisco IOS XR Session Border Controller Configuration Guide
SC	Cisco IOS XR System Security Configuration Guide
SMC	Cisco IOS XR System Management Configuration Guide
VFC	Cisco IOS XR Virtual Firewall Configuration Guide

Numerics

6PE
- BGP multipath **MPC-286**
- conditions for use **MPC-284**
- how to configure **MPC-286**
- how to deploy **MPC-285**
- overview **MPC-284**
- prerequisites **MPC-284**
- service provider considerations **MPC-284**
- supported protocols **MPC-287**

6VPE
- configuration tasks **MPC-424**
- how to configure PE-CE **MPC-428**
- how to configure VRFs **MPC-426**

A

access-lists, extended **MPC-57**

ACK (hello acknowledgment)
- objects **MPC-56**
- RSVP messages **MPC-56**

ACL-based prefix filtering, RSVP **MPC-57**

ACL match, how to return implicit deny **MPC-66**

active targeted hellos, prerequisites **MPC-17**

address space, PE routers **MPC-325**

ADM (Add Drop Multiplexer)
- how to define **MPC-199**
- O-UNI client device **MPC-200**
- with O-UNI **MPC-200**

advertisement, label **MPC-10**

advertising tunnels, VPN services **MPC-325**

aging, MAC address
- how to configure **MPC-276**
- how to define **MPC-245**

Any Transport over Multiprotocol (AToM)
- static labels, how to use **MPC-265**
- static pseudowire **MPC-265**

Asynchronous Transfer Mode (ATM)
- how to define **MPC-105**
- MPLS L2VPN **MPC-217**

attachment circuits
- how to define **MPC-243**
- xconnects **MPC-315**

automatic route distinguisher, MPLS Layer 3 VPN **MPC-353**

autonomous system **MPC-354**

B

bandwidth
- constraint models **MPC-101**
  - overview **MPC-101**
  - RDM and MAM **MPC-101**
- control channel, how to configure **MPC-61**

data channel, how to configure MPC-61
pools MPC-101
BGP
confederations MPC-355
distributing routes MPC-366
messages and MPLS labels MPC-366
routing information MPC-365
SSA function MPC-325
BGP4 configuration MPC-284
BGP multipath
6PE MPC-286
load sharing MPC-326
border control model
how to configure MPC-160
overview MPC-160
bridge domain
how to associate members MPC-251
how to configure parameters MPC-253
how to configure pseudowire MPC-249
how to create MPC-247
how to disable MPC-255
overview MPC-244
split horizon MPC-244

C

CEF (Cisco Express Forwarding)
L2TPv3 prerequisite MPC-291
CE-PE eBGP, how to configure route-policy definition MPC-329
CFI VRF interface, how to configure MPC-333
configuration examples
building MPLS-TE topology and tunnels MPC-188
fast reroute and SONET APS MPC-187
LDP
advertisement MPC-47
discovery MPC-46
discovery for targeted hellos MPC-46
forwarding MPC-47
IGP synchronization MPC-48, MPC-49
inbound label filtering MPC-48
link MPC-46
local label allocation control MPC-48
neighbors MPC-47
non-stop forwarding with graceful restart MPC-48
session protection MPC-48
with graceful restart MPC-46
MPLS L2VPN MPC-235
O-UNI
connection establishment MPC-211
connection tear-down MPC-211
neighbor and data link MPC-210
RSVP
ACL-based prefix filtering MPC-91
bandwidth (MAM) MPC-89
bandwidth (Prestandard) MPC-89
bandwidth (RDM) MPC-89
DSCP MPC-91
graceful restart MPC-90
control-channels
authentication parameters, how to configure MPC-300
L2TP maintenance parameters, how to configure MPC-308
message authentication MPC-300
method, how to configure MPC-301
parameters, how to configure MPC-298
timing parameters, how to configure MPC-299
control message
hashing, L2TPv3 MPC-302
with LDP MPC-3
control plane failure MPC-6
core network, how to configure MPC-334
CSC (Carrier Supporting Carrier)
configuration examples MPC-419
configuration options for backbone and customer carriers MPC-367
configuring a CSC-PE link MPC-413
configuring a static route to a peer MPC-419

customer carrier network options MPC-368
topology, how to identify MPC-412
CSC-CE link, how to configure MPC-417
CSC-PE link, how to configure MPC-413
customer edge router
6PE MPC-285
MPLS Layer 3 VPN MPC-349

D

Differentiated Services Traffic-Engineering
bandwidth, how to configure MPC-62
bandwidth constraints MPC-100
overview MPC-100
Diff-Serv
Russian Doll Model (RDM) and Maximum Allocation Model (MAM) MPC-100
digest secret graceful switchover, L2TPv3 MPC-304
DS-TE modes, prestandard and IETF MPC-62

E

eBGP MPC-348
end-to-end recovery, GMPLS MPC-107
EoMPLS
ethernet port mode MPC-219
Ethernet remote port shutdown MPC-219
inter-as port mode MPC-221
Mac-in-Mac MPC-222
overview MPC-218
QinAny mode MPC-222
QinQ mode MPC-221
ethernet port mode MPC-219
extended access-lists MPC-57
extensions, MPLS TE MPC-100

F

failure recovery, graceful restart MPC-8
flooding
MAC address MPC-245
MPLS-TE MPC-102
thresholds MPC-103
triggers MPC-103
frame relay, MPLS L2VPN MPC-217
FRR (Fast Reroute)
link protection MPC-103
over link bundles MPC-104
with MPLS TE MPC-103

G

GMPLS (Generalized Multiprotocol Label Switching)
benefits MPC-105
how to configure MPC-135
overview MPC-105
prerequisites MPC-107
protection and restoration MPC-106
support MPC-106
traffic engineering protocols MPC-105
graceful restart
failure recovery MPC-8
LDP MPC-6, MPC-27
mechanism MPC-7
NSR MPC-13
phases MPC-7
RSVP MPC-55
session parameters MPC-6

H

hello interval, how to change MPC-91
hierarchical optical network, GMPLS MPC-105
high availability, RSVP MPC-53

I

IETF DS-TE mode MPC-101

Ignore Intermediate System-to-Intermediate System (IS-IS)
- IP Fast Reroute Loopfree Alternative MPC-104
- overload bit setting
  - how to configure MPC-134
  - how to define MPC-104

IGP (Interior Gateway Protocols)
- prefixes MPC-4
- routing protocols MPC-3
- synchronization, LDP MPC-11
- with LDP MPC-1

implicit deny MPC-66

Inter-AS configurations
- BGP MPC-355
- interprovider VPN MPC-355
- L2VPN quality of service MPC-230
- supported MPC-355

Inter-AS mode MPC-221

interface ID application MPC-201

interprovider VPN, MPLS VPN MPC-355

IPCC (IP Control Channel)
- connectivity using O-UNI devices MPC-200
- how to configure MPC-136

IP router, O-UNI client device MPC-200

IP Time to Live (TTL) MPC-56

IPv4 loopback interface, how to configure MPC-331

IPv4 TNA address support MPC-201

ISP requirements, MPLS L2VPN MPC-217

L

L2TPv3
- benefits MPC-293
- control message hashing MPC-300
- dynamic
  - pseudowire, how to configure MPC-309
  - sessions MPC-294
- MPLS MPC-293
- multipoint tunnel network MPC-324
- operation MPC-292
- peer authentication MPC-294
- prerequisites MPC-291
- services MPC-292
- static sessions MPC-293
- xconnect support MPC-293

L2VPN, QoS restrictions MPC-230

label advertisement
- control, LDP MPC-10
- prerequisites MPC-21

label bindings
- how to configure MPC-4
- how to exchange MPC-4

LDP
- configuration examples MPC-45
- control communication failure MPC-8
- control messages MPC-3
- control plane MPC-3
  - failure MPC-6
- Control Protocol (example) MPC-3
- control state recovery MPC-8
- discovery
  - active targeted hellos, configuration MPC-17
  - parameters, configuring MPC-13
  - passive targeted hellos, configuration MPC-19
- discovery over a link
  - configuring MPC-15
  - prerequisites MPC-15
- dynamic path setup MPC-3
- forwarding, configuring MPC-25
- graceful restart MPC-6
  - failure recovery MPC-8
  - setting up LDP NSF MPC-27
- hello discovery mechanism MPC-3

hop-by-hop MPC-3
IGP prefixes MPC-4
IGP synchronization MPC-11
implementation MPC-13
keepalive mechanism MPC-4
label advertisement MPC-10
configuring MPC-21
control MPC-10
prerequisites MPC-21
local and remote label binding MPC-4
local label advertisement control MPC-10
local label allocation control MPC-10
LSPs, setting up MPC-4
neighbors
support for MPC-3
NSF services MPC-6
NSR MPC-12
peer control plane MPC-8
persistent forwarding MPC-8
prerequisites MPC-2
session protection MPC-11
LDP discovery prerequisites
for active targeted hellos MPC-17
for passive targeted hellos MPC-19
over a link MPC-15
LDP forwarding
how to configure MPC-25
how to set up MPC-5
prerequisites MPC-26
LDP label advertisement MPC-10
LDP neighbors prerequisites MPC-23
LDP NSF graceful restart prerequisites MPC-27
limit, MAC address
actions, types of MPC-246
how to configure MPC-274
LMP message exchange
disabling MPC-148
enabling MPC-146
local and remote TE links MPC-139
local label advertisement control, LDP MPC-10
local label allocation control, LDP MPC-10
local label binding MPC-4
local reservable bandwidth, how to configure MPC-142
local switching capability descriptors, how to configure MPC-143
loose hop reoptimization MPC-110
LSP
hierarchy MPC-159
how to configure MPC-159
how to define MPC-3
MPLS-TE MPC-99
with LDP MPC-3

M

MAC address
aging MPC-245
flooding MPC-245
forwarding MPC-245
limit actions MPC-246
related parameters MPC-244
source-based learning MPC-245
withdrawal MPC-246
Mac-in-Mac, Provider Backbone Bridging MPC-222
Maximum Allocation Model (MAM), constraint characteristics MPC-101
MDRR/WRED MPC-326
mesh restoration, GMPLS MPC-107
MFI (MPLS Forwarding Infrastructure)
control plane MPC-95
data plane services, about MPC-95
LDP MPC-95
TE MPC-95
modified deficit round robin MPC-326
MPLS forwarding forms MPC-95
MPLS L2VPN
configuration examples MPC-235
configuring MPC-224

high availability MPC-223
interface or connection, how to configure MPC-224
ISP requirements MPC-217
prerequisites MPC-216, MPC-242
Quality of service (QoS) MPC-222
VLAN mode, how to configure MPC-232
MPLS Layer 3 VPN
automatic route distinguisher MPC-353
autonomous system MPC-354
components MPC-349
concepts MPC-349
customer edge router MPC-349
customer router MPC-349
defined MPC-349
distributed routing information MPC-352
FIB MPC-348
how it works MPC-351
implementing MPC-349
major components MPC-353
MPLS forwarding MPC-352
PE router MPC-349
prerequisites MPC-348
provider router MPC-349
restrictions MPC-348
scalability MPC-350
security MPC-350
topology MPC-349
VPN routing information MPC-352
MPLS-TE
backbone MPC-99
benefits MPC-99
concepts MPC-100
engineering a backbone MPC-100
extensions MPC-100
fast reroute MPC-103
flooding MPC-102
flooding thresholds MPC-103
flooding triggers MPC-103
implementation MPC-115
link management module MPC-100
overview MPC-99
path calculation module MPC-100
prerequisites MPC-98
topology
building MPC-115
prerequisites MPC-115
tunnels
creating MPC-119
prerequisites MPC-119
with label switching forwarding MPC-100
with RSVP MPC-100
MPLS VPN
Inter-AS ASBRs MPC-354
major components MPC-353
MPLS VPNs over IP tunnels
BGP SSA MPC-325
concepts MPC-324
feature overview MPC-323
prerequisites MPC-324
QoS policy assignment MPC-326
quality of service MPC-326
restrictions MPC-324
task IDs MPC-324
tunnel types MPC-325
verifying MPC-342
VPN services MPC-324
MQC, quality of service MPC-326

N

NSF (Nonstop Forwarding)
enabling graceful restart MPC-63
high-availability MPC-54
with RSVP MPC-54
NSR (non-stop routing)
graceful restart MPC-13
how to define MPC-12
LDP MPC-12

numbered and unnumbered links, how to configure MPC-140
numbered optical TE tunnels, how to configure MPC-154

O

optical switches, GMPLS MPC-105
OSPF
- how to configure MPC-138
- over IPCC MPC-138

OTN (optical transport network)
- how to define MPC-199
- transport services MPC-200

O-UNI (Optical User Network Interface)
- active side, how to configure MPC-211
- bandwidth, how to configure MPC-63
- connections
  - how to configure MPC-202
  - how to establish MPC-201
  - how to tear down MPC-205
  - identifiers MPC-201
  - tear down example MPC-211
- database requirement MPC-200
- data-link, how to configure MPC-210
- how to verify MPC-207
- neighbor, how to configure MPC-210
- N node MPC-200
- passive side, how to configure MPC-211
- prerequisites
  - general MPC-200
  - how to set up a connection MPC-202
- router, how to configure MPC-210
- RSVP messages MPC-200
- standards MPC-199

O-UNI client devices
- ADMs MPC-200
- IP routers MPC-200

P

packet validation
- mechanism MPC-326
- MPLS VPNs over IP tunnels MPC-326

passive targeted hellos, how to configure MPC-19
path calculation module, MPLS-TE MPC-100
path protection, GMPLS
- forced reversion procedure MPC-164
- LSP
  - overview MPC-161
  - procedure MPC-161

PE-CE protocol, how to configure MPC-428
PE router
- address space MPC-325
- MPLS Layer 3 VPN MPC-349

persistent interface index, how to configure MPC-145
Policy-Based Tunnel Selection (PBTS)
- dynamic tunnel selection MPC-114
- figure, implementation MPC-113
- functions MPC-113
- how to configure MPC-185
- how to define MPC-113

port mode, MPLS L2VPN MPC-231
prefix filtering MPC-57
Prestandard DS-TE mode MPC-101
protection and restoration, GMPLS
- end-to-end recovery MPC-107
- requirements MPC-107
- shared mesh MPC-107
- span protection MPC-107

protocol-based CLI MPC-100
Provide Backbone Bridging protocol, Mac-in-Mac MPC-222
pseudowire (PW)
- bridge domain, how to configure MPC-249
- L2TPv3, how to configure MPC-309
- MPLS L2VPN MPC-219

pseudowire classes, how to configure L2TPv3 MPC-297

Q

QinAny mode MPC-222
QinQ mode MPC-221
QoS (quality of service)
- how to configure L2VPN MPC-231
- MPLS L2VPN MPC-222
- MPLS VPNs over IP tunnels MPC-326
- MQC MPC-326
- policy assignment, MPLS VPNs over IP tunnels MPC-326
- port mode, how to configure MPC-231

R

RDM bandwidth constraint model MPC-101
refresh interval, how to change MPC-90
remote label binding MPC-4
restart time, how to change MPC-91
route-policy definition, how to configure MPC-329
router IDs, how to configure MPC-136
RSVP
- ACL-based prefix filtering MPC-57
- compliance MPC-52
- configuration
  - ACL-based prefix filtering MPC-65
  - diffserv TE bandwidth MPC-62
  - graceful restart MPC-63
  - interface-based graceful restart MPC-63
  - O-UNI LSP MPC-52
  - Packet dropping MPC-66
  - tunnel bandwidth, engineering MPC-61
  - verifying MPC-68
- description MPC-51
- extensions MPC-53
  - generalized label request MPC-53
  - generalized UNI attribute MPC-53
  - New Error Spec sub-codes MPC-53
  - UNI session MPC-53
- fault handling MPC-55
- graceful restart MPC-54
- head node MPC-53
- hello messages MPC-56
- high availability MPC-53
- implementing MPC-61
- message rate limiting MPC-53
- node failure MPC-56
- overview MPC-52
- prerequisites MPC-52
- recovery time MPC-56
- refresh reduction MPC-53
- restart time MPC-56
- support for graceful restart MPC-53
- tail node MPC-53
- topology MPC-68
- with O-UNI LSP, configuring MPC-53

RSVP nodes
- head node MPC-53
- tail node MPC-53

Russian Doll Model (RDM) bandwidth constraint model MPC-101
RVSP node failure MPC-56

S

SAFI function MPC-325
SDH (Synchronous Digital Hierarchy) MPC-199
service provider edge routers, 6PE MPC-285
service providers, 6PE MPC-284
session protection, LDP MPC-11
signaling
- pseudowires MPC-244
- VPLS MPC-244

SONET (Synchronous Optical Network), with O-UNI MPC-200
source-based learning, how to configure MAC address MPC-269
span protection MPC-107

static
L2TPv3 pseudowires, how to configure MPC-312
L2TPv3 sessions, how to define MPC-293
point-to-point xconnects MPC-226
router to a peer, how to configure MPC-419
routes, how to configure MPC-330
summary refresh message size, how to change MPC-90

T

task IDs
required for MPLS VPNs over IP tunnels MPC-324
TE
class and attributes MPC-102
class mapping MPC-102
description MPC-97
thresholds, flooding MPC-103
TNA addresses MPC-201
triggers, flooding MPC-103
TTL
RSVP MPC-56
with graceful restart MPC-56
tunnel attributes MPC-325
tunnel bandwidth
MAM, how to configure MPC-62
RDM, how to configure MPC-62
tunnel types
6PE MPC-285
MPLS VPNs over IP tunnels MPC-325

U

unnumbered optical TE tunnels, how to configure MPC-154

V

verifying IP connectivity, CSC
MPLS Layer 3 VPN MPC-413
MPLS VPN over IP tunnel MPC-338
VFI (Virtual Forwarding Instance)
AToM pseudowires, how to configure MPC-265
bridge domain member, how to associate MPC-261
functions MPC-243
how to add under bridge domain MPC-257
how to disable MPC-267
pseudowire classes to pseudowires, how to attach MPC-263
pseudowires, how to associate MPC-259
VLAN
figure, mode packet flow MPC-220
mode MPC-220
VPLS (Virtual Private LAN Services)
attachment circuits MPC-243
bridge domain, how to define MPC-244
Layer 2 VPN, architecture MPC-243
overview MPC-243
signaling, how to define MPC-244
virtual bridge, how to simulate MPC-244
VPN services
L2TPv3 MPC-324
MPLS MPC-324
VRF (Virtual Routing and Forwarding)
CSC-PE routers, how to configure MPC-338, MPC-413
interface MPC-325

W

withdrawal, MAC address
fields MPC-273
how to define MPC-246
how to enable MPC-271

X

xconnects
attachment circuits, how to configure MPC-315
support MPC-293