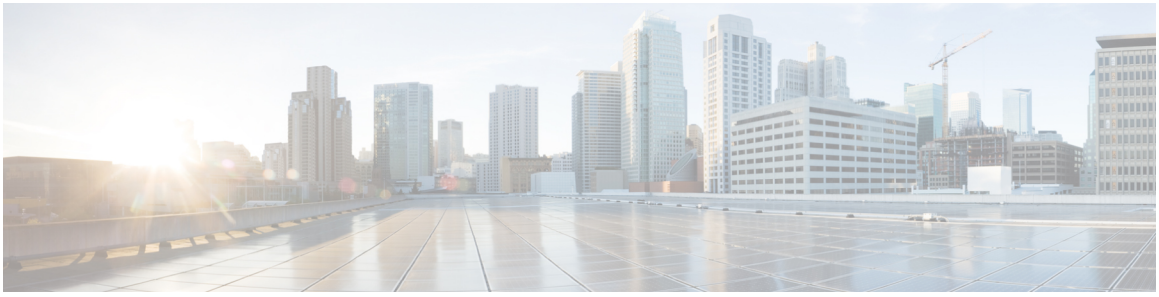


LAN and WAN Configuration Guide, Cisco IOS XE 17.x

First Published: 2022-11-05

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



CONTENTS

Full Cisco Trademarks with Software License ?

PREFACE

Preface xxvii

Preface xxvii

Audience and Scope xxvii

Feature Compatibility xxviii

Document Conventions xxviii

Communications, Services, and Additional Information xxix

Documentation Feedback xxx

Troubleshooting xxx

PART I

Local Area Networking 31

CHAPTER 1

Configuring ERSPAN 1

Restrictions for Configuring ERSPAN 1

Information About Configuring ERSPAN 2

ERSPAN Overview 2

ERSPAN Sources 3

ERSPAN Destination Ports 4

Using ERSPAN as Local SPAN 4

ERSPAN Support on WAN Interface 5

ERSPAN Dummy MAC Address Rewrite 5

ERSPAN IP Access Control Lists 5

How to Configure ERSPAN 5

Configuring an ERSPAN Source Session 5

Configuring an ERSPAN Destination Session 9
Configuring ERSPAN Dummy MAC Address Rewrite 11
Configuration Examples for ERSPAN 12
Example: Configuring an ERSPAN Source Session 12
Example: Configuring an ERSPAN Source Session on a WAN Interface 13
Example: Configuring an ERSPAN Destination Session 13
Example: Configuring an ERSPAN as a Local SPAN 13
Example: Configuring ERSPAN Dummy MAC Address Rewrite 13
Example: Configuring UDF-Based ERSPAN 14
Additional References for Configuring ERSPAN 14
Feature Information for Configuring ERSPAN 15

CHAPTER 2 **Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 17**
Restrictions for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 17
Information About Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 17
Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 17
How to Configure Routing Between VLANs with IEEE 802.1Q Encapsulation 18
Configuring IP Routing over IEEE 802.1Q 18
Enabling IP Routing 18
Defining the VLAN Encapsulation Format 19
Assigning an IP Address to Network Interface 20
Monitoring and Maintaining VLAN Subinterfaces 21
Configuration Examples for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 22
Configuring IP Routing over IEEE 802.1Q Example 22
Additional References 22
Feature Information for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 23

CHAPTER 3 **IEEE 802.1Q-in-Q VLAN Tag Termination 25**
Information About IEEE 802.1Q-in-Q VLAN Tag Termination 25
IEEE 802.1Q-in-Q VLAN Tag Termination on Subinterfaces 25
Unambiguous and Ambiguous Subinterfaces 26
IEEE802.1ad Support in Port-channels and Subinterfaces 27
Restrictions for IEEE802.1ad Support in Port-channels and Subinterfaces 27
How to Configure IEEE 802.1Q-in-Q VLAN Tag Termination 27

Configuring the Interfaces for IEEE 802.1Q-in-Q VLAN Tag Termination 27
Verifying the IEEE 802.1Q-in-Q VLAN Tag Termination 29
Configuring IEEE 802.1ad in Port-channels and Subinterfaces 31
Configuration Examples for IEEE 802.1Q-in-Q VLAN Tag Termination 32
Configuring any Keyword on Subinterfaces for IEEE 802.1Q-in-Q VLAN Tag Termination Example 32
Additional References 34
Feature Information for IEEE 802.1Q-in-Q VLAN Tag Termination 35

CHAPTER 4 **VLAN Mapping to Gigabit EtherChannel Member Links 37**
Prerequisites for VLAN Mapping to GEC Member Links 37
Restrictions for VLAN Mapping to GEC Member Links 37
Information About VLAN Mapping of GEC Member Links 38
VLAN-Manual Load Balancing 38
VLAN-to-Port Channel Member Link Mapping 39
VLAN Primary and Secondary Link Association 39
Adding Channel Member Links 41
Deleting Member Links 41
Port Channel Link Down Notification 41
Port Channel Link Up Notification 41
Disabling Load Balancing on the EtherChannel 42
Removing a Member Link from the EtherChannel 42
How to Configure VLAN Mapping to GEC Links 42
Configuring VLAN-Based Manual Load Balancing 42
Troubleshooting Tips 44
Configuration Examples for VLAN Mapping to GEC Member Links 44
Example: Configuring VLAN Manual Load Balancing 44
Example: Troubleshooting 46
Additional References 47
Feature Information for VLAN Mapping to GEC Member Links 47

CHAPTER 5 **Configuring Routing Between VLANs 49**
Information About Routing Between VLANs 49
Virtual Local Area Network Definition 49

LAN Segmentation 50
Security 50
Broadcast Control 50
VLAN Performance 51
Network Management 51
Network Monitoring Using SNMP 51
Communication Between VLANs 51
Relaying Function 51
Native VLAN 53
PVST+ 54
Ingress and Egress Rules 55
Integrated Routing and Bridging 55
VLAN Colors 55
Implementing VLANS 56
Communication Between VLANs 56
Inter-Switch Link Protocol 56
IEEE 802.10 Protocol 56
IEEE 802.1Q Protocol 57
ATM LANE Protocol 57
ATM LANE Fast Simple Server Replication Protocol 57
VLAN Interoperability 58
Inter-VLAN Communications 58
VLAN Translation 58
Designing Switched VLANs 59
Frame Tagging in ISL 59
IEEE 802.1Q-in-Q VLAN Tag Termination on Subinterfaces 60
Cisco 10000 Series Internet Router Application 61
Security ACL Application on the Cisco 10000 Series Internet Router 62
Unambiguous and Ambiguous Subinterfaces 62
How to Configure Routing Between VLANS 63
Configuring a VLAN Range 63
Restrictions 63
Configuring a Range of VLAN Subinterfaces 63
Configuring Routing Between VLANs with Inter-Switch Link Encapsulation 65

Configuring AppleTalk Routing over ISL 65
Configuring Banyan VINES Routing over ISL 66
Configuring DECnet Routing over ISL 68
Configuring the Hot Standby Router Protocol over ISL 69
Configuring IP Routing over TRISL 71
Configuring IPX Routing on 802.10 VLANs over ISL 73
Configuring IPX Routing over TRISL 74
Configuring VIP Distributed Switching over ISL 76
Configuring XNS Routing over ISL 78
Configuring CLNS Routing over ISL 79
Configuring IS-IS Routing over ISL 81
Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation 82
Prerequisites 82
Restrictions 82
Configuring AppleTalk Routing over IEEE 802.1Q 83
Configuring IP Routing over IEEE 802.1Q 84
Configuring IPX Routing over IEEE 802.1Q 85
Configuring a VLAN for a Bridge Group with Default VLAN1 86
Configuring a VLAN for a Bridge Group as a Native VLAN 87
Configuring IEEE 802.1Q-in-Q VLAN Tag Termination 89
Configuring EtherType Field for Outer VLAN Tag Termination 89
Configuring the Q-in-Q Subinterface 90
Verifying the IEEE 802.1Q-in-Q VLAN Tag Termination 92
Monitoring and Maintaining VLAN Subinterfaces 95
Monitoring and Maintaining VLAN Subinterfaces Example 95
Configuration Examples for Configuring Routing Between VLANs 96
Single Range Configuration Example 96
ISL Encapsulation Configuration Examples 97
AppleTalk Routing over ISL Configuration Example 97
Banyan VINES Routing over ISL Configuration Example 98
DECnet Routing over ISL Configuration Example 98
HSRP over ISL Configuration Example 98
IP Routing with RIF Between TrBRF VLANs Example 100
IP Routing Between a TRISL VLAN and an Ethernet ISL VLAN Example 101

IPX Routing over ISL Configuration Example 101
IPX Routing on FDDI Interfaces with SDE Example 103
Routing with RIF Between a TRISL VLAN and a Token Ring Interface Example 103
VIP Distributed Switching over ISL Configuration Example 104
XNS Routing over ISL Configuration Example 105
CLNS Routing over ISL Configuration Example 105
IS-IS Routing over ISL Configuration Example 105
Routing IEEE 802.10 Configuration Example 106
IEEE 802.1Q Encapsulation Configuration Examples 107
Configuring AppleTalk over IEEE 802.1Q Example 107
Configuring IP Routing over IEEE 802.1Q Example 107
Configuring IPX Routing over IEEE 802.1Q Example 107
VLAN 100 for Bridge Group 1 with Default VLAN1 Example 107
VLAN 20 for Bridge Group 1 with Native VLAN Example 107
VLAN ISL or IEEE 802.1Q Routing Example 108
VLAN IEEE 802.1Q Bridging Example 109
VLAN IEEE 802.1Q IRB Example 109
Configuring IEEE 802.1Q-in-Q VLAN Tag Termination Example 110
Additional References 112
Feature Information for Routing Between VLANs 113

CHAPTER 6 **EtherChannel Flow-Based Limited 1 1 Redundancy 117**
Restrictions for EtherChannel Flow-based Limited 1:1 Redundancy 117
Information About EtherChannel Flow-Based Limited 1 1 Redundancy 118
EtherChannel Flow-Based Limited 1 1 Redundancy 118
How to Configure EtherChannel Flow-Based Limited 1 1 Redundancy 118
Configuring EtherChannel Flow-Based Limited 1 1 Redundancy with Fast-Switchover 118
Setting the Switchover Rate with Carrier Delay 121
Verifying EtherChannel Flow-Based Limited 1 1 Redundancy 121
Configuration Examples for EtherChannel Flow-Based Limited1 1 Redundancy 123
EtherChannel 1 1 Active Standby Example 123
Setting Priority for 1 1 Redundancy Using LACP Example 123
Additional References 124
Feature Information for EtherChannel Flow-based Limited 1 1 Redundancy 125

CHAPTER 7 **Flow-Based per Port-Channel Load Balancing** 127

Restrictions for Flow-Based per Port-Channel Load Balancing 127
Information About Flow-Based per Port-Channel Load Balancing 127
Flow-Based Load Balancing 127
Buckets for Flow-Based Load Balancing 128
Load Balancing on Port Channels 129
How to Enable Flow-Based per Port-Channel Load Balancing 130
Configuring Load Balancing on a Port Channel 130
Verifying Load-Balancing Configuration on a GEC Interface 131
Configuration Examples for Flow-Based per Port-Channel Load Balancing 133
Flow-Based Load Balancing Example 133
Information About Five-Tuple Hash Support for GEC Flow-based Load Balancing 134
Restrictions for Five-Tuple Hash Support for GEC Flow-based Load Balancing 134
Configuring Five-Tuple Hash Support for GEC Flow-based Load Balancing 134
Additional References 134
Feature Information for Flow-Based per Port-Channel Load Balancing 136

CHAPTER 8 **VLANs over IP Unnumbered SubInterfaces** 137

Prerequisites for VLANs over IP Unnumbered Subinterfaces 137
Restrictions for VLANs over IP Unnumbered Subinterfaces 137
Information About VLANs over IP Unnumbered Subinterfaces 138
Support for VLANs over IP Unnumbered Subinterfaces 138
DHCP Option 82 138
Benefits of VLANs over IP Unnumbered Subinterfaces 139
How to Configure VLANs over IP Unnumbered Subinterfaces 140
Configuring IP Unnumbered Interface Support on an Ethernet VLAN Subinterface 140
Configuring IP Unnumbered Interface Support on a Range of Ethernet VLAN Subinterfaces 141
Configuration Examples for VLANs over IP Unnumbered Subinterfaces 143
Example: VLAN Configuration on a Single IP Unnumbered Subinterface 143
Example: VLAN Configuration on a Range of IP Unnumbered Subinterfaces 143
Additional References for VLANs over IP Unnumbered Subinterfaces 143
Feature Information for VLANs over IP Unnumbered Subinterfaces 144

CHAPTER 9 **Spanning Tree Protocol** 145

Information About Spanning Tree Protocol 145

Using the Spanning Tree Protocol with the EtherSwitch Network Module 145

Spanning Tree Port States 146

Default Spanning Tree Configuration 148

Bridge Protocol Data Units 149

STP Timers 152

Spanning Tree Port Priority 152

Spanning Tree Port Cost 152

Spanning Tree Root Bridge 153

How to Configure Spanning Tree Protocol 154

Enabling Spanning Tree Protocol 154

Configuring the Bridge Priority of a VLAN 155

Configuring STP Timers 156

Configuring Hello Time 156

Configuring the Forward Delay Time for a VLAN 156

Configuring the Maximum Aging Time for a VLAN 157

Configuring Spanning Tree Port Priority 158

Configuring Spanning Tree Port Cost 159

Configuring Spanning Tree Root Bridge 160

Verifying Spanning Tree on a VLAN 161

Configuration Examples for Spanning Tree Protocol 162

Example: Enabling Spanning Tree Protocol 162

Example: Configuring the Bridge Priority of a VLAN 163

Example: Configuring STP Timers 163

Example: Configuring Hello Time 163

Example: Configuring the Forward Delay Time for a VLAN 163

Example: Configuring the Maximum Aging Time for a VLAN 163

Example: Configuring Spanning Tree Port Priority 163

Example: Configuring Spanning Tree Port Cost 164

Example: Configuring Spanning Tree Root Bridge 165

Additional References 165

Feature Information for Spanning Tree Protocol 165

PART II Wide Area Networking 167

CHAPTER 10 Wide-Area Networking Overview 169

Frame Relay 169

Frame Relay-ATM Internetworking 171

Layer 2 Virtual Private Network 172

Layer 2 Tunneling Protocol Version 3 172

L2VPN Pseudowire Redundancy 172

Layer 2 Virtual Private Network Interworking 172

Layer 2 Local Switching 173

CHAPTER 11 Layer 2 Tunneling Protocol Version 3 175

Feature Information for Layer 2 Tunneling Protocol Version 3 175

Prerequisites for Layer 2 Tunneling Protocol Version 3 176

Restrictions for Layer 2 Tunneling Protocol Version 3 177

General L2TPv3 Restrictions 177

VLAN-Specific Restrictions 178

IPv6 Protocol Demultiplexing for L2TPv3 Restrictions 178

L2TPv3 Control Message Hashing Restrictions 178

L2TPv3 Digest Secret Graceful Switchover Restrictions 178

Quality of Service Restrictions in L2TPv3 Tunneling 179

Information About Layer 2 Tunneling Protocol Version 3 179

L2TPv3 Header Description 179

Session ID 180

Session Cookie 180

Pseudowire Control Encapsulation 180

L2TPv3 Operation 180

L2TPv3 Features 182

Static L2TPv3 Sessions 182

Dynamic L2TPv3 Sessions 182

Control Channel Parameters 182

L2TPv3 Control Channel Authentication Parameters 182

Ethernet over L2TPv3 184

GEC over L2TPv3 185
Sequencing 185
L2TPv3 Type of Service Marking 186
Keepalive 186
MTU Handling 186
L2TPv3 Control Message Hashing 187
L2TPv3 Control Message Rate Limiting 187
L2TPv3 Digest Secret Graceful Switchover 188
L2TPv3 Pseudowire 188
Manual Clearing of L2TPv3 Tunnels 189
L2TPv3 Tunnel Management 189
L2TPv3 Protocol Demultiplexing 189
L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations 189
HDLC over L2TPv3 189
L2TPv3 Benefits 190
Supported L2TPv3 Payloads 190
Ethernet 190
VLAN 191
IPv6 Protocol Demultiplexing 191
Performance Impact of L2TPv3 on Cisco ASR 1000 Series Routers 192
Layer 2 Protocol Tunneling and Forwarding 193
How to Configure Layer 2 Tunneling Protocol Version 3 193
Configuring L2TP Control Channel Parameters 193
Configuring L2TP Control Channel Timing Parameters 193
Configuring L2TPv3 Control Channel Authentication Parameters 195
Configuring L2TP Control Channel Maintenance Parameters 201
Configuring the L2TPv3 Pseudowire 202
Configuring the Xconnect Attachment Circuit 205
Configure L2TPv3 on a Switched Virtual Interface 207
Manually Configuring L2TPv3 Session Parameters 209
Configuring Protocol Demultiplexing for L2TPv3 211
Configuring Protocol Demultiplexing for Ethernet Interfaces 211
Configuring an L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations 212
Configuring GEC over L2TPv3 213

Configuring GEC with Dot1Q 216
Configuring GEC with QinQ 217
Manually Clearing L2TPv3 Tunnels 218
Configuration Examples for Layer 2 Tunneling Protocol Version 3 219
Example: Configuring an L2TPv3 Session for an Xconnect Ethernet Interface 219
Example: Configuring a Negotiated L2TPv3 Session for an Xconnect VLAN Subinterface 220
Example: Configure a Static L2TPv3 Session for a SVI 220
Example: Configuring a Negotiated L2TPv3 Session for Local HDLC Switching 220
Example: Verifying an L2TPv3 Session 221
Example: Verify a Static L2TPv3 Session for a Switched Virtual Interface 222
Example: Verifying an L2TP Control Channel 222
Example: Configuring L2TPv3 Control Channel Authentication 223
Example: Configuring L2TPv3 Digest Secret Graceful Switchover 223
Example: Verifying L2TPv3 Digest Secret Graceful Switchover 223
Example: Configuring a Pseudowire Class for Fragmentation of IP Packets 224
Example: Configuring Protocol Demultiplexing for L2TPv3 224
Example: Manually Clearing an L2TPv3 Tunnel 224
Example: Configuring an L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations 224
Example: Configuring an L2TPv3 HDLC Like-to-Like Layer 2 Transport 225
Example: Configuring an L2TPv3 HDLC Like-to-Like Layer 2 Transport on Dynamic Mode 225
Example: Configuring an L2TPv3 HDLC Like-to-Like Layer 2 Transport on Static Mode 225
Example: Configuring GEC over L2TPv3 225
Example: Configuring GEC with Dot1q over L2TPv3 226
Example: Configuring GEC with QinQ over L2TPv3 226
Additional References 226
Glossary 227

CHAPTER 12 **L2VPN Pseudowire Redundancy** 231
Prerequisites for L2VPN Pseudowire Redundancy 231
Restrictions for L2VPN Pseudowire Redundancy 232
Information About L2VPN Pseudowire Redundancy 232
Introduction to L2VPN Pseudowire Redundancy 232
How to Configure L2VPN Pseudowire Redundancy 234
Configuring the Pseudowire 234

Configuring the Pseudowire using the commands associated with the L2VPN Protocol-Based CLIs feature 235
Configuring L2VPN Pseudowire Redundancy 237
Configuring L2VPN Pseudowire Redundancy using the commands associated with the L2VPN Protocol-Based CLIs feature 238
Forcing a Manual Switchover to the Backup Pseudowire VC 240
Verifying the L2VPN Pseudowire Redundancy Configuration 241
Verifying the L2VPN Pseudowire Redundancy Configuration using the commands associated with the L2VPN Protocol-Based CLIs feature 243
Configuration Examples for L2VPN Pseudowire Redundancy 245
Example L2VPN Pseudowire Redundancy and AToM (Like to Like) 245
Example L2VPN Pseudowire Redundancy and L2VPN Interworking 245
Example L2VPN Pseudowire Redundancy with Layer 2 Local Switching 246
Example L2VPN Pseudowire Redundancy and Layer 2 Tunneling Protocol Version 3 246
Configuration Examples for L2VPN Pseudowire Redundancy using the commands associated with the L2VPN Protocol-Based CLIs feature 247
Example L2VPN Pseudowire Redundancy and AToM (Like to Like) using the commands associated with the L2VPN Protocol-Based CLIs feature 248
Example L2VPN Pseudowire Redundancy and L2VPN Interworking using the commands associated with the L2VPN Protocol-Based CLIs feature 248
Example L2VPN Pseudowire Redundancy and Layer 2 Tunneling Protocol Version 3 using the commands associated with the L2VPN Protocol-Based CLIs feature 249
Additional References 251
Feature Information for L2VPN Pseudowire Redundancy 252

CHAPTER 13 **Layer 2 Local Switching** 255
Prerequisites for Layer 2 Local Switching 255
Restrictions for Layer 2 Local Switching 256
Information About Layer 2 Local Switching 256
Layer 2 Local Switching Overview 256
NSF SSO—Local Switching Overview 256
Layer 2 Local Switching Applications 256
How to Configure Layer 2 Local Switching 257
Configuring Ethernet VLAN Same-Port Switching 257
Configuring Ethernet Port Mode to Ethernet VLAN Local Switching 258

Configuring ATM-to-ATM PVC Local Switching and Same-Port Switching 260
Configuring ATM-to-ATM PVP Local Switching 261
Configuring ATM PVP Same-Port Switching 262
Configuring Frame Relay-to-Frame Relay Local Switching 263
Verifying Layer 2 Local Switching 265
Verifying Layer 2 Local Switching Configuration 265
Verifying the NSF SSO Local Switching Configuration 265
Troubleshooting Tips 266
Configuration Examples for Layer 2 Local Switching 267
Example: Configuring Ethernet VLAN Same-Port Switching 267
Example: Configuring NSF SSO Ethernet Port Mode to Ethernet VLAN Local Switching 267
Example: ATM-to-ATM Local Switching 269
Example: ATM PVC Same-Port Switching 270
Example: ATM PVP Same-Port Switching 270
Example: Configuring Frame Relay-to-Frame Relay Local Switching 270
Additional References 270
Feature Information for Layer 2 Local Switching 271

PART III **Frame Relay** 273

CHAPTER 14 **Configuring Frame Relay** 275
Restrictions for Configuring Frame Relay 275
Information About Frame Relay 276
Frame Relay Hardware Configurations 276
Frame Relay Encapsulation 276
Dynamic or Static Address Mapping 277
Dynamic Address Mapping 277
Static Address Mapping 277
LMI 277
Activating LMI Autosense 278
MQC-Based Frame Relay Traffic Shaping 278
Traffic-Shaping Map Class for the Interface 279
Specifying Map Class with Queueing and Traffic-Shaping Parameters 279
Defining Access Lists 279

Understanding Frame Relay Subinterfaces 279
Subinterface Addressing 280
Backup Interface for a Subinterface 281
Disabling or Reenabling Frame Relay Inverse ARP 281
Frame Relay Fragmentation 281
End-to-End FRF.12 Fragmentation 281
TCP IP Header Compression 282
Specifying an Individual IP Map for TCP IP Header Compression 283
Specifying an Interface for TCP IP Header Compression 283
Real-Time Header Compression with Frame Relay Encapsulation 283
Discard Eligibility 284
DLCI Priority Levels 284
How to Configure Frame Relay 285
Enabling Frame Relay Encapsulation on an Interface 285
Configuring Static Address Mapping 286
Explicitly Configuring the LMI 287
Setting the LMI Type 287
Setting the LMI Keepalive Interval 288
Setting the LMI Polling and Timer Intervals 289
Configuring MQC-Based Frame Relay Traffic Shaping 289
Specifying a Traffic-Shaping Map Class for the Interface 289
Defining a Map Class with Queueing and Traffic-Shaping Parameters 290
Customizing Frame Relay for Your Network 291
Configuring Frame Relay Subinterfaces 291
Disabling or Reenabling Frame Relay Inverse ARP 293
Configuring Frame Relay Fragmentation 293
Configuring TCP IP Header Compression 294
Configuring Discard Eligibility 295
Configuring DLCI Priority Levels 295
Monitoring and Maintaining the Frame Relay Connections 296
Configuration Examples for Frame Relay 296
Example IETF Encapsulation 296
Example IETF Encapsulation on the Interface 296
Example IETF Encapsulation on a Per-DLCI Basis 297

Example Static Address Mapping 297
Example Two Routers in Static Mode 297
Example Subinterface 297
Example Basic Subinterface 297
Example Frame Relay Traffic Shaping 297
Example Configuring Class-Based Weighted Fair Queueing 297
Example Configuring Class-Based Weighted Fair Queueing with Fragmentation 298
Example Backward Compatibility 298
Example Booting from a Network Server over Frame Relay 299
Example Frame Relay Fragmentation Configuration 299
Example FRF.12 Fragmentation 299
Example TCP IP Header Compression 300
Example IP Map with Inherited TCP IP Header Compression 300
Example Using an IP Map to Override TCP IP Header Compression 300
Example Disabling Inherited TCP IP Header Compression 300
Example Disabling Explicit TCP IP Header Compression 301
Additional References 302
Feature Information for Configuring Frame Relay 303

CHAPTER 15 **Frame Relay Queueing and Fragmentation at the Interface** 305
Restrictions for Frame Relay Queueing and Fragmentation at the Interface 305
Information About Frame Relay Queueing and Fragmentation at the Interface 305
How Frame Relay Queueing and Fragmentation at the Interface Works 306
Benefits of Frame Relay Queueing and Fragmentation at the Interface 307
How to Configure Frame Relay Queueing and Fragmentation at the Interface 307
Configuring Class Policy for the Priority Queue 307
Configuring Class Policy for the Bandwidth Queues 308
Configuring the Shaping Policy Using the Class-Default Class 310
Configuring Queueing and Fragmentation on the Frame Relay Interface 311
Verifying Frame Relay Queueing and Fragmentation at the Interface 312
Monitoring and Maintaining Frame Relay Queueing and Fragmentation at the Interface 315
Configuration Examples for Frame Relay Queueing and Fragmentation at the Interface 315
Example Frame Relay Queueing Shaping and Fragmentation at the Interface 315
Example Frame Relay Queueing and Fragmentation at the Interface 316

Additional References 316
Feature Information for Frame Relay Queueing and Fragmentation at the Interface 317

CHAPTER 16 **Frame Relay MIB Enhancements 319**
Prerequisites for Frame Relay MIB Enhancements 319
Restrictions for Frame Relay MIB Enhancements 319
Information About Frame Relay MIB Enhancements 320
Feature Overview 320
Benefits 320
How to Configure Frame Relay MIB Enhancements 321
Setting the Load Interval for a PVC 321
Verifying the Load Interval 321
Configuration Examples for Frame Relay MIB Enhancements 321
Example Setting the Load Interval for a PVC 321
Additional References 322
Feature Information for Frame Relay MIB Enhancements 323

CHAPTER 17 **Frame Relay PVC Interface Priority Queueing 325**
Prerequisites for Frame Relay PVC Interface Priority Queueing 325
Restrictions for Frame Relay PVC Interface Priority Queueing 325
Information About Frame Relay PVC Interface Priority Queueing 326
Feature Overview 326
Benefits 327
How to Configure Frame Relay PVC Interface Priority Queueing 327
Configuring PVC Priority in a Map Class 327
Enabling FR PIPQ and Setting Queue Limits 327
Assigning a Map Class to a PVC 328
Verifying FR PIPQ 328
Monitoring and Maintaining FR PIPQ 329
Configuration Examples for Frame Relay PVC Interface Priority Queuing 329
FR PIPQ Configuration Example 329
Additional References 330
Feature Information for Frame Relay PVC Interface Priority Queueing 331
Glossary 332

CHAPTER 18 **ASR1K Frame Relay - Multilink (MLFR-FRF.16)** 333

Prerequisites for ASR1K Frame Relay - Multilink (MLFR-FRF.16) 333

Restrictions for ASR1K Frame Relay - Multilink (MLFR-FRF.16) 333

Information About ASR1K Frame Relay - Multilink (MLFR-FRF.16) 334

Benefits of ASR1K Frame Relay - Multilink (MLFR-FRF.16) 334

Flexible Pool of Bandwidth 334

Increased Service Resilience 334

Scalability 335

Link Integrity Protocol Control Messages 335

Variable Bandwidth Class Support 336

Class A Single Link 336

Class B All Links 336

Class C Threshold 336

Load Balancing 336

ASR1K FRF.12 Support on MFR Interfaces 337

Benefits of ASR1K FRF.12 337

Limitations of ASR1K FRF.12 337

Selecting a Fragment Size 337

How to Enable ASR1K Frame Relay - Multilink (MLFR-FRF.16) 338

Configuring an MFR Bundle 338

Configuring an MFR Bundle Link 340

Configuring FRF.12 on an MFR Bundle Interface 342

Monitoring and Maintaining MFR Bundles and Bundle Links 344

Configuration Examples for ASR1K Frame Relay - Multilink (MLFR-FRF.16) 345

Example: Configuring Multilink Frame Relay 345

Example: Configuring Variable Bandwidth Class Support 346

Example: Configuring FRF.12 on an MFR Interface 346

Additional References 347

Feature Information for ASR1K Frame Relay - Multilink (MLFR-FRF.16) 347

Glossary 348

CHAPTER 19 **Frame Relay show Command and debug Command Enhancements** 351

Information About Frame Relay show Command and debug Command Enhancements 351

Overview of the Frame Relay show Command and debug Command Enhancements 351
Benefits of the Frame Relay Show Command and Debug Command Enhancements 352
Additional References 352
Feature Information for Frame Relay show Command and debug Command Enhancements 353

CHAPTER 20 L2VPN Local Switching—Frame Relay-Ethernet/VLAN 355
Restrictions for L2VPN Local Switching—Frame Relay-Ethernet/VLAN 355
Information About L2VPN Local Switching—Frame Relay-Ethernet/VLAN 355
L2VPN Local Switching—Frame Relay-Ethernet/VLAN Overview 355
Frame Relay to Ethernet Port-Bridged Interworking 356
Frame Relay to Ethernet VLAN/QinQ–Bridged Interworking 357
How To Configure L2VPN Local Switching—Frame Relay-Ethernet/VLAN 358
Configuring Frame Relay-Ethernet Port-Bridged Interworking 358
Configuring Frame Relay-Ethernet VLAN/QinQ Interworking 360
Configuration Examples for L2VPN Local Switching—Frame Relay-Ethernet/VLAN 361
Example: Configuring Frame Relay-Ethernet Port Mode Bridged Interworking 361
Example: Configuring Frame Relay-Ethernet VLAN 802.1Q Bridged Interworking 362
Example: Configuring Frame Relay-VLAN QinQ Bridged Interworking 363
Additional References for L2VPN Local Switching—Frame Relay-Ethernet/VLAN 363
Feature Information for L2VPN Local Switching—Frame Relay-Ethernet/VLAN 364

PART IV WAAS 365

CHAPTER 21 mDNS for kWAAS 367
Information About mDNS for kWAAS 367
Overview of Service Discovery on mDNS for kWAAS 367
Overview of IP Networking on mDNS for kWAAS 368
Additional References for mDNS for kWAAS 368
Feature Information for mDNS for kWAAS 369

PART V Multilink PPP 371

CHAPTER 22 Multilink PPP Support 373
Cisco IOS XE Scaling Limits for MLP Bundles 373

Restrictions for MLP over Serial Interfaces 375
Restrictions for MLP over Ethernet at PTA and LAC 376
Restrictions for MLP over ATM at PTA and LAC 376
Restrictions for MLP at LAC 377
Restrictions for MLP over LNS 377
Restrictions for Broadband MLP at PTA and LNS 377
Information About Multilink PPP Support 378
Quality of Service 378
Multilink PPP Packet Overhead Accounting for Shaping and Policing 379
Downstream Model-F Shaper on LNS 380
Bandwidth 381
MTU 381
Downstream LFI 382
MLP Fragmentation Model 384
IP Type of Service Reflect 384
IP Tunnel Marking 385
Unsupported Features 385
Additional References for Multilink PPP Support 385
Feature Information for Multilink PPP Support 387

CHAPTER 23 **Configuring Multilink PPP Connections for Broadband and Serial Topologies** 389
Restrictions for Multilink PPP Connections for Broadband and Serial Topologies 389
Information About Multilink PPP Connections for Broadband and Serial Topologies 390
Multilink PPP 390
Multilink PPP Bundles 390
Multilink PPP Bundles and PPP Links 391
Link Fragmentation and Interleaving 392
Types of Multilink PPP Bundle Interfaces 392
Multilink Group Interface 392
Virtual Access Interface 392
Factors that Govern a Link Joining a Bundle 393
Rate of Session Establishment for Multilink PPP Bundles 394
Multilink PPP Packet Overhead 395
Multilink PPP over Serial Interfaces 395

Multilink PPP over Broadband 396
PTA Mode 397
LNS Mode 397
Performance- and Scalability-Related Commands 398
Multilink PPP over ATM on the PTA Device 399
Multilink PPP over Ethernet over ATM on the PTA Device 399
Multilink PPP over LNS 399
QoS Traffic and Shaping 401
How to Configure Multilink PPP Connections for Broadband and Serial Topologies 401
Configuring Multilink PPP 401
Creating a Multilink Bundle 401
Assigning an Interface to a Multilink Bundle 402
Configuring Minimum Multilink PPP Links 404
Changing the Default Endpoint Discriminator 405
Configuring Multilink PPP Interleaving and Queueing 406
Configuring Multilink PPP Interleaving 406
Disabling PPP Multilink Fragmentation 407
Configuring Multilink PPP over Broadband 408
Creating a Class Map 408
Creating a Policy Map 409
Defining a PPP over Ethernet Profile 410
Configuring a Virtual Template Interface 411
Configuring Multilink PPP over ATM on the CPE Device 412
Configuring Multilink PPP over Ethernet over ATM at the CPE 415
Configuring Multilink PPP over ATM on the PTA Device 416
Configuring Multilink PPP over Ethernet over ATM on the PTA Device 418
Configuring Multilink PPP over LNS 421
Configuring Multilink PPP over Serial Interfaces 424
Configuration Examples for Multilink PPP Connections for Broadband and Serial Topologies 426
Example: Configuring Multilink PPP 426
Example: Configuring Multilink PPP over ATM on the PTA Device 428
Example: Configuring Multilink PPP over ATM Using AAL5 MUX Encapsulation 428
Example: Configuring Multilink PPP over ATM Using AAL5 SNAP Encapsulation 429
Example: Configuring Multilink PPP over Ethernet over ATM on the PTA Device 430

Example: Configuring Multilink PPP over LNS 431
Example: Configuring an LNS Device to Initiate and Receive L2TP Traffic 431
Example: Configuring a LAC Device to Initiate and Receive L2TP Traffic 432
Example: Configuring Multilink PPP over Serial Interfaces 433
Additional References for Multilink PPP Connections for Broadband and Serial Topologies 435
Feature Information for Multilink PPP Connections for Broadband and Serial Topologies 436

CHAPTER 24 **MLPoE at PTA** 437
Prerequisites for MLPoE at PTA 437
Restrictions for MLPoE at PTA 437
Information About MLPoE at PTA 438
MLPoE at PTA Overview 438
How to Configure MLPoE at PTA 439
Configuring MLPoE at PTA 439
Configuring MLPoE over VLAN 440
Configuring MLPoE over QinQ 441
Configuration Examples for MLPoE at PTA 442
Example: Configuring MLPoE at PTA 442
Example: Configuring MLPoE over VLAN 442
Example: Configuring MLPoE over QinQ 443
Additional References for MLPoE at PTA 444
Feature Information for MLPoE at PTA 444

CHAPTER 25 **Configurable CHAP Challenge Length** 447
Prerequisites for Configurable CHAP Challenge Length 447
Information About Configurable CHAP Challenge Length 447
Configurable CHAP Challenge Length Overview 447
How to Configure Configurable CHAP Challenge Length 448
Configuring Configurable CHAP Challenge Length 448
Configuration Examples for Configurable CHAP Challenge Length 449
Example: Configuring Configurable CHAP Challenge Length 449
Additional References for Configurable CHAP Challenge Length 449
Feature Information for Configurable CHAP Challenge Length 450

PART VI Overlay Transport Virtualization 451

CHAPTER 26 Configuring Overlay Transport Virtualization 453

Prerequisites for OTV 453

Restrictions for OTV 454

Information About OTV 454

Functions of OTV 454

OTV Terms 455

OTV Overlay Network 456

Edge Devices 457

Site-to-Site Connectivity 457

Overlay Networks Mapping to Multicast Groups 457

OTV Packet Flow 458

Mobility 459

Sample OTV Topologies 459

OTV Features 461

Overlay Interface 462

MAC Address Learning 462

MAC Address Reachability Updates 463

Multicast Group Addresses and IGMP Snooping 463

ARP Cache 463

High Availability 463

OTV IS-IS 463

How to Configure OTV 464

Creating an Overlay Interface 464

Associating an Overlay Interface with a Physical Interface 465

Configuring a Multicast Group Address 467

Configuring a VLAN over an Overlay Interface 468

Configuring the Site Bridge Domain and the Site Identifier 469

Configuring Authentication for OTV IS-IS Hellos 471

Configuring Authentication for OTV IS-IS PDUs 472

Disabling ARP Caching 473

Tuning OTV Parameters 474

Configuration Examples for OTV Features 476
Example: Configuring Overlay Interface and VLANs 476
Verifying the OTV Configuration 484
Additional References 486
Feature Information for OTV 486

CHAPTER 27 **OTV Adjacency Server** 491
Restrictions for OTV Adjacency Server 491
Information About OTV Adjacency Server 492
Overview of a Unicast-Core Network 492
Adjacency Servers 492
Overview of an Adjacency Server 492
Functions of an Adjacency Server 492
Unicast-Only Edge Devices 493
Unicast Replication List 493
How Adjacency Servers and Edge Devices Work 493
Exclusivity Between Multicast- and Unicast-Core Networks 493
How to Configure an OTV Adjacency Server 494
Configuring an OTV Adjacency Server 494
Configuring an OTV Edge Device in a Unicast-Core Network 496
Configuration Examples for OTV Adjacency Server 497
Example: Configuring an OTV Adjacency Server 497
Example: Configuring an OTV Edge Device in a Unicast-Core Network 497
Additional References for OTV Adjacency Server 498
Feature Information for OTV Adjacency Server 498

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.

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

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

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

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. 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. (1721R)

© 2022 Cisco Systems, Inc. All rights reserved.



Preface

This preface describes the audience, organization, and conventions of this document. It also provides information on how to obtain other documentation.

This preface includes the following sections:

- Preface, on page xxvii
- Audience and Scope, on page xxvii
- Feature Compatibility, on page xxviii
- Document Conventions, on page xxviii
- Communications, Services, and Additional Information, on page xxix
- Documentation Feedback, on page xxx
- Troubleshooting, on page xxx

Preface

This preface describes the audience, organization, and conventions of this document. It also provides information on how to obtain other documentation.

This preface includes the following sections:

Audience and Scope

This document is designed for the person who is responsible for configuring your Cisco Enterprise router. This document is intended primarily for the following audiences:

- Customers with technical networking background and experience.
- System administrators familiar with the fundamentals of router-based internetworking but who might not be familiar with Cisco IOS software.
- System administrators who are responsible for installing and configuring internetworking equipment, and who are familiar with Cisco IOS software.

Feature Compatibility

For more information about the Cisco IOS XE software, including features available on your device as described in the configuration guides, see the respective router documentation set.

To verify support for specific features, use the Cisco Feature Navigator tool. This tool enables you to determine the Cisco IOS XE software images that support a specific software release, feature set, or a platform.

Document Conventions

This documentation uses the following conventions:

Convention	Description
^ or **Ctrl**	The ^ and **Ctrl** symbols represent the Control key. For example, the key combination **^D** or **Ctrl-D** means hold down the **Control** key while you press the **D** key. Keys are indicated in capital letters but are not case sensitive.
string	A string is a nonquoted set of characters shown in italics. For example, when setting an SNMP community string to public, do not use quotation marks around the string or the string will include the quotation marks.

The command syntax descriptions use the following conventions:

Convention	Description
bold	Bold text indicates commands and keywords that you enter exactly as shown.
italics	Italic text indicates arguments for which you supply values.
[x]	Square brackets enclose an optional element (keyword or argument).
|	A vertical line indicates a choice within an optional or required set of keywords or arguments.
[x | y]	Square brackets enclosing keywords or arguments separated by a vertical line indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a vertical line indicate a required choice.

Nested sets of square brackets or braces indicate optional or required choices within optional or required elements. For example, see the following table.

Convention	Description
[x {y | z}]	Braces and a vertical line within square brackets indicate a required choice within an optional element.

Examples use the following conventions:

Convention	Description
`screen`	Examples of information displayed on the screen are set in Courier font.
`bold screen`	Examples of text that you must enter are set in Courier bold font.
< >	Angle brackets enclose text that is not printed to the screen, such as passwords.
!	An exclamation point at the beginning of a line indicates a comment line. Exclamation points are also displayed by the Cisco IOS XE software for certain processes.
[ ]	Square brackets enclose default responses to system prompts.



Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.



Note Means *reader take note*. Notes contain helpful suggestions or references to materials that may not be contained in this manual.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

Documentation Feedback

To provide feedback about Cisco technical documentation, use the feedback form available in the right pane of every online document.

Troubleshooting

For the most up-to-date, detailed troubleshooting information, see the Cisco TAC website at https://www.cisco.com/en/US/support/index.html.

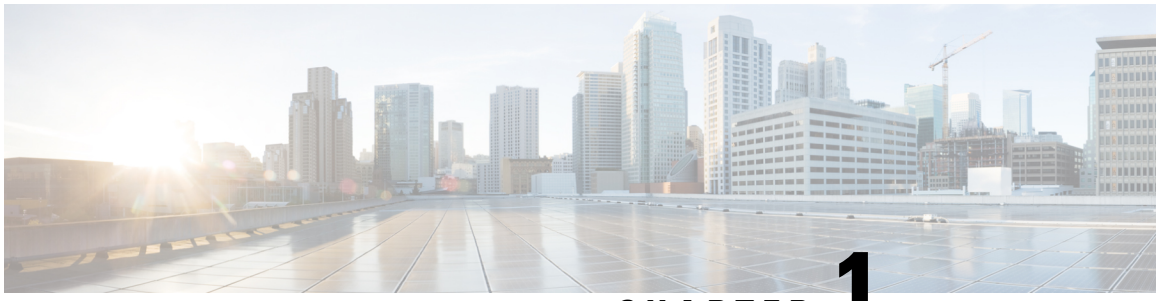
Go to **Products by Category** and choose your product from the list, or enter the name of your product. Look under **Troubleshoot and Alerts** to find information for the issue that you are experiencing.



PART I

Local Area Networking

- Configuring ERSPAN, on page 1
- Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation, on page 17
- IEEE 802.1Q-in-Q VLAN Tag Termination, on page 25
- VLAN Mapping to Gigabit EtherChannel Member Links, on page 37
- Configuring Routing Between VLANs, on page 49
- EtherChannel Flow-Based Limited 1 1 Redundancy, on page 117
- Flow-Based per Port-Channel Load Balancing, on page 127
- VLANs over IP Unnumbered SubInterfaces, on page 137
- Spanning Tree Protocol, on page 145



CHAPTER 1

Configuring ERSPAN

This module describes how to configure Encapsulated Remote Switched Port Analyzer (ERSPAN). The Cisco ERSPAN feature allows you to monitor traffic on one or more ports or VLANs and send the monitored traffic to one or more destination ports.

Note The ERSPAN feature is not supported on Layer 2 switching interfaces.

- Restrictions for Configuring ERSPAN, on page 1
- Information About Configuring ERSPAN, on page 2
- How to Configure ERSPAN, on page 5
- Configuration Examples for ERSPAN, on page 12
- Additional References for Configuring ERSPAN, on page 14
- Feature Information for Configuring ERSPAN , on page 15

Restrictions for Configuring ERSPAN

- The maximum number of allowed ERSPAN sessions on a Cisco ASR 1000 Series Router is 1024. A Cisco ASR 1000 Series Router can be used as an ERSPAN source device on which only source sessions are configured, an ERSPAN destination device on which only destination sessions are configured, or an ERSPAN source and destination device on which both source and destination sessions are configured. However, total number of sessions must not exceed 1024.
- The maximum number of available ports for each ERSPAN session is 128.
- ERSPAN on Cisco ASR 1000 Series Routers supports only Fast Ethernet, Gigabit Ethernet, TenGigabit Ethernet, and port-channel interfaces as source ports for a source session.
- ERSPAN on Cisco ASR 1000 Series Routers supports only Layer 3 interfaces. Ethernet interfaces are not supported on ERSPAN when configured as Layer 2 interfaces.
- ERSPAN users on Cisco ASR 1000 Series Routers can configure a list of ports as a source or a list of VLANs as a source, but cannot configure both for a given session.
- When a session is configured through the ERSPAN configuration CLI, the session ID and the session type cannot be changed. To change them, you must first use the **no** form of the configuration command to remove the session and then reconfigure the session.

- The **monitor session** *span-session-number* **type local** command is not supported on Cisco ASR 1000 Series Routers.
- The filter VLAN option is not functional in an ERSPAN monitoring session on WAN interfaces.

Information About Configuring ERSPAN

ERSPAN Overview

The Cisco ERSPAN feature allows you to monitor traffic on one or more ports or more VLANs, and send the monitored traffic to one or more destination ports. ERSPAN sends traffic to a network analyzer such as a Switch Probe device or other Remote Monitoring (RMON) probe. ERSPAN supports source ports, source VLANs, and destination ports on different routers, which provides remote monitoring of multiple routers across a network (see the figure below).

On a Cisco ASR 1000 Series Router, ERSPAN supports encapsulated packets of up to 9180 bytes. The default ERSPAN maximum transmission unit (MTU) size is 1500 bytes. If the ERSPAN payload length, which comprises the encapsulated IPv4 header, generic routing encapsulation (GRE) header, ERSPAN header, and the original packet, exceeds the ERSPAN MTU size, the replicated packet is truncated to the default ERSPAN MTU size.

ERSPAN consists of an ERSPAN source session, routable ERSPAN GRE encapsulated traffic, and an ERSPAN destination session.

You can configure an ERSPAN source session, an ERSPAN destination session, or both on a Cisco ASR 1000 Series Router. A device that has only an ERSPAN source session configured is called an ERSPAN source device, and a device that has only an ERSPAN destination session configured is called an ERSPAN termination device. A Cisco ASR 1000 Series Router can act as both an ERSPAN source device and an ERSPAN termination device. You can terminate an ERSPAN session with a destination session on the same Cisco ASR 1000 Series Router.

An ERSPAN source session is defined by the following parameters:

- A session ID
- List of source ports or source VLANs to be monitored by the session
- The destination and origin IP addresses, which are used as the destination and source IP addresses of the GRE envelope for the captured traffic, respectively
- ERSPAN flow ID
- Optional attributes, such as, IP type of service (TOS) and IP Time to Live (TTL), related to the GRE envelope

An ERSPAN destination session is defined by the following:

- Session ID
- Destination ports
- Source IP address, which is the same as the destination IP address of the corresponding source session
- ERSPAN flow ID, which is used to match the destination session with the source session

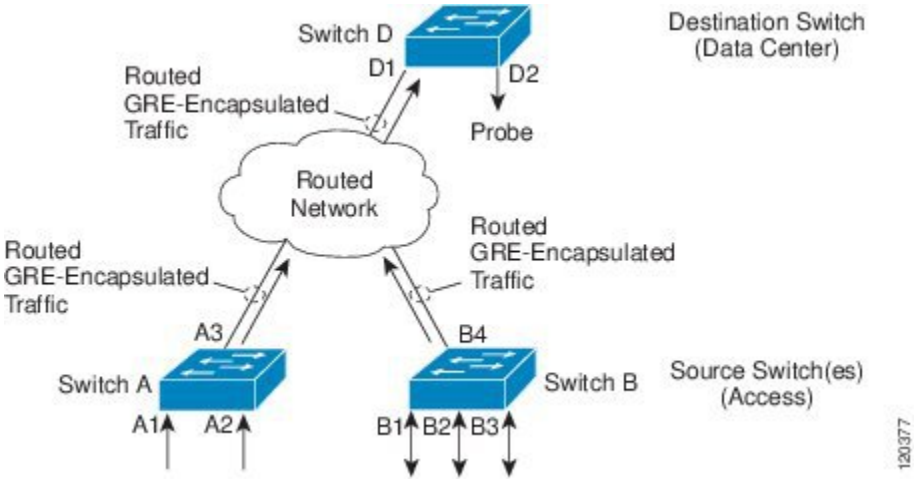
ERSPAN source sessions do not copy ERSPAN GRE-encapsulated traffic from source ports. Each ERSPAN source session can have either ports or VLANs as sources, but not both.

The ERSPAN source sessions copy traffic from the source ports or source VLANs and forwards the traffic using routable GRE-encapsulated packets to the ERSPAN destination session. The ERSPAN destination session switches the traffic to the destination ports.

Note When there is a change in the routing topology, the routing path for the ERSPAN destination could also change. If the egress bandwidth is not sufficient for ERSPAN traffic, the excess traffic is dropped.

If the specific route for the ERSPAN destination is not available in the routing table and there is a default route set, the ERSPAN traffic is sent via the default route.

Figure 1: ERSPAN Configuration



Monitored Traffic

For a source port or a source VLAN, the ERSPAN can monitor the ingress, egress, or both ingress and egress traffic. By default, ERSPAN monitors all traffic, including multicast and Bridge Protocol Data Unit (BPDU) frames.

ERSPAN Sources

The Cisco ERSPAN feature supports the following sources:

- Source ports—A source port that is monitored for traffic analysis. Source ports in any VLAN can be configured and trunk ports can be configured as source ports along with nontrunk source ports.
- Source VLANs—A VLAN that is monitored for traffic analysis.

The following tunnel interfaces are supported as source ports for a ERSPAN source session:

- GRE
- IPinIP
- IPv6
- IPv6 over IP tunnel

- Multipoint GRE (mGRE)
- Secure Virtual Tunnel Interfaces (SVTI)



Note SVTI and IPinIP tunnel interfaces support the monitoring of both IPsec-protected and non-IPsec-protected tunnel packets. Monitoring of tunnel packets allows you to see the clear-text tunnel packet after IPsec decryption if that tunnel is IPsec protected.

The following limitations apply to the enhancements introduced in Cisco IOS XE Release 3.4S:

- Monitoring of non-IPsec-protected tunnel packets is supported on IPv6 and IPv6 over IP tunnel interfaces.
- The enhancements apply only to ERSPAN source sessions, not to ERSPAN destination sessions.

ERSPAN has the following behavior in Cisco IOS XE Release 3.4S:

- The tunnel interface is removed from the ERSPAN database at all levels when the tunnel interface is deleted. If you want to create the same tunnel again, you must manually configure it in source monitor sessions to keep monitoring the tunnel traffic.
- The Layer 2 Ethernet header is generated with both source and destination MAC addresses set to zero.

In Cisco IOS XE Release 3.5S, support was added for the following types of WAN interfaces as source ports for a source session:

- Serial (T1/E1, T3/E3, DS0)
- Packet over SONET (POS) (OC3, OC12)
- Multilink PPP
- The **multilink**, **pos**, and **serial** keywords were added to the **source interface** command.

ERSPAN Destination Ports

A destination port is a Layer 2 or Layer 3 LAN port to which ERSPAN sends traffic for analysis.

When you configure a port as a destination port, it can no longer receive any traffic and, the port is dedicated for use only by the ERSPAN feature. An ERSPAN destination port does not forward any traffic except that required for the ERSPAN session. You can configure trunk ports as destination ports, which allows destination trunk ports to transmit encapsulated traffic.

Using ERSPAN as Local SPAN

To use ERSPAN to monitor traffic through one or more ports or VLANs, you must create an ERSPAN source and ERSPAN destination sessions.

You can create the two sessions either on the same router or on different routers. If the two sessions are created on two different routers, the monitoring traffic will be forwarded from the source to the destination by ERSPAN. However, if the two sessions are created on the same router, data flow takes place inside the router, which is similar to that in local SPAN.

The following factors are applicable while using ERSPAN as a local SPAN:

- Both sessions have the same ERSPAN ID.

- Both sessions have the same IP address. This IP address is the router's own IP address; that is, the loopback IP address or the IP address configured on any port.

ERSPAN Support on WAN Interface

In Cisco IOS Release 3.5S an ERSPAN source on WAN is added to allow monitoring of traffic on WAN interfaces. ERSPAN replicates the original frame and encapsulates the replicated frame inside an IP or GRE packet by adding Fabric Interface ASIC (FIA) entries on the WAN interface. The frame header of the replicated packet is modified for capturing. After encapsulation, ERSPAN sends the IP or GRE packet through an IP network to a device on the network. This device sends the original frame to an analyzing device that is directly connected to the network device.

ERSPAN Dummy MAC Address Rewrite

ERSPAN dummy MAC address rewrite supports customized MAC value for WAN interface and tunnel interface. It also allows you to monitor the traffic going through WAN interface.

ERSPAN IP Access Control Lists

From Cisco IOS XE Everest 16.4.1 release, ERSPAN has been enhanced to better monitor packets and reduce network traffic. This enhancement supports ACL on ERSPAN source session to filter only specific IP traffic according to the ACL, and is supported on the IOS XE platform. Both IPv4 and IPv6 traffic can be monitored by associating an ACL with the ERSPAN session. The ERSPAN session can associate only one IP ACL entry with its name.

How to Configure ERSPAN

ERSPAN uses separate source and destination sessions. You configure the source and destination sessions on either the same router or on different routers.

Configuring an ERSPAN Source Session

The ERSPAN source session defines the session configuration parameters and the ports or VLANs to be monitored.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-type interface-number*
4. **plim ethernet vlan filter disable**
5. **monitor session** *span-session-number* **type erspan-source**
6. **description** *string*
7. [**no**] **header-type 3**
8. **source interface** *interface-name interface-number*
9. **source vlan** {*id-single* | *id-list* | *id-range* | *id-mixed*} [**rx** | **tx** | **both**]

10. **filter vlan** {*id-single* | *id-list* | *id-range* | *id-mixed*}
11. **filter access-group** *acl-filter*
12. **destination**
13. **erspan-id** *erspan-flow-id*
14. **ip address** *ip-address*
15. **ip prec** *prec-value*
16. **ip dscp** *dscp-value*
17. **ip ttl** *ttl-value*
18. **mtu** *mtu-size*
19. **origin ip address** *ip-address* [**force**]
20. **vrf** *vrf-id*
21. **no shutdown**
22. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-type interface-number* **Example:** `Device(config)# interface GigabitEthernet1/0/1`	Specifies the interface on which ERSPAN source session is configured.
Step 4	**plim ethernet vlan filter disable** **Example:** `Device(config-if)# plim ethernet vlan filter disable`	(Optional) Disables the VLAN filtering option for Ethernet interfaces. Use this command if you are using the **vlan filter** command or if the source interface is using dot1q encapsulation.
Step 5	**monitor session** *span-session-number* **type erspan-source** **Example:** `Device(config)# monitor session 1 type erspan-source`	Defines an ERSPAN source session using the session ID and the session type, and enters ERSPAN monitor source session configuration mode. • The *span-session-number* argument range is from 1 to 1024. The same session number cannot be used more than once. • The session IDs for source sessions or destination sessions are in the same global ID space, so each session ID is globally unique for both session types.

	Command or Action	Purpose
		• The session ID (configured by the *span-session-number* argument) and the session type (configured by the **erspan-source** keyword) cannot be changed once entered. Use the **no** form of this command to remove the session and then re-create the session, with a new session ID or a new session type.
Step 6	**description** *string* **Example:** `Device(config-mon-erspan-src)# description source1`	(Optional) Describes the ERSPAN source session. • The *string* argument can be up to 240 characters and cannot contain special characters or spaces.
Step 7	[**no**] **header-type 3** **Example:** `Device(config-mon-erspan-src)# header-type 3`	Configures a switch to ERSPAN header type III.
Step 8	**source interface** *interface-name interface-number* **Example:** `Device(config-mon-erspan-src)# source interface GigabitEthernet1/0/1 rx`	Configures more than one WAN interface in a single ERSPAN session.
Step 9	**source vlan** {*id-single* | *id-list* | *id-range* | *id-mixed*} [**rx** | **tx** | **both**] **Example:** `Device(config-mon-erspan-src)# source vlan 1`	(Optional) Associates the ERSPAN source session number with the VLANs, and selects the traffic direction to be monitored. • You cannot include source VLANs and filter VLANs in the same session. You can either include source VLANs or filter VLANs, but not both at the same time.
Step 10	**filter vlan** {*id-single* | *id-list* | *id-range* | *id-mixed*} **Example:** `Device(config-mon-erspan-src)# filter vlan 1`	(Optional) Configures source VLAN filtering when the ERSPAN source is a trunk port. • You cannot include source VLANs and filter VLANs in the same session. You can have source VLANs or filter VLANs, but not both at the same time.
Step 11	**filter access-group** *acl-filter* **Example:** `Device(config-mon-erspan-src)# filter access-group ACL1`	(Optional) Associates an ACL with the ERSPAN session. • Use the **no filter access-group** *acl-filter* command to detach the ACL from the ERSPAN session. • Only ACL name is supported to associate to the ERSPAN source session. If the ACL does not exist or if there is no entry defined in the access control list, the ACL name is not attached to the ERSPAN source session. • When the ERSPAN source session is active, you cannot detach the ACL from the ERSPAN source

	Command or Action	Purpose
		session. The source session must be shut down before detaching the ACL. After the session shutdown, you must exit the session for the **shutdown** command to execute, and then re-enter the session to detach the ACL.
Step 12	**destination** **Example:** `Device(config-mon-erspan-src)# destination`	Enters ERSPAN source session destination configuration mode.
Step 13	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-src-dst)# erspan-id 100`	Configures the ID used by the source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN destination session configuration.
Step 14	**ip address** *ip-address* **Example:** `Device(config-mon-erspan-src-dst)# ip address 10.10.0.1`	Configures the IP address that is used as the destination of the ERSPAN traffic.
Step 15	**ip prec** *prec-value* **Example:** `Device(config-mon-erspan-src-dst)# ip prec 5`	(Optional) Configures the IP precedence value of the packets in the ERSPAN traffic. • You can optionally use either the **ip prec** command or the **ip dscp** command, but not both.
Step 16	**ip dscp** *dscp-value* **Example:** `Device(config-mon-erspan-src-dst)# ip dscp 10`	(Optional) Enables the use of IP differentiated services code point (DSCP) for packets that originate from a circuit emulation (CEM) channel. • You can optionally use either the **ip prec** command or the **ip dscp** command, but not both.
Step 17	**ip ttl** *ttl-value* **Example:** `Device(config-mon-erspan-src-dst)# ip ttl 32`	(Optional) Configures the IP TTL value of the packets in the ERSPAN traffic.
Step 18	**mtu** *mtu-size* **Example:** `Device(config-mon-erspan-src-dst)# mtu 1500`	Configures the maximum transmission unit (MTU) size, in bytes, for ERSPAN encapsulation. • Valid values are from 64 to 9180. The default value is 1500.
Step 19	**origin ip address** *ip-address* [**force**] **Example:** `Device(config-mon-erspan-src-dst)# origin ip address 10.10.0.1`	Configures the IP address used as the source of the ERSPAN traffic.

	Command or Action	Purpose
Step 20	**vrf** *vrf-id* **Example:** `Device(config-mon-erspan-src-dst)# vrf 1`	(Optional) Configures the VRF name to use instead of the global routing table.
Step 21	**no shutdown** **Example:** `Device(config-mon-erspan-src-dst)# no shutdown`	Enables the configured sessions on an interface.
Step 22	**end** **Example:** `Device(config-mon-erspan-src-dst)# end`	Exits ERSPAN source session destination configuration mode, and returns to privileged EXEC mode.

Configuring an ERSPAN Destination Session

Perform this task to configure an Encapsulated Remote Switched Port Analyzer (ERSPAN) destination session. The ERSPAN destination session defines the session configuration parameters and the ports that will receive the monitored traffic.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **monitor session** *session-number* **type erspan-destination**
4. **description** *string*
5. **destination interface** {**gigabitethernet** | **port-channel**} [*interface-number*]
6. **source**
7. **erspan-id** *erspan-flow-id*
8. **ip address** *ip-address* [**force**]
9. **vrf** *vrf-id*
10. **no shutdown**
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**monitor session** *session-number* **type erspan-destination** **Example:** `Device(config)# monitor session 1 type erspan-destination`	Defines an ERSPAN destination session using the session ID and the session type, and enters in ERSPAN monitor destination session configuration mode. • The *session-number* argument range is from 1 to 1024. The session number must be unique and cannot be used more than once. • The session IDs for source sessions or destination sessions are in the same global ID space, so each session ID is globally unique for both session types. • The session ID (configured by the *session-number* argument) and the session type (configured by the **erspan-destination**) cannot be changed once entered. Use the **no** form of this command to remove the session, and then recreate the session with a new session ID or a new session type.
Step 4	**description** *string* **Example:** `Device(config-mon-erspan-dst)# description source1`	(Optional) Describes the ERSPAN destination session. • The *string* argument can be up to 240 characters in length and cannot contain special characters or spaces.
Step 5	**destination interface** {**gigabitethernet** | **port-channel**} [*interface-number*] **Example:** `Device(config-mon-erspan-dst)# destination interface GigabitEthernet1/0/1`	Associates the ERSPAN destination session number with the source ports, and selects the traffic direction to be monitored.
Step 6	**source** **Example:** `Device(config-mon-erspan-dst)# source`	Enters ERSPAN destination session source configuration mode.
Step 7	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-dst-src)# erspan-id 100`	Configures the ID used by the source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN source session configuration.
Step 8	**ip address** *ip-address* [**force**] **Example:** `Device(config-mon-erspan-dst-src)# ip address 10.10.0.1`	Configures the IP address that is used as the source of the ERSPAN traffic. • The **ip address** *ip-address* **force** command changes the source IP address for all ERSPAN destination sessions.
Step 9	**vrf** *vrf-id* **Example:**	(Optional) Configures the VRF name to use instead of the global routing table.

	Command or Action	Purpose
	`Device(config-mon-erspan-dst-src)# vrf 1`	
Step 10	**no shutdown** **Example:** `Device(config-mon-erspan-dst-src)# no shutdown`	Enables the configured sessions on an interface.
Step 11	**end** **Example:** `Device(config-mon-erspan-dst-src)# end`	Exits ERSPAN destination session source configuration mode, and returns to privileged EXEC mode.

Configuring ERSPAN Dummy MAC Address Rewrite

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **monitor session** *span-session-number* **type erspan-source**
4. **source interface** *interface-name interface-number*
5. **s-mac** *address*
6. **d-mac** *address*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**monitor session** *span-session-number* **type erspan-source** **Example:** `Device(config)# monitor session 100 type erspan-source`	Defines an ERSPAN source session using the session ID and the session type, and enters ERSPAN monitor source session configuration mode. • The *span-session-number* argument range is from 1 to 1024. The same session number cannot be used more than once.

	Command or Action	Purpose
		• The session IDs for source sessions or destination sessions are in the same global ID space, so each session ID is globally unique for both session types. • The session ID (configured by the *span-session-number* argument) and the session type (configured by the **erspan-source** keyword) cannot be changed once entered. Use the **no** form of this command to remove the session and then re-create the session, with a new session ID or a new session type.
Step 4	**source interface** *interface-name interface-number* **Example:** `Device(config-mon-erspan-src)# source interface GigabitEthernet1/0/1 rx`	Configures more than one WAN interface in a single ERSPAN session.
Step 5	**s-mac** *address* **Example:** `Device(config-mon-erspan-src)# s-mac 1111.1111.1111`	Defines source pseudo mac for wan interface.
Step 6	**d-mac** *address* **Example:** `Device(config-mon-erspan-src)# d-mac 2222.2222.2222`	Defines destination pseudo mac for wan interface.
Step 7	**end** **Example:** `Device(config-mon-erspan-src)# end`	Exits ERSPAN source session destination configuration mode, and returns to privileged EXEC mode.

Configuration Examples for ERSPAN

Example: Configuring an ERSPAN Source Session

The following example shows how to configure an ERSPAN source session:

```
Device> enable
Device# configure terminal
Device(config)# monitor session 1 type erspan-source
Device(config-mon-erspan-src)# description source1
Device(config-mon-erspan-src)# source interface GigabitEthernet1/0/1 rx
Device(config-mon-erspan-src)# source interface GigabitEthernet1/0/4 - 8 tx
Device(config-mon-erspan-src)# source interface GigabitEthernet1/0/3
Device(config-mon-erspan-src)# destination
Device(config-mon-erspan-src-dst)# erspan-id 100
Device(config-mon-erspan-src-dst)# origin ip address 10.1.0.1
Device(config-mon-erspan-src-dst)# ip prec 5
Device(config-mon-erspan-src-dst)# ip ttl 32
Device(config-mon-erspan-src-dst)# mtu 1700
Device(config-mon-erspan-src-dst)# origin ip address 10.10.0.1
```

```
Device(config-mon-erspan-src-dst)# vrf 1
Device(config-mon-erspan-src-dst)# no shutdown
Device(config-mon-erspan-src-dst)# end
```

Example: Configuring an ERSPAN Source Session on a WAN Interface

The following example shows how to configure more than one WAN interface in a single ERSPAN source monitor session. Multiple interfaces have been separated by a commas.

```
monitor session 100 type erspan-source
    source interface Serial 0/1/0:0, Serial 0/1/0:6
```

Example: Configuring an ERSPAN Destination Session

The following example shows how to configure an ERSPAN destination session:

```
monitor session 2 type erspan-destination
 destination interface GigabitEthernet1/3/2
 destination interface GigabitEthernet2/2/0
 source
  erspan-id 100
  ip address 10.10.0.1
```

Example: Configuring an ERSPAN as a Local SPAN

The following example shows how to configure an ERSPAN as a local SPAN.

```
monitor session 10 type erspan-source
 source interface GigabitEthernet0/0/0
 destination
 erspan-id 10
 ip address 10.10.10.1
 origin ip address 10.10.10.1
monitor session 20 type erspan-destination
destination interface GigabitEthernet0/0/1
 source
  erspan-id 10
  ip address 10.10.0.1
```

Example: Configuring ERSPAN Dummy MAC Address Rewrite

```
monitor session 1 type erspan-source
 s-mac 1111.1111.1111
 d-mac 2222.2222.2222
 source interface Gi2/2/0
 destination
  erspan-id 100
  mtu 1464
  ip address 200.0.0.1
  origin ip address 100.0.0.1
```

Example: Configuring UDF-Based ERSPAN

This example shows how to configure UDF-based ERSPAN to match on the inner TCP flags of an encapsulated IP-in-IP packet using the following match criteria:

- Outer source IP address: 10.0.0.2
- Inner TCP flags: Urgent TCP flag is set
- Bytes: Eth Hdr (14) + Outer IP (20) + Inner IP (20) + Inner TCP (20, but TCP flags at 13th byte)
- Offset from packet-start: 14 + 20 + 20 + 13 = 67
- UDF match value: 0x20 • UDF mask: 0xFF

```
udf udf_tcpflags packet-start 67 1
ip access-list acl-udf
permit ip 10.0.0.2/32 any udf udf_tcpflags 0x20 0xff
monitor session 1 type erspan-source
source interface Ethernet 1/1
filter access-group acl-udf
```

This example shows how to configure UDF-based ERSPAN to match regular IP packets with a packet signature (DEADBEEF) at 6 bytes after a Layer 4 header start using the following match criteria:

- Outer source IP address: 10.0.0.2
- Inner TCP flags: Urgent TCP flag is set
- Bytes: Eth Hdr (14) + IP (20) + TCP (20) + Payload: 112233445566DEADBEEF7788
- Offset from Layer 4 header start: 20 + 6 = 26
- UDF match value: 0xDEADBEEF (split into two-byte chunks and two UDFs)
- UDF mask: 0xFFFFFFFF

```
udf udf_pktsig_msb header outer l3 26 2
udf udf_pktsig_lsb header outer l3 28 2
ip access-list acl-udf-pktsig
permit udf udf_pktsig_msb 0xDEAD 0xFFFF udf udf_pktsig_lsb 0xBEEF 0xFFFF
monitor session 1 type erspan-source
source interface Ethernet 1/1
filter access-group acl-udf-pktsig
```

Additional References for Configuring ERSPAN

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
LAN Switching commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	LAN Switching Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Configuring ERSPAN

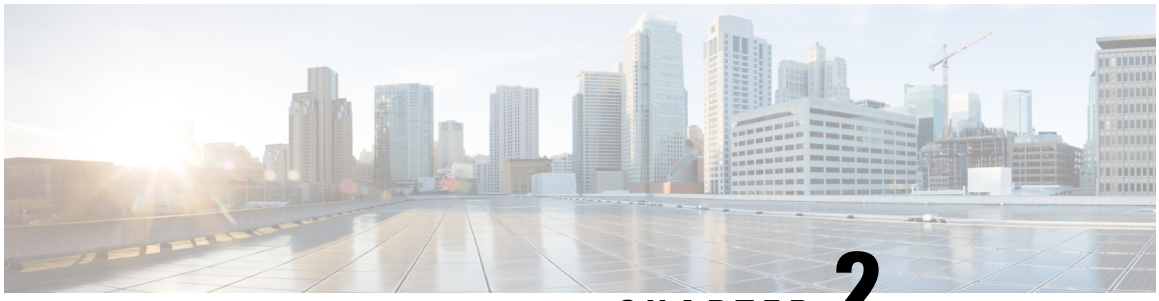
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for Configuring ERSPAN

Feature Name	Releases	Feature Information
ERSPAN	Cisco IOS XE Release 2.1 Cisco IOS XE Release 3.8S	The Cisco ERSPAN feature allows you to monitor traffic on one or more ports or VLANs, and send the monitored traffic to one or more destination ports. The following commands were introduced or modified by this feature: **description**, **destination**, **erspan-id**, **filter**, **ip dscp**, **ip prec**, **ip ttl, monitor permit-list**, **monitor session**, **origin ip address**, **show monitor permit-list**, **source**, **switchport**, **switchport mode trunk**, **switchport nonegotiate**, **switchport trunk encapsulation**, **vrf**. In Cisco IOS XE 3.8S release, ERSPAN was enhanced to support MTU data size up to 9180 bytes. The following command was added by this feature: **mtu**.
ERSPAN Support on WAN Interface	Cisco IOS XE Release 3.5S	ERSPAN has been enhanced to support WAN interface as an ERSPAN source. The following command was modified by this feature: **source interface**.
ERSPAN Type III Header	Cisco IOS XE Denali 16.2	ERSPAN has been enhanced to configure a switch to ERSPAN type III header. The following command was introduced by this feature: **header-type 3**.

Feature Name	Releases	Feature Information
ERSPAN IP ACL	Cisco IOS XE Everest 16.4.1	ERSPAN has been enhanced to better monitor packets and reduce network traffic. This enhancement supports ACL on ERSPAN source session to filter only specific IP traffic according to the ACL. The following command was introduced by this feature: **filter access-group** *acl-filter*.



CHAPTER 2

Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

This chapter describes the required and optional tasks for configuring routing between VLANs with IEEE 802.1Q encapsulation.

- Restrictions for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation, on page 17
- Information About Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation, on page 17
- How to Configure Routing Between VLANs with IEEE 802.1Q Encapsulation, on page 18
- Configuration Examples for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation, on page 22
- Additional References, on page 22
- Feature Information for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation, on page 23

Restrictions for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

Shared port adapters (SPAs) on Cisco ASR 1000 Series Aggregation Services Router have a limit of 8,000 TCAM entries, which limits the number of VLANs you can create on a single SPA.

Information About Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

The IEEE 802.1Q protocol is used to interconnect multiple switches and routers, and for defining VLAN topologies. The IEEE 802.1Q standard is extremely restrictive to untagged frames. The standard provides only a per-port VLANs solution for untagged frames. For example, assigning untagged frames to VLANs takes into consideration only the port from which they have been received. Each port has a parameter called

a *permanent virtual identification* (Native VLAN) that specifies the VLAN assigned to receive untagged frames.

The main characteristics of IEEE 802.1Q are as follows:

- Assigns frames to VLANs by filtering.
- The standard assumes the presence of a single spanning tree and of an explicit tagging scheme with one-level tagging.

How to Configure Routing Between VLANs with IEEE 802.1Q Encapsulation

Configuring IP Routing over IEEE 802.1Q

IP routing over IEEE 802.1Q extends IP routing capabilities to include support for routing IP frame types in VLAN configurations using the IEEE 802.1Q encapsulation.

To route IP over IEEE 802.1Q between VLANs, you need to customize the subinterface to create the environment in which it will be used. Perform the tasks described in the following sections in the order in which they appear:

Enabling IP Routing

IP routing is automatically enabled in the Cisco IOS XE software for routers. To reenable IP routing if it has been disabled, perform the following steps.

Once you have IP routing enabled on the router, you can customize the characteristics to suit your environment. If necessary, refer to the IP configuration chapters in the *Cisco IOS XE IP Routing Protocols Configuration Guide* , Release 2, for guidelines on configuring IP.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Router(config)# ip routing`	Enables IP routing on the router.
Step 4	**end** **Example:** `Router(config)# exit`	Exits privileged EXEC mode.

Defining the VLAN Encapsulation Format

To define the encapsulation format as IEEE 802.1Q, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *card* / *spaslot* / *port* **.** *subinterface-number*
4. **encapsulation dot1q** *vlanid*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *card* / *spaslot* / *port* **.** *subinterface-number* **Example:**	Specifies the subinterface on which IEEE 802.1Q will be used, and enters interface configuration mode.

	Command or Action	Purpose
	`Router(config)# interface gigabitethernet 0/0/0.101`	
Step 4	**encapsulation dot1q** *vlanid* **Example:** `Router(config-subif)# encapsulation dot1q 101`	Defines the encapsulation format as IEEE 802.1Q (**dot1q**), and specifies the VLAN identifier
Step 5	**end** **Example:** `Router(config-subif)# end`	Exits subinterface configuration mode.

Assigning an IP Address to Network Interface

An interface can have one primary IP address. To assign a primary IP address and a network mask to a network interface, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *card* / *spaslot* / *port* **.** *subinterface-number*
4. **ip address** *ip-address mask*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *card* / *spaslot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface gigabitethernet 0/0/0.101`	Specifies the subinterface on which IEEE 802.1Q will be used, and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.0.0.0 255.0.0.0`	Sets a primary IP address for an interface. • Enter the primary IP address for an interface. **Note** A mask identifies the bits that denote the network number in an IP address. When you use the mask to subnet a network, the mask is then referred to as a subnet mask.
Step 5	**end** **Example:** `Router(config-subif)# end`	Exits subinterface configuration mode.

Monitoring and Maintaining VLAN Subinterfaces

To indicate whether a VLAN is a native VLAN, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **show vlans**
3. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show vlans** **Example:** `Router# show vlans`	Displays VLAN information.
Step 3	**end** **Example:** `Router# end`	Exits privileged EXEC mode.

Configuration Examples for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

Configuring IP Routing over IEEE 802.1Q Example

This configuration example shows IP being routed on VLAN 101:

```
!
ip routing
!
interface gigabitethernet 4/1/1.101
  encapsulation dot1q 101
  ip addr 10.0.0.0 255.0.0.0
!
```

Additional References

Related Documents

Related Topic	Document Title
IP LAN switching commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS LAN Switching Services Command Reference

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing standards has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

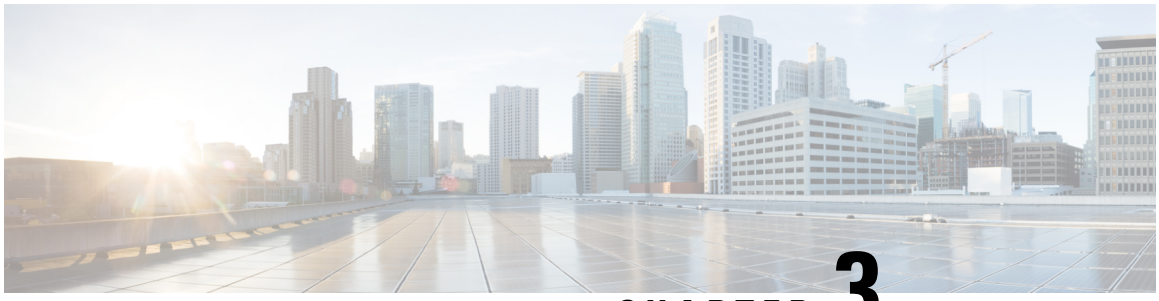
Feature Information for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

Feature Name	Releases	Feature Information
Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation	Cisco IOS XE Release 2.1	This feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers.



CHAPTER 3

IEEE 802.1Q-in-Q VLAN Tag Termination

Encapsulating IEEE 802.1Q VLAN tags within 802.1Q enables service providers to use a single VLAN to support customers who have multiple VLANs. The IEEE 802.1Q-in-Q VLAN Tag Termination feature on the subinterface level preserves VLAN IDs and keeps traffic in different customer VLANs segregated.

- Information About IEEE 802.1Q-in-Q VLAN Tag Termination, on page 25
- How to Configure IEEE 802.1Q-in-Q VLAN Tag Termination, on page 27
- Configuration Examples for IEEE 802.1Q-in-Q VLAN Tag Termination, on page 32
- Additional References, on page 34
- Feature Information for IEEE 802.1Q-in-Q VLAN Tag Termination, on page 35

Information About IEEE 802.1Q-in-Q VLAN Tag Termination

IEEE 802.1Q-in-Q VLAN Tag Termination on Subinterfaces

IEEE 802.1Q-in-Q VLAN Tag Termination simply adds another layer of IEEE 802.1Q tag (called “metro tag” or “PE-VLAN”) to the 802.1Q tagged packets that enter the network. The purpose is to expand the VLAN space by tagging the tagged packets, thus producing a “double-tagged” frame. The expanded VLAN space allows the service provider to provide certain services, such as Internet access on specific VLANs for specific customers, and yet still allows the service provider to provide other types of services for their other customers on other VLANs.

Generally the service provider’s customers require a range of VLANs to handle multiple applications. Service providers can allow their customers to use this feature to safely assign their own VLAN IDs on subinterfaces because these subinterface VLAN IDs are encapsulated within a service-provider designated VLAN ID for that customer. Therefore there is no overlap of VLAN IDs among customers, nor does traffic from different customers become mixed. The double-tagged frame is “terminated” or assigned on a subinterface with an expanded **encapsulation dot1q** command that specifies the two VLAN ID tags (outer VLAN ID and inner VLAN ID) terminated on the subinterface (see the figure below).

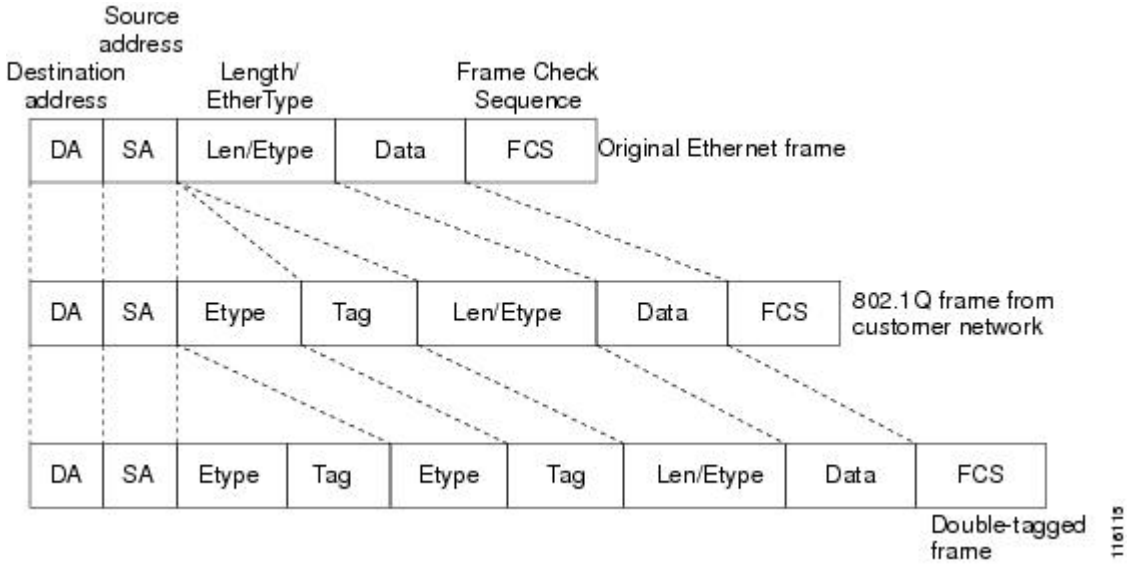
IEEE 802.1Q-in-Q VLAN Tag Termination is generally supported on whichever Cisco IOS XE features or protocols are supported on the subinterface. The only restriction is whether you assign ambiguous or unambiguous subinterfaces for the inner VLAN ID. See the Unambiguous and Ambiguous Subinterfaces section.

The primary benefit for the service provider is reduced number of VLANs supported for the same number of customers. Other benefits of this feature include:

- PPPoE scalability. By expanding the available VLAN space from 4096 to approximately 16.8 million (4096 times 4096), the number of PPPoE sessions that can be terminated on a given interface is multiplied.
- When deploying Gigabyte Ethernet DSL Access Multiplexer (DSLAM) in wholesale model, you can assign the inner VLAN ID to represent the end-customer virtual circuit (VC) and assign the outer VLAN ID to represent the service provider ID.

Whereas switches require IEEE 802.1Q tunnels on interfaces to carry double-tagged traffic, routers need only encapsulate Q-in-Q VLAN tags within another level of 802.1Q tags in order for the packets to arrive at the correct destination.

Figure 2: Untagged, 802.1Q-Tagged, and Double-Tagged Ethernet Frames



Unambiguous and Ambiguous Subinterfaces

The **encapsulation dot1q** command is used to configure Q-in-Q termination on a subinterface. The command accepts an Outer VLAN ID and one or more Inner VLAN IDs. The outer VLAN ID always has a specific value, while inner VLAN ID can either be a specific value or a range of values.

A subinterface that is configured with a single Inner VLAN ID is called an unambiguous Q-in-Q subinterface. In the following example, Q-in-Q traffic with an Outer VLAN ID of 101 and an Inner VLAN ID of 1001 is mapped to the Gigabit Ethernet 1/1/0.100 subinterface:

```
Device(config)# interface gigabitEehernet1/1/0.100
Device(config-subif)# encapsulation dot1q 101 second-dot1q 1001
```

A subinterface that is configured with multiple Inner VLAN IDs is called an ambiguous Q-in-Q subinterface. By allowing multiple Inner VLAN IDs to be grouped together, ambiguous Q-in-Q subinterfaces allow for a smaller configuration, improved memory usage and better scalability.

In the following example, Q-in-Q traffic with an Outer VLAN ID of 101 and Inner VLAN IDs anywhere in the 2001-2100 and 3001-3100 range is mapped to the Gigabit Ethernet 1/1/0.101 subinterface:

```
Device(config)# interface gigabitethernet1/1/0.101
Device(config-subif)# encapsulation dot1q 101 second-dot1q 2001-2100,3001-3100
```

Ambiguous subinterfaces can also use the **any**keyword to specify the inner VLAN ID.

See the Configuration Examples for IEEE 802.1Q-in-Q VLAN Tag Termination section for an example of how VLAN IDs are assigned to subinterfaces, and for a detailed example of how the **any** keyword is used on ambiguous subinterfaces.

Only PPPoE is supported on ambiguous subinterfaces. Standard IP routing is not supported on ambiguous subinterfaces.

IEEE802.1ad Support in Port-channels and Subinterfaces

This enhancement introduces IEEE802.1ad support on port-channel, port-channel subinterfaces, and port-channel member links, with EtherType 0x88a8, 0x9100, and 0x9200, in addition to the existing EtherType 0x8100.

Run the **dot1q tunneling ethertype xxxx** command to configure IEEE802.1ad on port-channel, port-channel subinterface, and port-channel member links.

This configuration is supported on the following platforms:

- Cisco ASR1006-X
- Cisco ASR1009-X: dual RP3, dual ESP 200, dual ESP 200-X, MIP100, and EPA (1X100GE, 10X10GE)
- Cisco ASR 1000 fixed routers

High availability is configured on ASR1006-X, ASR1009-X, and SSO/ISSU.

Restrictions for IEEE802.1ad Support in Port-channels and Subinterfaces

- This feature is not supported with SIP and SPA cards.
- Dot1q tunnel ethertype configuration is not supported in subinterfaces with high-availability configuration.

How to Configure IEEE 802.1Q-in-Q VLAN Tag Termination

Configuring the Interfaces for IEEE 802.1Q-in-Q VLAN Tag Termination

Perform this task to configure the main interface used for the Q-in-Q double tagging and to configure the subinterfaces. An optional step in this task shows you how to configure the EtherType field to be 0x9100 for the outer VLAN tag, if that is required. After the subinterface is defined, the 802.1Q encapsulation is configured to use the double tagging.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **dot1q tunneling ethertype** *ethertype*
5. **interface** *type number* **.** *subinterface-number*
6. **encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id* | *vlan-id* - *vlan-id* [*vlan-id* - *vlan-id*]}

7. **pppoe enable** [**group** *group-name*] [**max-sessions** *max-sessions-number*]
8. **exit**
9. Repeat Step 5 to configure another subinterface.
10. Repeat Step 6 and Step 7 to specify the VLAN tags to be terminated on the subinterface.
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/0/0`	Configures an interface and enters interface configuration mode.
Step 4	**dot1q tunneling ethertype** *ethertype* **Example:** `Device(config-if)# dot1q tunneling ethertype 0x9100`	(Optional) Defines the Ethertype field type used by peer devices when implementing Q-in-Q VLAN tagging.
Step 5	**interface** *type number* **.** *subinterface-number* **Example:** `Device(config-if)# interface gigabitethernet 1/0/0.1`	Configures a subinterface and enters subinterface configuration mode.
Step 6	**encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id* | *vlan-id* - *vlan-id* [*vlan-id* - *vlan-id*]} **Example:** `Device(config-subif)# encapsulation dot1q 100 second-dot1q 200`	(Required) Enables the 802.1Q encapsulation of traffic on a specified subinterface in a VLAN. • Use the **second-dot1q** keyword and the *vlan-id*argument to specify the VLAN tags to be terminated on the subinterface. • In this example, an unambiguous Q-in-Q subinterface is configured because only one inner VLAN ID is specified.

	Command or Action	Purpose
		• Q-in-Q frames with an outer VLAN ID of 100 and an inner VLAN ID of 200 will be terminated.
Step 7	**pppoe enable** [**group** *group-name*] [**max-sessions** *max-sessions-number*] **Example:** `Device(config-subif)# pppoe enable group vpn1`	Enables PPPoE sessions on a subinterface. The example specifies that the PPPoE profile, vpn1, will be used by PPPoE sessions on the subinterface.
Step 8	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and returns to interface configuration mode. • Repeat this step one more time to exit interface configuration mode.
Step 9	Repeat Step 5 to configure another subinterface. **Example:** `Device(config-if)# interface gigabitethernet 1/0/0.2`	(Optional) Configures a subinterface and enters subinterface configuration mode.

Step 10	Repeat Step 6 and Step 7 to specify the VLAN tags to be terminated on the subinterface.
	Example:
	`Device(config-subif)# encapsulation dot1q 100 second-dot1q 100-199,201-600`
	Example:
	`Device(config-subif)# pppoe enable group vpn1`
		Step 6 enables the 802.1Q encapsulation of traffic on a specified subinterface in a VLAN.
		• Use the **second-dot1q** keyword and the *vlan-id*argument to specify the VLAN tags to be terminated on the subinterface.
		• In the example, an ambiguous Q-in-Q subinterface is configured because a range of inner VLAN IDs is specified.
		• Q-in-Q frames with an outer VLAN ID of 100 and an inner VLAN ID in the range of 100 to 199 or 201 to 600 will be terminated.
		Step 7 enables PPPoE sessions on the subinterface. The example specifies that the PPPoE profile, vpn1, will be used by PPPoE sessions on the subinterface.

Step 11	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Verifying the IEEE 802.1Q-in-Q VLAN Tag Termination

Perform this optional task to verify the configuration of the IEEE 802.1Q-in-Q VLAN Tag Termination feature.

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **show vlans dot1q** [**internal** *interface-type interface-number* **.***subinterface-number*[**detail**] | **second-dot1q** *inner-id* **any**]] [**detail]**

DETAILED STEPS

Procedure

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Device> enable
```

Step 2 **show running-config**

Use this command to show the currently running configuration on the device. You can use delimiting characters to display only the relevant parts of the configuration.

Example:

```
Device# show running-config
```

Step 3 **show vlans dot1q** [**internal** *interface-type interface-number* **.***subinterface-number*[**detail**] | **second-dot1q** *inner-id* **any**]] [**detail]**

Use this command to show the statistics for all the 802.1Q VLAN IDs. In this example, only the outer VLAN ID is displayed.

Example:

```
Router# show vlans dot1q

Total statistics for 802.1Q VLAN 1:
   441 packets, 85825 bytes input
   1028 packets, 69082 bytes output
Total statistics for 802.1Q VLAN 101:
   5173 packets, 510384 bytes input
   3042 packets, 369567 bytes output
Total statistics for 802.1Q VLAN 201:
   1012 packets, 119254 bytes input
   1018 packets, 120393 bytes output
Total statistics for 802.1Q VLAN 301:
   3163 packets, 265272 bytes input
   1011 packets, 120750 bytes output
Total statistics for 802.1Q VLAN 401:
   1012 packets, 119254 bytes input
   1010 packets, 119108 bytes output
```

Configuring IEEE 802.1ad in Port-channels and Subinterfaces

Perform this task to configure IEEE802.1ad in port-channels, port-channel subinterfaces, and port-channel member links.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **dot1q tunneling ethertype** {*0x88A8* | *0x9100* | *0x9200*}
5. **interface** *type number* **.** *subinterface-number*
6. **encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id* | *vlan-id* **-** *vlan-id* [*vlan-id* **-** *vlan-id*]}
7. **ip address** *ip-address*
8. **pppoe enable** [**group** *group name*]
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface port-channel 12`	Configures an interface and enters interface configuration mode.
Step 4	**dot1q tunneling ethertype** {*0x88A8* | *0x9100* | *0x9200*} **Example:** `Device(config-if)# dot1q tunneling ethertype 0x9100`	Defines the EtherType field used by peer devices when implementing Q-in-Q VLAN tagging.
Step 5	**interface** *type number* **.** *subinterface-number* **Example:** `Device(config-if)# interface port-channel 12.1`	Configures a subinterface and enters subinterface configuration mode.
Step 6	**encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id* | *vlan-id* **-** *vlan-id* [*vlan-id* **-** *vlan-id*]}	Enables the 802.1Q encapsulation of traffic on a specified subinterface in a VLAN.

	Command or Action	Purpose
	Example: `Device(config-subif)# encapsulation dot1q 100 second-dot1q 200`	**Note** • Use the **second-dot1q** keyword and the *vlan-id* argument to specify the VLAN tags to be terminated on the subinterface. • In this example, an unambiguous Q-in-Q subinterface is configured because only one inner VLAN ID is specified. • Q-in-Q frames with an outer VLAN ID of 100 and an inner VLAN ID of 200 will be terminated.
Step 7	**ip address** *ip-address* **Example:** `Device(config-subif)# ip address 192.0.2.1.255.255.255.0`	Defines an IP address on the interface.
Step 8	**pppoe enable** [**group** *group name*] **Example:** `Device(config-subif)# pppoe enable group 2`	Enables PPPoE sessions on the subinterface.
Step 9	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Configuration Examples for IEEE 802.1Q-in-Q VLAN Tag Termination

Configuring any Keyword on Subinterfaces for IEEE 802.1Q-in-Q VLAN Tag Termination Example

Some ambiguous subinterfaces can use the **any** keyword for the inner VLAN ID specification. The **any** keyword represents any inner VLAN ID that is not explicitly configured on any other interface. In the following example, seven subinterfaces are configured with various outer and inner VLAN IDs.

Note The **any** keyword can be configured on only one subinterface of a physical interface, outer VLAN ID, or a specified port-channel subinterface.

```
interface GigabitEthernet1/0/0.1
 encapsulation dot1q 100 second-dot1q 100
interface GigabitEthernet1/0/0.2
 encapsulation dot1q 100 second-dot1q 200
```

```
interface GigabitEthernet1/0/0.3
 encapsulation dot1q 100 second-dot1q 300-400,500-600
interface GigabitEthernet1/0/0.4
 encapsulation dot1q 100 second-dot1q any
interface GigabitEthernet1/0/0.5
 encapsulation dot1q 200 second-dot1q 50
interface GigabitEthernet1/0/0.6
 encapsulation dot1q 200 second-dot1q 1000-2000,3000-4000
interface GigabitEthernet1/0/0.7
 encapsulation dot1q 200 second-dot1q any
```

The table below shows which subinterfaces are mapped to different values of the outer and inner VLAN ID on Q-in-Q frames that come in on Gigabit Ethernet interface 1/0/0.

Table 3: Subinterfaces Mapped to Outer and Inner VLAN IDs for GE Interface 1/0/0

Outer VLAN ID	Inner VLAN ID	Subinterface mapped to
100	1 through 99	GigabitEthernet1/0/0.4
100	100	GigabitEthernet1/0/0.1
100	101 through 199	GigabitEthernet1/0/0.4
100	200	GigabitEthernet1/0/0.2
100	201 through 299	GigabitEthernet1/0/0.4
100	300 through 400	GigabitEthernet1/0/0.3
100	401 through 499	GigabitEthernet1/0/0.4
100	500 through 600	GigabitEthernet1/0/0.3
100	601 through 4095	GigabitEthernet1/0/0.4
200	1 through 49	GigabitEthernet1/0/0.7
200	50	GigabitEthernet1/0/0.5
200	51 through 999	GigabitEthernet1/0/0.7
200	1000 through 2000	GigabitEthernet1/0/0.6
200	2001 through 2999	GigabitEthernet1/0/0.7
200	3000 through 4000	GigabitEthernet1/0/0.6
200	4001 through 4095	GigabitEthernet1/0/0.7

A new subinterface is now configured:

```
interface GigabitEthernet1/0/0.8
 encapsulation dot1q 200 second-dot1q 200-600,900-999
```

The table below shows the changes made to the table for the outer VLAN ID of 200. Notice that subinterface 1/0/0.7 configured with the **any** keyword now has new inner VLAN ID mappings.

Table 4: Subinterfaces Mapped to Outer and Inner VLAN IDs for GE Interface 1/0/0--Changes Resulting from Configuring GE Subinterface 1/0/0.8

Outer VLAN ID	Inner VLAN ID	Subinterface mapped to
200	1 through 49	GigabitEthernet1/0/0.7
200	50	GigabitEthernet1/0/0.5
200	51 through 199	GigabitEthernet1/0/0.7
200	200 through 600	GigabitEthernet1/0/0.8
200	601 through 899	GigabitEthernet1/0/0.7
200	900 through 999	GigabitEthernet1/0/0.8
200	1000 through 2000	GigabitEthernet1/0/0.6
200	2001 through 2999	GigabitEthernet1/0/0.7
200	3000 through 4000	GigabitEthernet1/0/0.6
200	4001 through 4095	GigabitEthernet1/0/0.7

Additional References

The following sections provide references related to the IEEE 802.1Q-in-Q VLAN Tag Termination feature.

Related Documents

Related Topic	Document Title
Related commands	*Cisco IOS LAN Switching Command Reference*

Standards

Standards	Title
IEEE 802.1Q	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

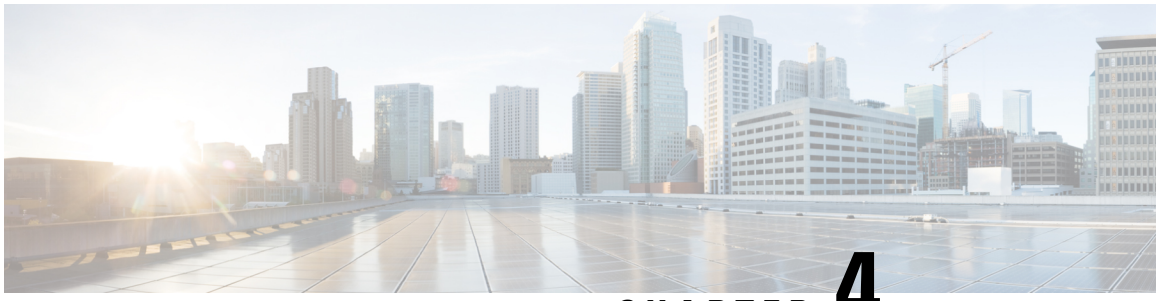
Feature Information for IEEE 802.1Q-in-Q VLAN Tag Termination

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5: Feature Information for IEEE 802.1Q-in-Q VLAN Tag Termination

Feature Name	Releases	Feature Information
IEEE 802.1Q-in-Q VLAN Tag Termination	Cisco IOS XE Release 2.1	This feature is introduced in the Cisco ASR 1000 Series Aggregation Services Routers. The following commands have been modified for this feature: **dot1q tunneling ethertype**, **encapsulation dot1q**, and **show vlans dot1q**
IEEE802.1ad Support in Port-channels and Subinterfaces	Cisco IOS XE Bengaluru 17.6	This feature is introduced in the Cisco ASR1006-X, Cisco ASR1009-X: dual RP3, dual ESP 200, dual ESP 200-X, MIP100 and EPA (1X100GE, 10X10GE), and Cisco ASR 1000 fixed routers. This feature allows IEEE802.1ad configuration on port-channels, port-channel subinterfaces, and member links with EtherTypes 0x88a8, 0x9100, and 0x9200.



CHAPTER 4

VLAN Mapping to Gigabit EtherChannel Member Links

The VLAN Mapping to Gigabit EtherChannel (GEC) Member Links feature allows you to configure static assignment of user traffic, as identified by a VLAN ID, to a given member link of a GEC bundle. You can manually assign virtual LAN (VLAN) subinterfaces to a primary and secondary link. This feature allows load balancing to downstream equipment regardless of vendor equipment capabilities, and provides failover protection by redirecting traffic to the secondary member link if the primary link fails. Member links are supported with up to 64 GEC interfaces and 14 member links per GEC interface.

- Prerequisites for VLAN Mapping to GEC Member Links, on page 37
- Restrictions for VLAN Mapping to GEC Member Links, on page 37
- Information About VLAN Mapping of GEC Member Links, on page 38
- How to Configure VLAN Mapping to GEC Links, on page 42
- Configuration Examples for VLAN Mapping to GEC Member Links, on page 44
- Additional References, on page 47
- Feature Information for VLAN Mapping to GEC Member Links, on page 47

Prerequisites for VLAN Mapping to GEC Member Links

- Each VLAN must have IEEE 802.1Q encapsulation configured.
- One primary and one secondary link must be associated with each VLAN.
- Configure per VLAN load balancing either on the main port-channel interface or enable it globally.

Restrictions for VLAN Mapping to GEC Member Links

The following restrictions are applicable for IPv6 load balancing on Gigabit EtherChannel (GEC) links:

- IPv6 traffic distribution is enabled only on port channels with flow load balancing.
- Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) is not supported on port channels.
- For Cisco ASR 1000 Series Aggregation Services Routers, the minimum number of member links per GEC interface is 1 and the maximum number is 14.

- 10 Gigabit, 40 Gigabit, 100 Gigabit Ethernet supported as a member link in VLAN mapping.
- The port-channel QinQ subinterface is not supported.
- The quality of service (QoS) policy can be applied to a port-channel subinterface when the following conditions are met:
  - Manual virtual LAN (VLAN) load balancing is supported.
  - A policy map has the appropriate service-fragment policy configured on a physical member link.

Information About VLAN Mapping of GEC Member Links

VLAN-Manual Load Balancing

When load balancing is configured for GEC links, traffic flows are mapped to different buckets as dictated by the load balancing algorithm. For each EtherChannel, a set of 16 buckets are created. The EtherChannel module decides how the buckets are distributed across member links. Each bucket has an active link associated with it that represents the interface to be used for all flows that are mapped to the same bucket.

All packets to be forwarded over the same VLAN subinterface are considered to be part of the same flow that is mapped to one bucket. Each bucket is associated with a primary and secondary pair, and the buckets point to the active interface in the pair. Only one pair is active at a time. Multiple VLAN flows can be mapped to the same bucket if their (primary and secondary) mapping is the same.

The buckets are created when VLAN manual load balancing is enabled. When VLAN load balancing is removed, the buckets are deleted. All port channels use either VLAN manual load balancing or dynamic flow-based load balancing. For information about flow-based load balancing, see the “Flow-Based Per Port-Channel Load Balancing” module.

One primary and one secondary link must be associated with a given VLAN. The primary and secondary options are available only if VLAN manual load balancing is enabled. If the following conditions are met, the load balancing information is downloaded in the forwarding plane. If any of these conditions are not met , the load balancing information is removed from the forwarding plane.

- VLAN load balancing must be enabled globally.
- IEEE 802.1Q encapsulation must be configured on each VLAN.
- One primary and one secondary member link must be enabled to manually map the VLAN traffic to the EtherChannel links.
- The primary and secondary links must be part of the port channel for traffic to use these links.

If only a primary link is specified, a secondary link is selected as the default. If neither a primary nor a secondary link is explicitly configured, the primary and secondary links are selected by default. There is no attempt to perform equal VLAN distribution across links when default links are chosen.

If the interfaces specified as primary or secondary links are not configured as part of the port channel, or if the global VLAN load balancing is not enabled, warning messages are displayed.

Warning

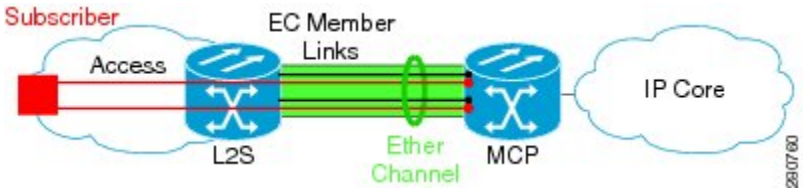
VLAN 500's main interface is not the channel group of primary=GigabitEthernet 4/0/1 Per-VLAN manual load-balancing will not take effect until channel-group is configured under the primary interface.

VLAN 500's main interface is not the channel group of secondary=GigabitEthernet 1/0/0 Per-VLAN manual load-balancing will not take effect until channel-group is configured under the primary interface.

VLAN-to-Port Channel Member Link Mapping

The figure below illustrates the traffic flow for the VLAN-to-port channel mapping.

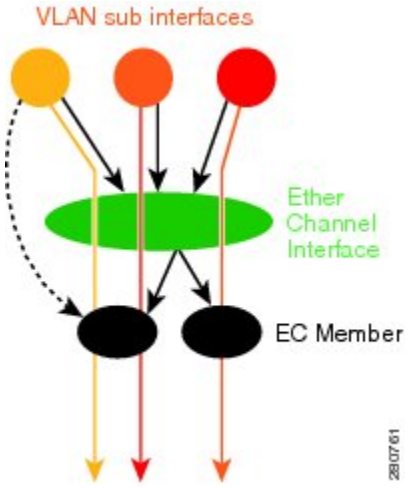
Figure 3: VLAN-to-Port Channel Member Link Mapping



The black lines represent the physical 1 Gigabit Ethernet interfaces connecting the MCP router with the Layer 2 (L2) switch. These interfaces are bundled together in port-channels, shown in green.

In the figure below, subscriber VLAN subinterfaces, shown in shades of orange and red, are configured as Layer 3 (L3) interfaces on top of EtherChannel interfaces. Mapping of the VLAN to the member link (shown with the dotted black arrow) is done through configuration and downloaded in the dataplane so that the outgoing VLA traffic (shown with orange and red arrows) is sent over the associated active primary or secondary member link. The QoS configuration in this model is applied at the VLAN subinterface and member link interface level, implying that QoS queues are created at both levels.

Figure 4: Mapping of VLAN to Member Links



VLAN Primary and Secondary Link Association

In a port-channel traffic distribution, a member link can have either a configured primary state or a secondary state, and an operational active or standby state. When the interface is up, the primary link is active. If the

primary link is down, the interface is in primary standby state while the secondary interface is in secondary active state. If the primary link is up, the secondary link is in secondary standby even if the interface is operationally up.

The primary and secondary member links are each associated with a routed VLAN configured on a port-channel main interface. When forwarding traffic for this VLAN, the primary interface is used as the outgoing interface when this interface is up; the secondary interface, if operational, is used when the primary interface is down.

If all the conditions for per-VLAN traffic distribution are not met, the mapping is not downloaded in the forwarding plane. If all the conditions are met, the dataplane is updated with this mapping.

The table below describes the primary and secondary link configuration status and the resulting function of each configuration.

Table 6: VLAN Primary and Secondary Link Mapping Status

Primary Status	Secondary Status	Description
Configured	Configured	Both primary and secondary links are specified with the **encapsulation dot1q** command. `encapsulation dot1Q vlan-id primary`
Defaulted	Defaulted	Neither a primary nor a secondary link is specified. `encapsulation dot1Q vlan-id` In a stable system, defaults for both primary and secondary links are selected in the same manner for all VLANs. The first link up that is added to the EC is selected as primary, and the second link up as secondary. If there are no links up, the primary and secondary links are selected from the down links.
Configured	Defaulted	Only the primary link is specified. `encapsulation dot1Q vlan-id primary` A secondary link that is different than the primary link is internally selected.
Configured	–	Only a primary link is specified and only one link is defined. `encapsulation dot1Q vlan-id primary` No secondary link can be selected as default when only one link is defined in the EC.
Defaulted	–	Neither a primary nor secondary link is specified, and only one link is defined. `encapsulation dot1Q vlan-id` A default for a primary link is selected. However, no default link can be selected for a secondary link if only one link is defined in the EC.

Primary Status	Secondary Status	Description
–	–	Neither a primary nor secondary link is specified, and no links are defined. `encapsulation dot1Q vlan-id` Defaults cannot be selected and no links are defined in the EC.



Note Default mappings do not override user-configured mappings even if the user-configured mappings are defined incorrectly. Once the (VLAN, primary, and secondary) association is performed (either through the CLI, default or a combination of both), the system validates the mapping and downloads it to the dataplane. If there are no VLANs configured, all traffic forwarded over the port channel is dropped.

Adding Channel Member Links

When a new member link is added, new buckets are created and downloaded in the dataplane. For all VLANs that have the interface as either primary or secondary, new VLAN-to-bucket mappings are downloaded in the dataplane. For all VLANs that need a default for primary and secondary, the default selection algorithm is triggered, and if QoS validation passes, the VLAN-to-bucket mappings are downloaded. QoS policies create VLAN queues on the newly added link.

Deleting Member Links

When a member link is removed, a warning message is displayed. All VLAN queues from the member link, VLAN-to-bucket mappings, and all affected buckets are removed.

Port Channel Link Down Notification

When a link goes down, all traffic for VLANs that have the Port Channel link assigned as primary link must be switched to secondary link if the secondary is up. The traffic for the VLANs with the Port Channel link assigned as secondary, is not affected. The Port Channel Link Down notification causes all buckets associated with a primary-secondary pair (where the primary link is down and the secondary link is up) to be updated with the secondary link. This change is communicated to the dataplane.

All buckets associated with a primary-secondary pair (and the secondary link is the down link and where primary link is up) are updated so that the primary link is now the active link. This change is communicated to the dataplane.

Port Channel Link Up Notification

When a link goes up, all traffic for VLANs that have this link assigned as primary is switched to this link. The traffic for VLANs that have this link assigned as secondary is not affected. The Port Channel Link Up notification causes all buckets associated with a primary-secondary pair, where the primary link is the link that came up, and the secondary link is up, to be notified that the primary link is up. The change is communicated to the dataplane.

All buckets associated with a primary-secondary pair, where the secondary link is the link that came up and the primary link is down are notified that the secondary link is now the primary link. The change is communicated to the dataplane.

Disabling Load Balancing on the EtherChannel

To disable load balancing on the EtherChannel, use the **no port-channel load-balancing vlan-manual** command. The following warning message is displayed if any VLAN subinterfaces exist:

```
Warning: Removing the Global VLAN LB command will affect traffic
c for all dot1Q VLANs
```

Removing a Member Link from the EtherChannel

To remove a member link from the EtherChannel (EC), use the **no channel-group** command

When a member link is removed from the EC is included in a VLAN mapping, the following warning message is displayed:

```
Warning: Removing GigabitEthernet 4/0/0 from the port-channel will affect traffic for the
dot1Q VLANs that include this link in their mapping.
```

How to Configure VLAN Mapping to GEC Links

Configuring VLAN-Based Manual Load Balancing

Perform this task to link VLAN port-channel and to enable VLAN load balancing on port channels.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **port-channel load-balancing vlan-manual**
4. **interface port-channel** *channel-number*
5. **ip address** *ip-address address-mask*
6. **exit**
7. **interface** *type subinterface-number*
8. **channel-group** *channel-number*
9. **exit**
10. **interface port-channel** *interface-number.subinterface-number*
11. **encapsulation dot1Q** *vlan-id* **primary** *interface-type slot/port* **secondary** *interface-type slot/port*
12. **ip address** *ip-address address-mask*
13. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**port-channel load-balancing vlan-manual** **Example:** `Router(config)# port-channel load-balancing vlan-manual`	Enables port-channel load balancing on the router.
Step 4	**interface port-channel** *channel-number* **Example:** `Router(config)# interface port-channel 1`	Enters interface configuration mode and defines the interface as a port channel.
Step 5	**ip address** *ip-address address-mask* **Example:** `Router(config-if)# ip address 172.16.2.3 255.255.0.0`	Specifies the IP address and mask.
Step 6	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**interface** *type subinterface-number* **Example:** `Router(config)# interface gigbabitethernet 1/1/0`	Enters interface configuration mode on the Gigabit Ethernet interface.
Step 8	**channel-group** *channel-number* **Example:** `Router(config-if)# channel-group 1`	Assigns the Gigabit Ethernet interface to the specified channel group. • The channel number is the same channel number that you specified when you created the port-channel interface.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 10	**interface port-channel** *interface-number.subinterface-number* **Example:** `Device(config)# interface port-channel 1.100`	Specifies the interface type, interface number, and subinterface number.
Step 11	**encapsulation dot1Q** *vlan-id* **primary** *interface-type slot/port* **secondary** *interface-type slot/port* **Example:** `Device(config-if)# encapsulation dot1Q 100 primary GigabitEthernet 1/1/1 secondary GigabitEthernet 1/2/1`	Enables IEEE 802.1Q encapsulation on the interface.
Step 12	**ip address** *ip-address address-mask* **Example:** `Device(config-if)# ip address 172.16.2.100 255.255.255.0`	Specifies the port channel IP address and mask.
Step 13	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode, and returns to privileged EXEC mode.

Troubleshooting Tips

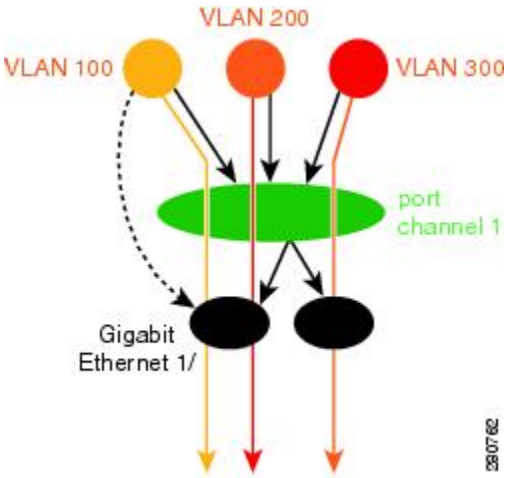
- Use the **show etherchannel load-balancing** command to display the current port channel load balancing method.
- Use the **show interfaces port-channel etherchannel** command to display the current traffic distribution.

Configuration Examples for VLAN Mapping to GEC Member Links

Example: Configuring VLAN Manual Load Balancing

This example shows how the load balancing configuration can be globally applied to define policies for handling traffic by using the **port-channel load-balancing** command. Note that IEEE 802.1Q encapsulation is configured on each port-channel interface. The figure below illustrates the port channel bundle with three VLANs used in the following configuration example:

Figure 5: Port Channel Bundle




```
port-channel load-balancing vlan-manual
!
class-map match-all BestEffort
!
class-map match-all video
!
class-map match-all voice
!
policy-map subscriber
  class voice
    priority level 1
  class video
    priority level 2
  class class-default service-fragment BE
    shape average 10000
    bandwidth remaining percent 80
policy-map aggregate-member-link
    class BestEffort service-fragment BE
    shape average 100000
!
interface Port-channel1
 ip address 172.16.2.3 255.255.0.0
!
interface Port-channel1.100
 encapsulation dot1Q 100 primary GigabitEthernet 1/1/1
              secondary GigabitEthernet 1/2/1
 ip address 172.16.2.100 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.200
 encapsulation dot1Q 200 primary GigabitEthernet 1/2/1
 ip address 172.16.2.200 255.255.255.0
 service-policy output subscriber
!
interface Port-channel1.300
 encapsulation dot1Q 300
 ip address 172.16.2.300 255.255.255.0
 service-policy output subscriber
!
interface GigabitEthernet 1/1/1
 no ip address
 channel-group 1 mode on
```

```
 service-policy output aggregate-member-link
!
interface GigabitEthernet 1/2/1
 no ip address
 channel-group 1 mode on
 service-policy output aggregate-member-link
```

Example: Troubleshooting

Example 1:

```
Device# show etherchannel load-balancing

EtherChannel Load-Balancing Configuration: vlan-manual
```

Example 2:

```
Device# show etherchannel load-balancing

EtherChannel Load-Balancing Configuration: not configured
```

Use the **show interfaces port-channel** command to display the traffic distribution currently in use.

```
Device# show interfaces port-channel 1 etherchannel

Active Member List contains 0 interfaces
 Passive Member List contains 2 interfaces
  Port: GigabitEthernet 4/0/0
    VLAN 1 (Pri, Ac, D, P)    VLAN 100 (Pri, Ac, C, P)    VLAN 200 (Sec, St, C, P)
  Port: GigabitEthernet 1/0/0
    VLAN 1 (Sec, St, D, P)    VLAN 100 (Sec, St, C, P)    VLAN 200 (Pri, Ac, C, P)
 Bucket Information for VLAN Manual LB:
    Bucket 0   (p=GigabitEthernet 4/0/0, s=GigabitEthernet 4/0/0) active GigabitEthernet
4/0/0
    Bucket 1   (p=Gigabitthernet 4/0/0, s=GigabitEthernet 1/0/0) active GigabitEthernet
4/0/0
    Bucket 4   (p=GigabitEthernet 1/0/0, s=GigabitEthernet 4/0/0) active GigabitEthernet
1/0/0
    Bucket 5   (p=GigabitEthernet 1/0/0, s=GigabitEthernet 1/0/0) active GigabitEthernet
1/0/0
```

To see the mapping of a VLAN to primary and secondary links, use the **show vlans** command.

```
Device# show vlans 100
VLAN ID: 100 (IEEE 802.1Q Encapsulation)
   Protocols Configured:          Received:        Transmitted:
VLAN trunk interfaces for VLAN ID 100:
Port-channel1.1 (100)
      Mapping for traffic load-balancing using bucket 1:
          primary   = GigabitEthernet 4/0/0 (active, C, P)
          secondary = GigabitEthernet 1/0/0 (standby, C, P)
      Total 0 packets, 0 bytes input
      Total 0 packets, 0 bytes output
No subinterface configured with ISL VLAN ID 100
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
LAN Switching commands	*Cisco IOS LAN Switching Command Reference*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

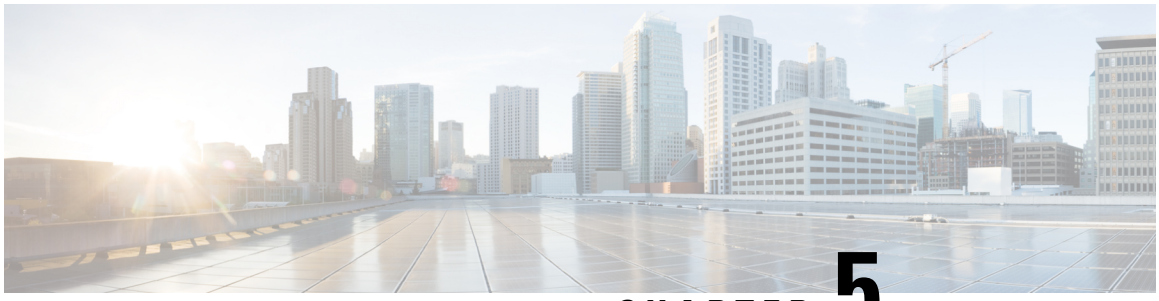
Feature Information for VLAN Mapping to GEC Member Links

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 7: Feature Information for VLAN Mapping to Gigabit EtherChannel Member Links

Feature Name	Releases	Feature Information
VLAN Mapping to Gigabit EtherChannel Member Links	Cisco IOS XE Release 2.1	The VLAN Mapping to Gigabit EtherChannel Member Links feature allows you to configure static assignment of user traffic as identified by a VLAN ID to a given member link of a GEC bundle. You can manually assign VLAN subinterfaces to a primary and secondary link. This feature allows load balancing to downstream equipment, regardless of vendor equipment capabilities, and provides failover protection by redirecting traffic to the secondary member link if the primary link fails. Member links are supported with up to 16 bundles per chassis. The following commands were modified by this feature: **encapsulation dot1q**, **port-channel load-balancing vlan-manual**, **show etherchannel load-balancing**, and **show interfaces port-channel vlan mapping**.



CHAPTER 5

Configuring Routing Between VLANs

This module provides an overview of VLANs. It describes the encapsulation protocols used for routing between VLANs and provides some basic information about designing VLANs. This module contains tasks for configuring routing between VLANS.

- Information About Routing Between VLANs, on page 49
- How to Configure Routing Between VLANS, on page 63
- Configuration Examples for Configuring Routing Between VLANs, on page 96
- Additional References, on page 112
- Feature Information for Routing Between VLANs, on page 113

Information About Routing Between VLANs

Virtual Local Area Network Definition

A virtual local area network (VLAN) is a switched network that is logically segmented on an organizational basis, by functions, project teams, or applications rather than on a physical or geographical basis. For example, all workstations and servers used by a particular workgroup team can be connected to the same VLAN, regardless of their physical connections to the network or the fact that they might be intermingled with other teams. Reconfiguration of the network can be done through software rather than by physically unplugging and moving devices or wires.

A VLAN can be thought of as a broadcast domain that exists within a defined set of switches. A VLAN consists of a number of end systems, either hosts or network equipment (such as bridges and routers), connected by a single bridging domain. The bridging domain is supported on various pieces of network equipment; for example, LAN switches that operate bridging protocols between them with a separate bridge group for each VLAN.

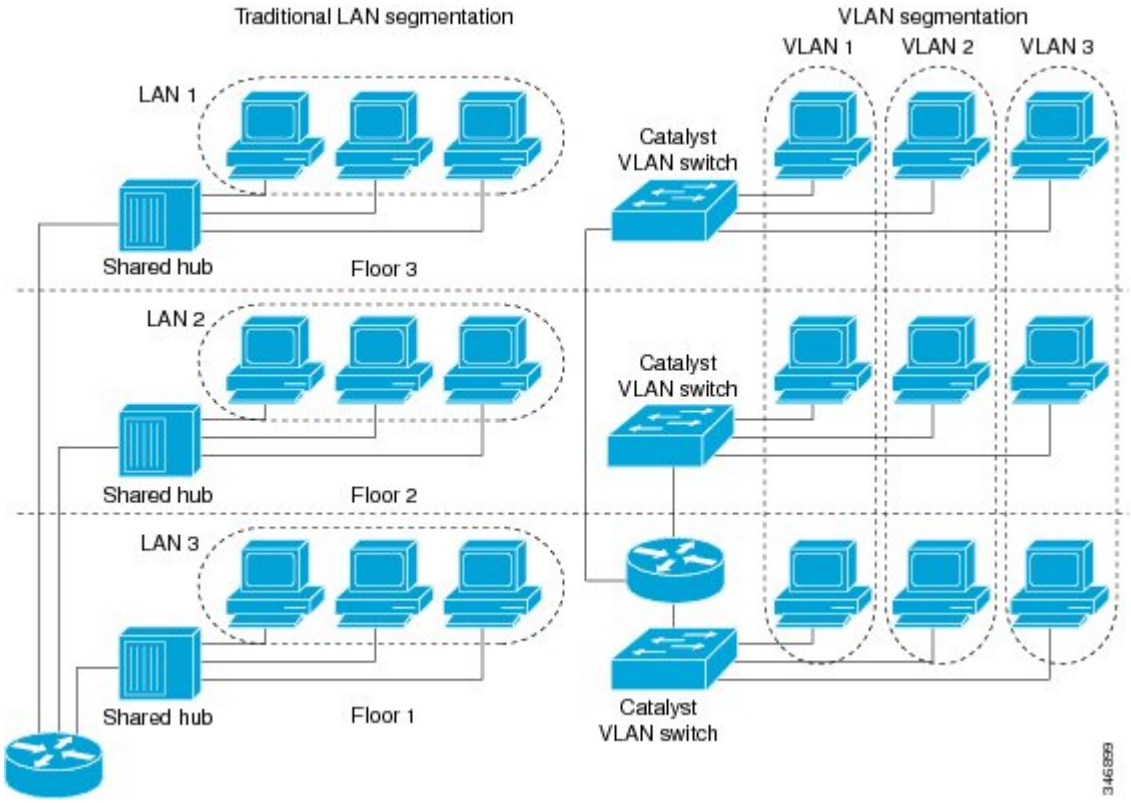
VLANs are created to provide the segmentation services traditionally provided by routers in LAN configurations. VLANs address scalability, security, and network management. Routers in VLAN topologies provide broadcast filtering, security, address summarization, and traffic flow management. None of the switches within the defined group will bridge any frames, not even broadcast frames, between two VLANs. Several key issues described in the following sections need to be considered when designing and building switched LAN internetworks:

LAN Segmentation

VLANs allow logical network topologies to overlay the physical switched infrastructure such that any arbitrary collection of LAN ports can be combined into an autonomous user group or community of interest. The technology logically segments the network into separate Layer 2 broadcast domains whereby packets are switched between ports designated to be within the same VLAN. By containing traffic originating on a particular LAN only to other LANs in the same VLAN, switched virtual networks avoid wasting bandwidth, a drawback inherent to traditional bridged and switched networks in which packets are often forwarded to LANs with no need for them. Implementation of VLANs also improves scalability, particularly in LAN environments that support broadcast- or multicast-intensive protocols and applications that flood packets throughout the network.

The figure below illustrates the difference between traditional physical LAN segmentation and logical VLAN segmentation.

Figure 6: LAN Segmentation and VLAN Segmentation



Security

VLANs improve security by isolating groups. High-security users can be grouped into a VLAN, possibly on the same physical segment, and no users outside that VLAN can communicate with them.

Broadcast Control

Just as switches isolate collision domains for attached hosts and only forward appropriate traffic out a particular port, VLANs provide complete isolation between VLANs. A VLAN is a bridging domain, and all broadcast and multicast traffic is contained within it.

VLAN Performance

The logical grouping of users allows an accounting group to make intensive use of a networked accounting system assigned to a VLAN that contains just that accounting group and its servers. That group's work will not affect other users. The VLAN configuration improves general network performance by not slowing down other users sharing the network.

Network Management

The logical grouping of users allows easier network management. It is not necessary to pull cables to move a user from one network to another. Adds, moves, and changes are achieved by configuring a port into the appropriate VLAN.

Network Monitoring Using SNMP

SNMP support has been added to provide mib-2 interfaces sparse table support for Fast Ethernet subinterfaces. Monitor your VLAN subinterface using the **show vlans** EXEC command. For more information on configuring SNMP on your Cisco network device or enabling an SNMP agent for remote access, see the "Configuring SNMP Support" module in the *Cisco IOS Network Management Configuration Guide* .

Communication Between VLANs

Communication between VLANs is accomplished through routing, and the traditional security and filtering functions of the router can be used. Cisco IOS software provides network services such as security filtering, quality of service (QoS), and accounting on a per-VLAN basis. As switched networks evolve to distributed VLANs, Cisco IOS software provides key inter-VLAN communications and allows the network to scale.

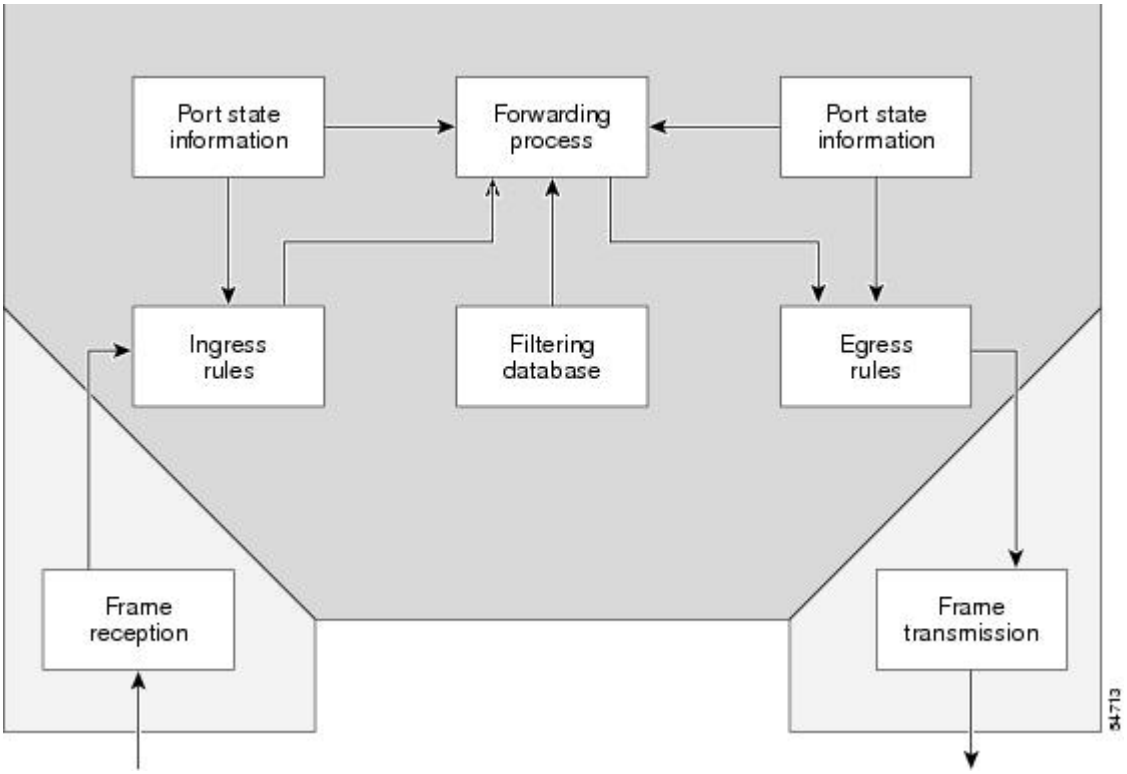
Before Cisco IOS Release 12.2, Cisco IOS support for interfaces that have 802.1Q encapsulation configured is IP, IP multicast, and IPX routing between respective VLANs represented as subinterfaces on a link. New functionality has been added in IEEE 802.1Q support for bridging on those interfaces and the capability to configure and use integrated routing and bridging (IRB).

Relaying Function

The relaying function level, as displayed in the figure below, is the lowest level in the architectural model described in the IEEE 802.1Q standard and presents three types of rules:

- Ingress rules--Rules relevant to the classification of received frames belonging to a VLAN.
- Forwarding rules between ports--Rules decide whether to filter or forward the frame.
- Egress rules (output of frames from the switch)--Rules decide if the frame must be sent tagged or untagged.

Figure 7: Relaying Function

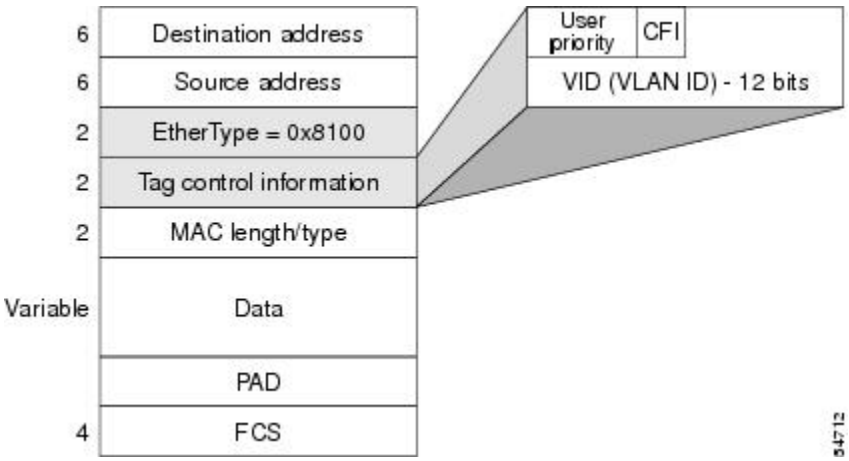


The Tagging Scheme

The figure below shows the tagging scheme proposed by the 802.3ac standard, that is, the addition of the four octets after the source MAC address. Their presence is indicated by a particular value of the EtherType field (called TPID), which has been fixed to be equal to 0x8100. When a frame has the EtherType equal to 0x8100, this frame carries the tag IEEE 802.1Q/802.1p. The tag is stored in the following two octets and it contains 3 bits of user priority, 1 bit of Canonical Format Identifier (CFI), and 12 bits of VLAN ID (VID). The 3 bits of user priority are used by the 802.1p standard; the CFI is used for compatibility reasons between Ethernet-type networks and Token Ring-type networks. The VID is the identification of the VLAN, which is basically used by the 802.1Q standard; being on 12 bits, it allows the identification of 4096 VLANs.

After the two octets of TPID and the two octets of the Tag Control Information field there are two octets that originally would have been located after the Source Address field where there is the TPID. They contain either the MAC length in the case of IEEE 802.3 or the EtherType in the case of Ethernet version 2.

Figure 8: Tagging Scheme

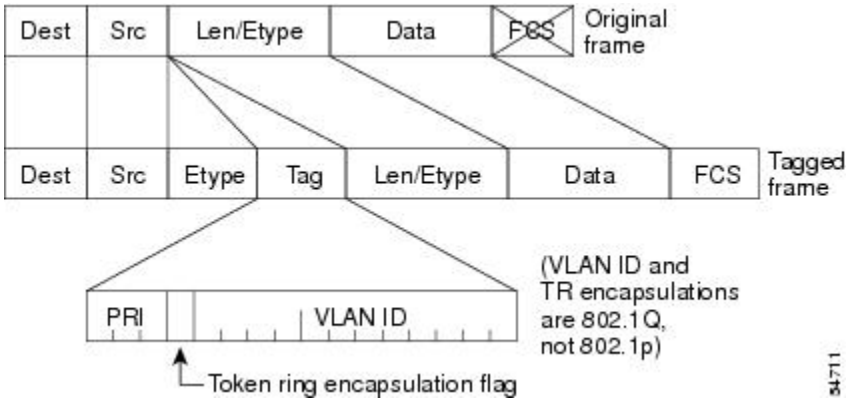


The EtherType and VLAN ID are inserted after the MAC source address, but before the original Ethertype/Length or Logical Link Control (LLC). The 1-bit CFI included a T-R Encapsulation bit so that Token Ring frames can be carried across Ethernet backbones without using 802.1H translation.

Frame Control Sequence Recomputation

The figure below shows how adding a tag in a frame recomputes the Frame Control Sequence. 802.1p and 802.1Q share the same tag.

Figure 9: Adding a Tag Recomputes the Frame Control Sequence

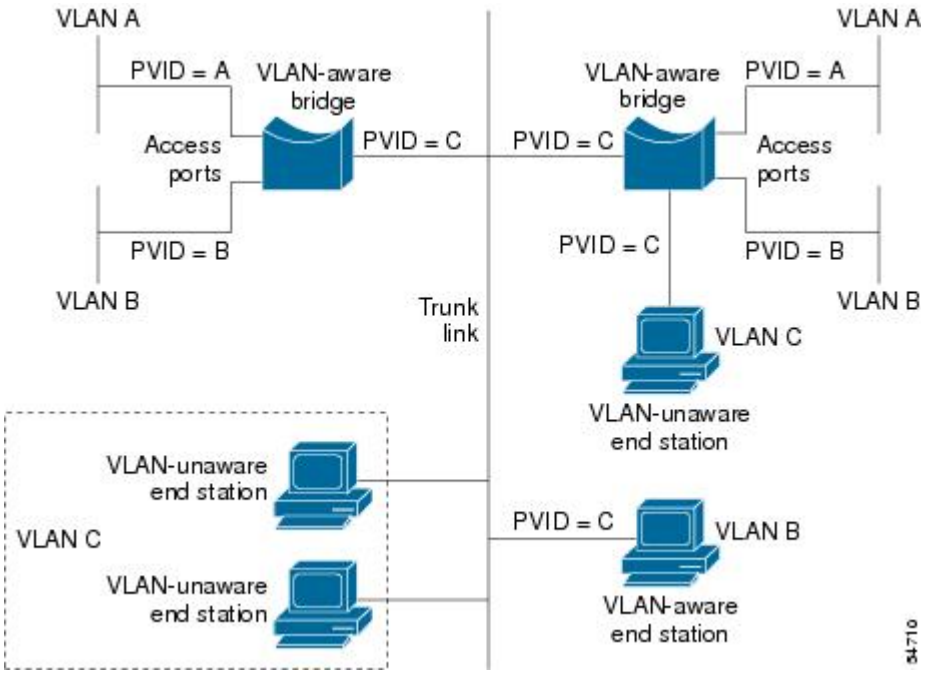


Native VLAN

Each physical port has a parameter called PVID. Every 802.1Q port is assigned a PVID value that is of its native VLAN ID (default is VLAN 1). All untagged frames are assigned to the LAN specified in the PVID parameter. When a tagged frame is received by a port, the tag is respected. If the frame is untagged, the value contained in the PVID is considered as a tag. Because the frame is untagged and the PVID is tagged to allow the coexistence, as shown in the figure below, on the same pieces of cable of VLAN-aware bridge/stations and of VLAN-unaware bridges/stations. Consider, for example, the two stations connected to the central trunk link in the lower part of the figure below. They are VLAN-unaware and they will be associated to the VLAN C, because the PVIDs of the VLAN-aware bridges are equal to VLAN C. Because the VLAN-unaware stations

will send only untagged frames, when the VLAN-aware bridge devices receive these untagged frames they will assign them to VLAN C.

Figure 10: Native VLAN



PVST+

PVST+ provides support for 802.1Q trunks and the mapping of multiple spanning trees to the single spanning tree of 802.1Q switches.

The PVST+ architecture distinguishes three types of regions:

- A PVST region
- A PVST+ region
- A MST region

Each region consists of a homogenous type of switch. A PVST region can be connected to a PVST+ region by connecting two ISL ports. Similarly, a PVST+ region can be connected to an MST region by connecting two 802.1Q ports.

At the boundary between a PVST region and a PVST+ region the mapping of spanning trees is one-to-one. At the boundary between a MST region and a PVST+ region, the ST in the MST region maps to one PVST in the PVST+ region. The one it maps to is called the common spanning tree (CST). The default CST is the PVST of VLAN 1 (Native VLAN).

All PVSTs, except for the CST, are tunneled through the MST region. Tunneling means that bridge protocol data units (BPDUs) are flooded through the MST region along the single spanning tree present in the MST region.

Ingress and Egress Rules

The BPDU transmission on the 802.1Q port of a PVST+ router will be implemented in compliance with the following rules:

- The CST BPDU (of VLAN 1, by default) is sent to the IEEE address.
- All the other BPDUs are sent to Shared Spanning Tree Protocol (SSTP)-Address and encapsulated with Logical Link Control-Subnetwork Access Protocol (LLC-SNAP) header.
- The BPDU of the CST and BPDU of the VLAN equal to the PVID of the 802.1Q trunk are sent untagged.
- All other BPDUs are sent tagged with the VLAN ID.
- The CST BPDU is also sent to the SSTP address.
- Each SSTP-addressed BPDU is also tailed by a Tag-Length-Value for the PVID checking.

The BPDU reception on the 802.1Q port of a PVST+ router will follow these rules:

- All untagged IEEE addressed BPDUs must be received on the PVID of the 802.1Q port.
- The IEEE addressed BPDUs whose VLAN ID matches the Native VLAN are processed by CST.
- All the other IEEE addressed BPDUs whose VLAN ID does not match the Native VLAN and whose port type is not of 802.1Q are processed by the spanning tree of that particular VLAN ID.
- The SSTP addressed BPDU whose VLAN ID is not equal to the TLV are dropped and the ports are blocked for inconsistency.
- All the other SSTP addressed BPDUs whose VLAN ID is not equal to the Native VLAN are processed by the spanning tree of that particular VLAN ID.
- The SSTP addressed BPDUs whose VLAN ID is equal to the Native VLAN are dropped. It is used for consistency checking.

Integrated Routing and Bridging

IRB enables a user to route a given protocol between routed interfaces and bridge groups or route a given protocol between the bridge groups. Integrated routing and bridging is supported on the following protocols:

- IP
- IPX
- AppleTalk

VLAN Colors

VLAN switching is accomplished through *frame tagging* where traffic originating and contained within a particular virtual topology carries a unique VLAN ID as it traverses a common backbone or trunk link. The VLAN ID enables VLAN switching devices to make intelligent forwarding decisions based on the embedded VLAN ID. Each VLAN is differentiated by a *color* , or VLAN identifier. The unique VLAN ID determines the *frame coloring* for the VLAN. Packets originating and contained within a particular VLAN carry the identifier that uniquely defines that VLAN (by the VLAN ID).

The VLAN ID allows VLAN switches and routers to selectively forward packets to ports with the same VLAN ID. The switch that receives the frame from the source station inserts the VLAN ID and the packet is switched onto the shared backbone network. When the frame exits the switched LAN, a switch strips the header and forwards the frame to interfaces that match the VLAN color. If you are using a Cisco network management product such as VlanDirector, you can actually color code the VLANs and monitor VLAN graphically.

Implementing VLANS

Network managers can logically group networks that span all major topologies, including high-speed technologies such as, ATM, FDDI, and Fast Ethernet. By creating virtual LANs, system and network administrators can control traffic patterns and react quickly to relocations and keep up with constant changes in the network due to moving requirements and node relocation just by changing the VLAN member list in the router configuration. They can add, remove, or move devices or make other changes to network configuration using software to make the changes.

Issues regarding creating VLANs should have been addressed when you developed your network design. Issues to consider include the following:

- Scalability
- Performance improvements
- Security
- Network additions, moves, and changes

Communication Between VLANs

Cisco IOS software provides full-feature routing at Layer 3 and translation at Layer 2 between VLANs. Five different protocols are available for routing between VLANs:

All five of these technologies are based on OSI Layer 2 bridge multiplexing mechanisms.

Inter-Switch Link Protocol

The Inter-Switch Link (ISL) protocol is used to interconnect two VLAN-capable Ethernet, Fast Ethernet, or Gigabit Ethernet devices, such as the Catalyst 3000 or 5000 switches and Cisco 7500 routers. The ISL protocol is a packet-tagging protocol that contains a standard Ethernet frame and the VLAN information associated with that frame. The packets on the ISL link contain a standard Ethernet, FDDI, or Token Ring frame and the VLAN information associated with that frame. ISL is currently supported only over Fast Ethernet links, but a single ISL link, or trunk, can carry different protocols from multiple VLANs.

Procedures for configuring ISL and Token Ring ISL (TRISL) features are provided in the Configuring Routing Between VLANs with Inter-Switch Link Encapsulation section.

IEEE 802.10 Protocol

The IEEE 802.10 protocol provides connectivity between VLANs. Originally developed to address the growing need for security within shared LAN/MAN environments, it incorporates authentication and encryption techniques to ensure data confidentiality and integrity throughout the network. Additionally, by functioning at Layer 2, it is well suited to high-throughput, low-latency switching environments. The IEEE 802.10 protocol can run over any LAN or HDLC serial interface.

Procedures for configuring routing between VLANs with IEEE 802.10 encapsulation are provided in the Configuring Routing Between VLANs with IEEE 802.10 section.

IEEE 802.1Q Protocol

The IEEE 802.1Q protocol is used to interconnect multiple switches and routers, and for defining VLAN topologies. Cisco currently supports IEEE 802.1Q for Fast Ethernet and Gigabit Ethernet interfaces.



Note Cisco does not support IEEE 802.1Q encapsulation for Ethernet interfaces.

Procedures for configuring routing between VLANs with IEEE 802.1Q encapsulation are provided in the Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation.

ATM LANE Protocol

The ATM LAN Emulation (LANE) protocol provides a way for legacy LAN users to take advantage of ATM benefits without requiring modifications to end-station hardware or software. LANE emulates a broadcast environment like IEEE 802.3 Ethernet on top of an ATM network that is a point-to-point environment.

LANE makes ATM function like a LAN. LANE allows standard LAN drivers like NDIS and ODI to be used. The virtual LAN is transparent to applications. Applications can use normal LAN functions without the underlying complexities of the ATM implementation. For example, a station can send broadcasts and multicasts, even though ATM is defined as a point-to-point technology and does not support any-to-any services.

To accomplish this, special low-level software is implemented on an ATM client workstation, called the LAN Emulation Client (LEC). The client software communicates with a central control point called a LAN Emulation Server (LES). A broadcast and unknown server (BUS) acts as a central point to distribute broadcasts and multicasts. The LAN Emulation Configuration Server (LECS) holds a database of LECs and the ELANs they belong to. The database is maintained by a network administrator.

These protocols are described in detail in the *Cisco Internetwork Design Guide* .

ATM LANE Fast Simple Server Replication Protocol

To improve the ATM LANE Simple Server Replication Protocol (SSRP), Cisco introduced the ATM LANE Fast Simple Server Replication Protocol (FSSRP). FSSRP differs from LANE SSRP in that all configured LANE servers of an ELAN are always active. FSSRP-enabled LANE clients have virtual circuits (VCs) established to a maximum of four LANE servers and BUSs at one time. If a single LANE server goes down, the LANE client quickly switches over to the next LANE server and BUS, resulting in no data or LE ARP table entry loss and no extraneous signalling.

The FSSRP feature improves upon SSRP such that LANE server and BUS switchover for LANE clients is immediate. With SSRP, a LANE server would go down, and depending on the network load, it may have taken considerable time for the LANE client to come back up joined to the correct LANE server and BUS. In addition to going down with SSRP, the LANE client would do the following:

- Clear out its data direct VCs
- Clear out its LE ARP entries
- Cause substantial signalling activity and data loss

FSSRP was designed to alleviate these problems with the LANE client. With FSSRP, each LANE client is simultaneously joined to up to four LANE servers and BUSs. The concept of the master LANE server and BUS is maintained; the LANE client uses the master LANE server when it needs LANE server BUS services. However, the difference between SSRP and FSSRP is that if and when the master LANE server goes down, the LANE client is already connected to multiple backup LANE servers and BUSs. The LANE client simply uses the next backup LANE server and BUS as the master LANE server and BUS.

VLAN Interoperability

Cisco IOS features bring added benefits to the VLAN technology. Enhancements to ISL, IEEE 802.10, and ATM LANE implementations enable routing of all major protocols between VLANs. These enhancements allow users to create more robust networks incorporating VLAN configurations by providing communications capabilities between VLANs.

Inter-VLAN Communications

The Cisco IOS supports full routing of several protocols over ISL and ATM LANE VLANs. IP, Novell IPX, and AppleTalk routing are supported over IEEE 802.10 VLANs. Standard routing attributes such as network advertisements, secondaries, and help addresses are applicable, and VLAN routing is fast switched. The table below shows protocols supported for each VLAN encapsulation format and corresponding Cisco IOS software releases in which support was introduced.

Table 8: Inter-VLAN Routing Protocol Support

Protocol	ISL	ATM LANE	IEEE 802.10
IP	Release 11.1	Release 10.3	Release 11.1
Novell IPX (default encapsulation)	Release 11.1	Release 10.3	Release 11.1
Novell IPX (configurable encapsulation)	Release 11.3	Release 10.3	Release 11.3
AppleTalk Phase II	Release 11.3	Release 10.3	--
DECnet	Release 11.3	Release 11.0	--
Banyan VINES	Release 11.3	Release 11.2	--
XNS	Release 11.3	Release 11.2	--
CLNS	Release 12.1	--	--
IS-IS	Release 12.1	--	--

VLAN Translation

VLAN translation refers to the ability of the Cisco IOS software to translate between different VLANs or between VLAN and non-VLAN encapsulating interfaces at Layer 2. Translation is typically used for selective inter-VLAN switching of nonroutable protocols and to extend a single VLAN topology across hybrid switching environments. It is also possible to bridge VLANs on the main interface; the VLAN encapsulating header is preserved. Topology changes in one VLAN domain do not affect a different VLAN.

Designing Switched VLANs

By the time you are ready to configure routing between VLANs, you will have already defined them through the switches in your network. Issues related to network design and VLAN definition should be addressed during your network design. See the *Cisco Internetwork Design Guide* and the appropriate switch documentation for information on these topics:

- Sharing resources between VLANs
- Load balancing
- Redundant links
- Addressing
- Segmenting networks with VLANs--Segmenting the network into broadcast groups improves network security. Use router access lists based on station addresses, application types, and protocol types.
- Routers and their role in switched networks--In switched networks, routers perform broadcast management, route processing, and distribution, and provide communication between VLANs. Routers provide VLAN access to shared resources and connect to other parts of the network that are either logically segmented with the more traditional subnet approach or require access to remote sites across wide-area links.

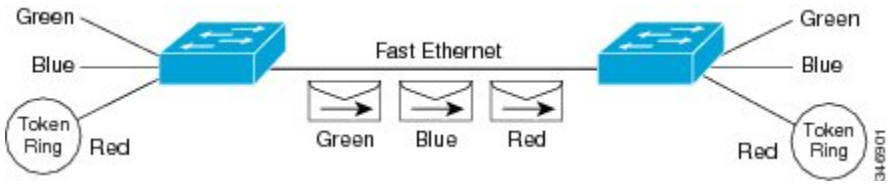
Frame Tagging in ISL

ISL is a Cisco protocol for interconnecting multiple switches and maintaining VLAN information as traffic goes between switches. ISL provides VLAN capabilities while maintaining full wire speed performance on Fast Ethernet links in full- or half-duplex mode. ISL operates in a point-to-point environment and will support up to 1000 VLANs. You can define virtually as many logical networks as are necessary for your environment.

With ISL, an Ethernet frame is encapsulated with a header that transports VLAN IDs between switches and routers. A 26-byte header that contains a 10-bit VLAN ID is propounded to the Ethernet frame.

A VLAN ID is added to the frame only when the frame is prepended for a nonlocal network. The figure below shows VLAN packets traversing the shared backbone. Each VLAN packet carries the VLAN ID within the packet header.

Figure 11: VLAN Packets Traversing the Shared Backbone



You can configure routing between any number of VLANs in your network. This section documents the configuration tasks for each protocol supported with ISL encapsulation. The basic process is the same, regardless of the protocol being routed. It involves the following tasks:

- Enabling the protocol on the router
- Enabling the protocol on the interface
- Defining the encapsulation format as ISL or TRISL

- Customizing the protocol according to the requirements for your environment

IEEE 802.1Q-in-Q VLAN Tag Termination on Subinterfaces

IEEE 802.1Q-in-Q VLAN Tag Termination simply adds another layer of IEEE 802.1Q tag (called "metro tag" or "PE-VLAN") to the 802.1Q tagged packets that enter the network. The purpose is to expand the VLAN space by tagging the tagged packets, thus producing a "double-tagged" frame. The expanded VLAN space allows the service provider to provide certain services, such as Internet access on specific VLANs for specific customers, and yet still allows the service provider to provide other types of services for their other customers on other VLANs.

Generally the service provider's customers require a range of VLANs to handle multiple applications. Service providers can allow their customers to use this feature to safely assign their own VLAN IDs on subinterfaces because these subinterface VLAN IDs are encapsulated within a service-provider designated VLAN ID for that customer. Therefore there is no overlap of VLAN IDs among customers, nor does traffic from different customers become mixed. The double-tagged frame is "terminated" or assigned on a subinterface with an expanded **encapsulation dot1q** command that specifies the two VLAN ID tags (outer VLAN ID and inner VLAN ID) terminated on the subinterface. See the figure below.

IEEE 802.1Q-in-Q VLAN Tag Termination is generally supported on whichever Cisco IOS features or protocols are supported on the subinterface; the exception is that Cisco 10000 series Internet router only supports PPPoE. For example if you can run PPPoE on the subinterface, you can configure a double-tagged frame for PPPoE. The only restriction is whether you assign ambiguous or unambiguous subinterfaces for the inner VLAN ID. See the figure below.



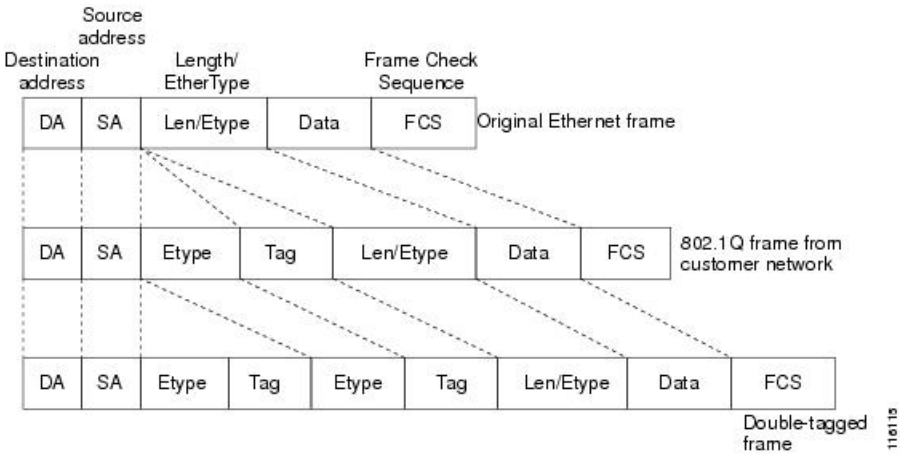
Note The Cisco 10000 series Internet router only supports Point-to-Point Protocol over Ethernet (PPPoE) and IP packets that are double-tagged for Q-in-Q VLAN tag termination. Specifically PPPoEoQ-in-Q and IPoQ-in-Q are supported.

The primary benefit for the service provider is reduced number of VLANs supported for the same number of customers. Other benefits of this feature include:

- PPPoE scalability. By expanding the available VLAN space from 4096 to approximately 16.8 million (4096 times 4096), the number of PPPoE sessions that can be terminated on a given interface is multiplied.
- When deploying Gigabyte Ethernet DSL Access Multiplexer (DSLAM) in wholesale model, you can assign the inner VLAN ID to represent the end-customer virtual circuit (VC) and assign the outer VLAN ID to represent the service provider ID.

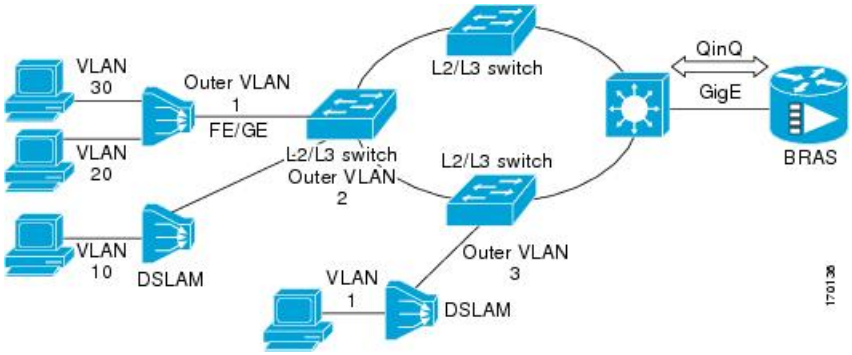
The Q-in-Q VLAN tag termination feature is simpler than the IEEE 802.1Q tunneling feature deployed for the Catalyst 6500 series switches or the Catalyst 3550 and Catalyst 3750 switches. Whereas switches require IEEE 802.1Q tunnels on interfaces to carry double-tagged traffic, routers need only encapsulate Q-in-Q VLAN tags within another level of 802.1Q tags in order for the packets to arrive at the correct destination as shown in figure below.

Figure 12: Untagged, 802.1Q-Tagged, and Double-Tagged Ethernet Frames



Cisco 10000 Series Internet Router Application

For the emerging broadband Ethernet-based DSLAM market, the Cisco 10000 series Internet router supports Q-in-Q encapsulation. With the Ethernet-based DSLAM model shown in the figure below, customers typically get their own VLAN and all these VLANs are aggregated on a DSLAM.

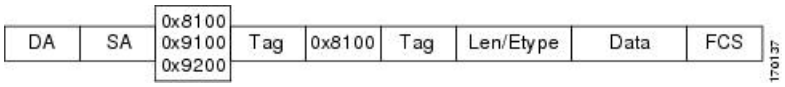


VLAN aggregation on a DSLAM will result in a lot of aggregate VLANs that at some point need to be terminated on the broadband remote access servers (BRAS). Although the model could connect the DSLAMs directly to the BRAS, a more common model uses the existing Ethernet-switched network where each DSLAM VLAN ID is tagged with a second tag (Q-in-Q) as it connects into the Ethernet-switched network.

The only model that is supported is PPPoE over Q-in-Q (PPPoEoQinQ). This can either be a PPP terminated session or as a L2TP LAC session.

The Cisco 10000 series Internet router already supports plain PPPoE and PPP over 802.1Q encapsulation. Supporting PPP over Q-in-Q encapsulation is new. PPP over Q-in-Q encapsulation processing is an extension to 802.1q encapsulation processing. A Q-in-Q frame looks like a VLAN 802.1Q frame, only it has two 802.1Q tags instead of one.

PPP over Q-in-Q encapsulation supports configurable outer tag Ethertype. The configurable Ethertype field values are 0x8100 (default), 0x9100, and 0x9200. See the figure below.



Security ACL Application on the Cisco 10000 Series Internet Router

The IEEE 802.1Q-in-Q VLAN Tag Termination feature provides limited security access control list (ACL) support for the Cisco 10000 series Internet router.

If you apply an ACL to PPPoE traffic on a Q-in-Q subinterface in a VLAN, apply the ACL directly on the PPPoE session, using virtual access interfaces (VAIs) or RADIUS attribute 11 or 242.

You can apply ACLs to virtual access interfaces by configuring them under virtual template interfaces. You can also configure ACLs by using RADIUS attribute 11 or 242. When you use attribute 242, a maximum of 30,000 sessions can have ACLs.

ACLs that are applied to the VLAN Q-in-Q subinterface have no effect and are silently ignored. In the following example, ACL 1 that is applied to the VLAN Q-in-Q subinterface level will be ignored:

```
Router(config)# interface FastEthernet3/0/0.100
Router(config-subif)# encapsulation dot1q 100 second-dot1q 200
Router(config-subif)# ip access-group 1
```

Unambiguous and Ambiguous Subinterfaces

The **encapsulation dot1q** command is used to configure Q-in-Q termination on a subinterface. The command accepts an Outer VLAN ID and one or more Inner VLAN IDs. The outer VLAN ID always has a specific value, while inner VLAN ID can either be a specific value or a range of values.

A subinterface that is configured with a single Inner VLAN ID is called an unambiguous Q-in-Q subinterface. In the following example, Q-in-Q traffic with an Outer VLAN ID of 101 and an Inner VLAN ID of 1001 is mapped to the Gigabit Ethernet 1/0.100 subinterface:

```
Router(config)# interface gigabitEehernet1/0.100
Router(config-subif)# encapsulation dot1q 101 second-dot1q 1001
```

A subinterface that is configured with multiple Inner VLAN IDs is called an ambiguous Q-in-Q subinterface. By allowing multiple Inner VLAN IDs to be grouped together, ambiguous Q-in-Q subinterfaces allow for a smaller configuration, improved memory usage and better scalability.

In the following example, Q-in-Q traffic with an Outer VLAN ID of 101 and Inner VLAN IDs anywhere in the 2001-2100 and 3001-3100 range is mapped to the Gigabit Ethernet 1/0.101 subinterface.:

```
Router(config)# interface gigabitethernet1/0.101
Router(config-subif)# encapsulation dot1q 101 second-dot1q 2001-2100,3001-3100
```

Ambiguous subinterfaces can also use the **any**keyword to specify the inner VLAN ID.

See the Monitoring and Maintaining VLAN Subinterfaces section for an example of how VLAN IDs are assigned to subinterfaces, and for a detailed example of how the **any** keyword is used on ambiguous subinterfaces.

Only PPPoE is supported on ambiguous subinterfaces. Standard IP routing is not supported on ambiguous subinterfaces.



Note On the Cisco 10000 series Internet router, Modular QoS services are only supported on unambiguous subinterfaces.

How to Configure Routing Between VLANS

Configuring a VLAN Range

Using the VLAN Range feature, you can group VLAN subinterfaces together so that any command entered in a group applies to every subinterface within the group. This capability simplifies configurations and reduces command parsing.

The VLAN Range feature provides the following benefits:

- Simultaneous Configurations: Identical commands can be entered once for a range of subinterfaces, rather than being entered separately for each subinterface.
- Overlapping Range Configurations: Overlapping ranges of subinterfaces can be configured.
- Customized Subinterfaces: Individual subinterfaces within a range can be customized or deleted.

Restrictions

- Each command you enter while you are in interface configuration mode with the **interface range** command is executed as it is entered. The commands are not batched together for execution after you exit interface configuration mode. If you exit interface configuration mode while the commands are being executed, some commands might not be executed on some interfaces in the range. Wait until the command prompt reappears before exiting interface configuration mode.
- The **no interface range** command is not supported. You must delete individual subinterfaces to delete a range.

Configuring a Range of VLAN Subinterfaces

Use the following commands to configure a range of VLAN subinterfaces.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface range** {{**ethernet** | **fastethernet** | **gigabitethernet** | **atm**} *slot* / *interface* **.** *subinterface* -{{**ethernet** | **fastethernet** | **gigabitethernet** | **atm**}*slot* / *interface* **.** *subinterface*}
4. **encapsulation dot1Q** *vlan-id*
5. **no shutdown**
6. **exit**
7. **show running-config**
8. **show interfaces**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface range** {{**ethernet** | **fastethernet** | **gigabitethernet** | **atm**} *slot* / *interface* **.** *subinterface* **-**{{**ethernet** | **fastethernet** | **gigabitethernet** | **atm**}*slot* / *interface* **.** *subinterface*} **Example:** `Router(config)# interface range fastethernet5/1.1 - fastethernet5/1.4`	Selects the range of subinterfaces to be configured. **Note** The spaces around the dash are required. For example, the command **interface range fastethernet 1 - 5**is valid; the command **interface range fastethernet 1-5** is not valid.
Step 4	**encapsulation dot1Q** *vlan-id* **Example:** `Router(config-if)# encapsulation dot1Q 301`	Applies a unique VLAN ID to each subinterface within the range. • *vlan-id* --Virtual LAN identifier. The allowed range is from 1 to 4095. • The VLAN ID specified by the *vlan-id* argument is applied to the first subinterface in the range. Each subsequent interface is assigned a VLAN ID, which is the specified *vlan-id* plus the subinterface number minus the first subinterface number (VLAN ID + subinterface number - first subinterface number).
Step 5	**no shutdown** **Example:** `Router(config-if)# no shutdown`	Activates the interface. • This command is required only if you shut down the interface.
Step 6	**exit** **Example:** `Router(config-if)# exit`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:**	Verifies subinterface configuration.

	Command or Action	Purpose
	`Router# show running-config`	
Step 8	**show interfaces** **Example:** `Router# show interfaces`	Verifies that subinterfaces have been created.

Configuring Routing Between VLANs with Inter-Switch Link Encapsulation

This section describes the Inter-Switch Link (ISL) protocol and provides guidelines for configuring ISL and Token Ring ISL (TRISL) features. This section contains the following:

Configuring AppleTalk Routing over ISL

AppleTalk can be routed over VLAN subinterfaces using the ISL and IEEE 802.10 VLAN encapsulation protocols. The AppleTalk Routing over ISL and IEEE 802.10 Virtual LANs feature provides full-feature Cisco IOS software AppleTalk support on a per-VLAN basis, allowing standard AppleTalk capabilities to be configured on VLANs.

To route AppleTalk over ISL or IEEE 802.10 between VLANs, you need to customize the subinterface to create the environment in which it will be used. Perform the steps in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **appletalk routing** [**eigrp** *router-number*]
4. **interface** *type slot* / *port* **.** *subinterface-number*
5. **encapsulation isl** *vlan-identifier*
6. **appletalk cable-range** *cable-range* [*network* **.** *node*]
7. **appletalk zone** *zone-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**appletalk routing** [**eigrp** *router-number*] **Example:** `Router(config)# appletalk routing`	Enables AppleTalk routing globally on either ISL or 802.10 interfaces.
Step 4	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface Fddi 1/0.100`	Specifies the subinterface the VLAN will use.
Step 5	**encapsulation isl** *vlan-identifier* **Example:** **Example:** or **Example:** **encapsulation sde** *said* **Example:** `Router(config-if)#  encapsulation sde 100`	Defines the encapsulation format as either ISL (**isl**) or IEEE 802.10 (**sde**), and specifies the VLAN identifier or security association identifier, respectively.
Step 6	**appletalk cable-range** *cable-range* [*network* **.** *node*] **Example:** `Router(config-if)#  appletalk cable-range 100-100 100.2`	Assigns the AppleTalk cable range and zone for the subinterface.
Step 7	**appletalk zone** *zone-name* **Example:** `Router(config-if)# appletalk zone 100`	Assigns the AppleTalk zone for the subinterface.

Configuring Banyan VINES Routing over ISL

Banyan VINES can be routed over VLAN subinterfaces using the ISL encapsulation protocol. The Banyan VINES Routing over ISL Virtual LANs feature provides full-feature Cisco IOS software Banyan VINES support on a per-VLAN basis, allowing standard Banyan VINES capabilities to be configured on VLANs.

To route Banyan VINES over ISL between VLANs, you need to configure ISL encapsulation on the subinterface. Perform the steps in the following task in the order in which they appear:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vines routing** [*address*]
4. **interface** *type slot* / *port* **.** *subinterface-number*
5. **encapsulation isl** *vlan-identifier*
6. **vines metric** [*whole* [*fraction*]]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**vines routing** [*address*] **Example:** `Router(config)# vines routing`	Enables Banyan VINES routing globally.
Step 4	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface fastethernet 1/0.1`	Specifies the subinterface on which ISL will be used.
Step 5	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 200`	Defines the encapsulation format as ISL (**isl**), and specifies the VLAN identifier.
Step 6	**vines metric** [*whole* [*fraction*]] **Example:** `Router(config-if)#vines metric 2`	Enables VINES routing metric on an interface.

Configuring DECnet Routing over ISL

DECnet can be routed over VLAN subinterfaces using the ISL VLAN encapsulation protocols. The DECnet Routing over ISL Virtual LANs feature provides full-feature Cisco IOS software DECnet support on a per-VLAN basis, allowing standard DECnet capabilities to be configured on VLANs.

To route DECnet over ISL VLANs, you need to configure ISL encapsulation on the subinterface. Perform the steps described in the following task in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Router(config)# **decnet**[*network-number*] **routing**[*decnet-address*]
4. **interface** *type slot* / *port* **.** *subinterface-number*
5. **encapsulation isl** *vlan-identifier*
6. **decnet cost** [*cost-value*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	Router(config)# **decnet**[*network-number*] **routing**[*decnet-address*] **Example:** `Router(config)# decnet routing 2.1`	Enables DECnet on the router.
Step 4	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface fastethernet 1/0.1`	Specifies the subinterface on which ISL will be used.
Step 5	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 200`	Defines the encapsulation format as ISL (**isl**), and specifies the VLAN identifier.

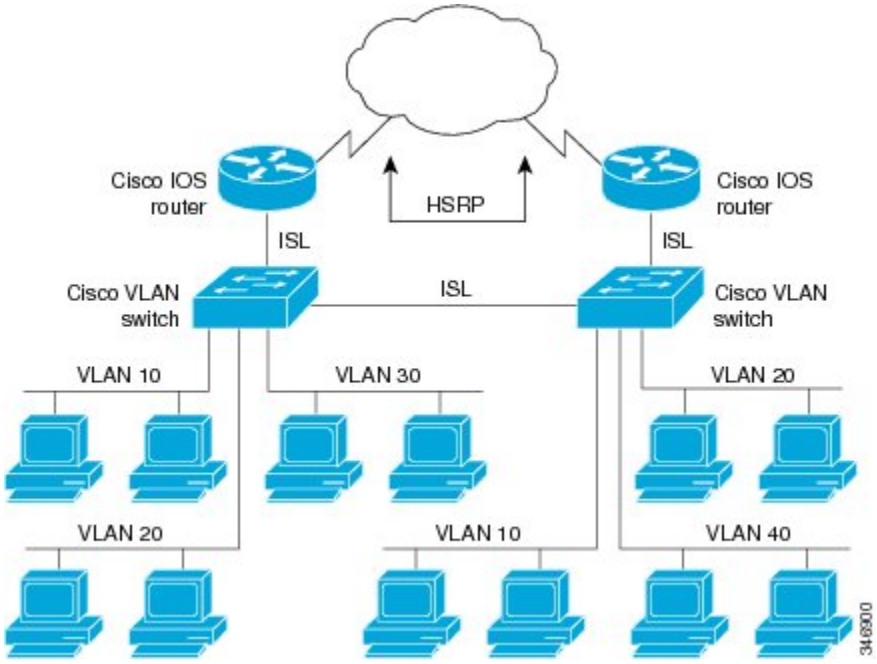
	Command or Action	Purpose
Step 6	**decnet cost** [*cost-value*] **Example:** `Router(config-if)# decnet cost 4`	Enables DECnet cost metric on an interface.

Configuring the Hot Standby Router Protocol over ISL

The Hot Standby Router Protocol (HSRP) provides fault tolerance and enhanced routing performance for IP networks. HSRP allows Cisco IOS routers to monitor each other's operational status and very quickly assume packet forwarding responsibility in the event the current forwarding device in the HSRP group fails or is taken down for maintenance. The standby mechanism remains transparent to the attached hosts and can be deployed on any LAN type. With multiple Hot Standby groups, routers can simultaneously provide redundant backup and perform loadsharing across different IP subnets.

The figure below illustrates HSRP in use with ISL providing routing between several VLANs.

Figure 13: Hot Standby Router Protocol in VLAN Configurations



A separate HSRP group is configured for each VLAN subnet so that Cisco IOS router A can be the primary and forwarding router for VLANs 10 and 20. At the same time, it acts as backup for VLANs 30 and 40. Conversely, Router B acts as the primary and forwarding router for ISL VLANs 30 and 40, as well as the secondary and backup router for distributed VLAN subnets 10 and 20.

Running HSRP over ISL allows users to configure redundancy between multiple routers that are configured as front ends for VLAN IP subnets. By configuring HSRP over ISLs, users can eliminate situations in which a single point of failure causes traffic interruptions. This feature inherently provides some improvement in overall networking resilience by providing load balancing and redundancy capabilities between subnets and VLANs.

To configure HSRP over ISLs between VLANs, you need to create the environment in which it will be used. Perform the tasks described in the following sections in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot / port* **.** *subinterface-number*
4. **encapsulation isl** *vlan-identifier*
5. **ip address** *ip-address mask* [**secondary**]
6. Router(config-if)# **standby** [*group-number*] **ip**[*ip-address*[**secondary**]]
7. **standby** [*group-number*] **timers** *hellotime holdtime*
8. **standby** [*group-number*] **priority** *priority*
9. **standby** [*group-number*] **preempt**
10. **standby** [*group-number*] **track** *type-number*[*interface-priority*]
11. **standby** [*group-number*] **authentication** *string*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot / port* **.** *subinterface-number* **Example:** `Router(config)# interface FastEthernet 1/1.110`	Specifies the subinterface on which ISL will be used and enters interface configuration mode.
Step 4	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)#  encapsulation isl 110`	Defines the encapsulation format, and specifies the VLAN identifier.
Step 5	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.1.1.2 255.255.255.0`	Specifies the IP address for the subnet on which ISL will be used.

	Command or Action	Purpose
Step 6	Router(config-if)# **standby** [*group-number*] **ip**[*ip-address*[**secondary**]] **Example:** `Router(config-if)# standby 1 ip 10.1.1.101`	Enables HSRP.
Step 7	**standby** [*group-number*] **timers** *hellotime holdtime* **Example:** `Router(config-if)# standby 1 timers 10 10`	Configures the time between hello packets and the hold time before other routers declare the active router to be down.
Step 8	**standby** [*group-number*] **priority** *priority* **Example:** `Router(config-if)# standby 1 priority 105`	Sets the Hot Standby priority used to choose the active router.
Step 9	**standby** [*group-number*] **preempt** **Example:** `Router(config-if)# standby 1 priority 105`	Specifies that if the local router has priority over the current active router, the local router should attempt to take its place as the active router.
Step 10	**standby** [*group-number*] **track** *type-number*[*interface-priority*] **Example:** `Router(config-if)# standby 1 track 4 5`	Configures the interface to track other interfaces, so that if one of the other interfaces goes down, the Hot Standby priority for the device is lowered.
Step 11	**standby** [*group-number*] **authentication** *string* **Example:** `Router(config-if)# standby 1 authentication hsrpword7`	Selects an authentication string to be carried in all HSRP messages.

What to do next

Note For more information on HSRP, see the "Configuring HSRP" module in the *Cisco IOS IP Application Services Configuration Guide* .

Configuring IP Routing over TRISL

The IP routing over TRISL VLANs feature extends IP routing capabilities to include support for routing IP frame types in VLAN configurations.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **interface** *type slot* / *port* **.** *subinterface-number*
5. **encapsulation tr-isl trbrf-vlan** *vlanid* **bridge-num** *bridge-number*
6. **ip address** *ip-address mask*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Router(config)# ip routing`	Enables IP routing on the router.
Step 4	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface FastEthernet4/0.1`	Specifies the subinterface on which TRISL will be used and enters interface configuration mode.
Step 5	**encapsulation tr-isl trbrf-vlan** *vlanid* **bridge-num** *bridge-number* **Example:** `Router(config-if# encapsulation tr-isl trbrf-vlan 999 bridge-num 14`	Defines the encapsulation for TRISL. • The DRiP database is automatically enabled when TRISL encapsulation is configured, and at least one TrBRF is defined, and the interface is configured for SRB or for routing with RIF.
Step 6	**ip address** *ip-address mask* **Example:** `Router(config-if# ip address 10.5.5.1 255.255.255.0`	Sets a primary IP address for an interface. • A mask identifies the bits that denote the network number in an IP address. When you use the mask to subnet a network, the mask is then referred to as a *subnet mask*. **Note**

	Command or Action	Purpose
		TRISL encapsulation must be specified for a subinterface before an IP address can be assigned to that subinterface.

Configuring IPX Routing on 802.10 VLANs over ISL

The IPX Encapsulation for 802.10 VLAN feature provides configurable IPX (Novell-FDDI, SAP, SNAP) encapsulation over 802.10 VLAN on router FDDI interfaces to connect the Catalyst 5000 VLAN switch. This feature extends Novell NetWare routing capabilities to include support for routing all standard IPX encapsulations for Ethernet frame types in VLAN configurations. Users with Novell NetWare environments can now configure any one of the three IPX Ethernet encapsulations to be routed using Secure Data Exchange (SDE) encapsulation across VLAN boundaries. IPX encapsulation options now supported for VLAN traffic include the following:

- Novell-FDDI (IPX FDDI RAW to 802.10 on FDDI)
- SAP (IEEE 802.2 SAP to 802.10 on FDDI)
- SNAP (IEEE 802.2 SNAP to 802.10 on FDDI)

NetWare users can now configure consolidated VLAN routing over a single VLAN trunking FDDI interface. Not all IPX encapsulations are currently supported for SDE VLAN. The IPX interior encapsulation support can be achieved by messaging the IPX header before encapsulating in the SDE format. Fast switching will also support all IPX interior encapsulations on non-MCI platforms (for example non-AGS+ and non-7000). With configurable Ethernet encapsulation protocols, users have the flexibility of using VLANs regardless of their NetWare Ethernet encapsulation. Configuring Novell IPX encapsulations on a per-VLAN basis facilitates migration between versions of Netware. NetWare traffic can now be routed across VLAN boundaries with standard encapsulation options (*arpa* , *sap* , and *snap*) previously unavailable. Encapsulation types and corresponding framing types are described in the “Configuring Novell IPX ” module of the *Cisco IOS Novell IPX Configuration Guide* .

Note Only one type of IPX encapsulation can be configured per VLAN (subinterface). The IPX encapsulation used must be the same within any particular subnet; a single encapsulation must be used by all NetWare systems that belong to the same VLAN.

To configure Cisco IOS software on a router with connected VLANs to exchange different IPX framing protocols, perform the steps described in the following task in the order in which they are appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipx routing** [*node*]
4. **interface** *fddi slot* / *port* **.** *subinterface-number*
5. **encapsulation sde** *vlan-identifier*
6. **ipx network** *network* **encapsulation** *encapsulation-type*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ipx routing** [*node*] **Example:** `Router(config)# ipx routing`	Enables IPX routing globally.
Step 4	**interface** *fddi slot / port . subinterface-number* **Example:** `Router(config)# interface 2/0.1`	Specifies the subinterface on which SDE will be used and enters interface configuration mode.
Step 5	**encapsulation sde** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 20`	Defines the encapsulation format and specifies the VLAN identifier.
Step 6	**ipx network** *network* **encapsulation** *encapsulation-type* **Example:** `Router(config-if)# ipx network 20 encapsulation sap`	Specifies the IPX encapsulation among Novell-FDDI, SAP, or SNAP.

Configuring IPX Routing over TRISL

The IPX Routing over ISL VLANs feature extends Novell NetWare routing capabilities to include support for routing all standard IPX encapsulations for Ethernet frame types in VLAN configurations. Users with Novell NetWare environments can configure either SAP or SNAP encapsulations to be routed using the TRISL encapsulation across VLAN boundaries. The SAP (Novell Ethernet_802.2) IPX encapsulation is supported for VLAN traffic.

NetWare users can now configure consolidated VLAN routing over a single VLAN trunking interface. With configurable Ethernet encapsulation protocols, users have the flexibility of using VLANs regardless of their NetWare Ethernet encapsulation. Configuring Novell IPX encapsulations on a per-VLAN basis facilitates migration between versions of Netware. NetWare traffic can now be routed across VLAN boundaries with standard encapsulation options (*sap* and *snap*) previously unavailable. Encapsulation types and corresponding

framing types are described in the “Configuring Novell IPX ” module of the *Cisco IOS Novell IPX Configuration Guide* .



Note Only one type of IPX encapsulation can be configured per VLAN (subinterface). The IPX encapsulation used must be the same within any particular subnet: A single encapsulation must be used by all NetWare systems that belong to the same LANs.

To configure Cisco IOS software to exchange different IPX framing protocols on a router with connected VLANs, perform the steps in the following task in the order in which they are appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipx routing** [*node*]
4. **interface** *type slot / port* **.** *subinterface-number*
5. **encapsulation tr-isl trbrf-vlan** *trbrf-vlan* **bridge-num** *bridge-num*
6. **ipx network** *network* **encapsulation** *encapsulation-type*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ipx routing** [*node*] **Example:** `Router(config)# source-bridge ring-group 100`	Enables IPX routing globally.
Step 4	**interface** *type slot / port* **.** *subinterface-number* **Example:** `Router(config)# interface TokenRing 3/1`	Specifies the subinterface on which TRISL will be used and enters interface configuration mode.
Step 5	**encapsulation tr-isl trbrf-vlan** *trbrf-vlan* **bridge-num** *bridge-num*	Defines the encapsulation for TRISL.

	Command or Action	Purpose
	Example: `Router(config-if)#encapsulation tr-isl trbrf-vlan 999 bridge-num 14`	
Step 6	**ipx network** *network* **encapsulation** *encapsulation-type* **Example:** `Router(config-if)# ipx network 100 encapsulation sap`	Specifies the IPX encapsulation on the subinterface by specifying the NetWare network number (if necessary) and the encapsulation type.

What to do next

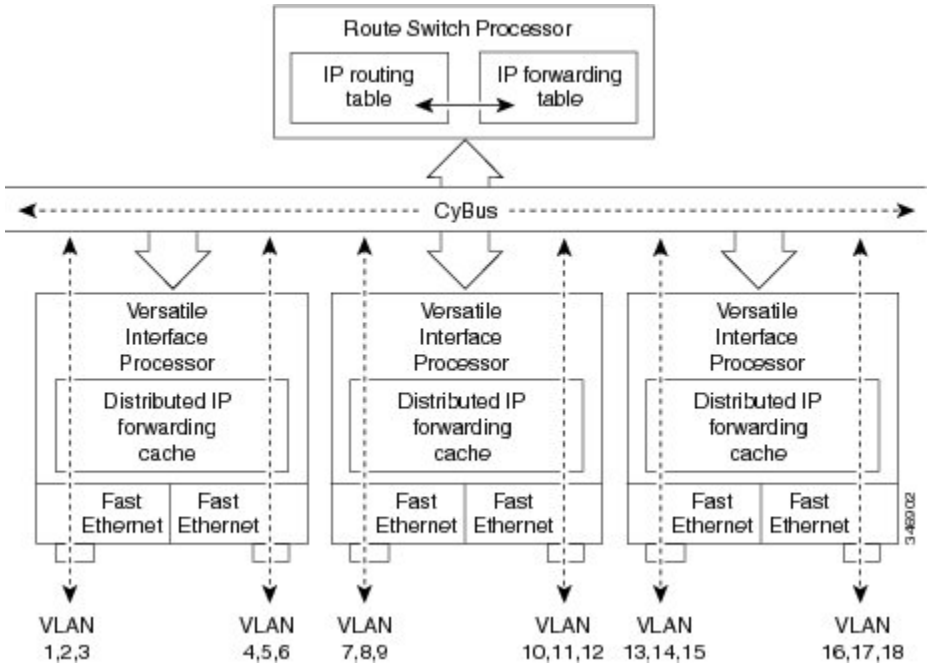
Note The default IPX encapsulation format for Cisco IOS routers is "novell-ether" (Novell Ethernet_802.3). If you are running Novell Netware 3.12 or 4.0, the new Novell default encapsulation format is Novell Ethernet_802.2 and you should configure the Cisco router with the IPX encapsulation format "sap."

Configuring VIP Distributed Switching over ISL

With the introduction of the VIP distributed ISL feature, ISL encapsulated IP packets can be switched on Versatile Interface Processor (VIP) controllers installed on Cisco 7500 series routers.

The second generation VIP2 provides distributed switching of IP encapsulated in ISL in VLAN configurations. Where an aggregation route performs inter-VLAN routing for multiple VLANs, traffic can be switched autonomously on-card or between cards rather than through the central Route Switch Processor (RSP). The figure below shows the VIP distributed architecture of the Cisco 7500 series router.

Figure 14: Cisco 7500 Distributed Architecture



This distributed architecture allows incremental capacity increases by installation of additional VIP cards. Using VIP cards for switching the majority of IP VLAN traffic in multiprotocol environments substantially increases routing performance for the other protocols because the RSP offloads IP and can then be dedicated to switching the non-IP protocols.

VIP distributed switching offloads switching of ISL VLAN IP traffic to the VIP card, removing involvement from the main CPU. Offloading ISL traffic to the VIP card substantially improves networking performance. Because you can install multiple VIP cards in a router, VLAN routing capacity is increased linearly according to the number of VIP cards installed in the router.

To configure distributed switching on the VIP, you must first configure the router for IP routing. Perform the tasks described below in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **interface** *type slot / port-adapter / port*
5. **ip route-cache distributed**
6. **encapsulation isl** *vlan-identifier*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Router(config)# ip routing`	Enables IP routing on the router. • For more information about configuring IP routing, see the appropriate Cisco IOS *IP Routing Configuration Guide* for the version of Cisco IOS you are using.
Step 4	**interface** *type slot / port-adapter / port* **Example:** `Router(config)# interface FastEthernet1/0/0`	Specifies the interface and enters interface configuration mode.
Step 5	**ip route-cache distributed** **Example:** `Router(config-if)# ip route-cache distributed`	Enables VIP distributed switching of IP packets on the interface.
Step 6	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 1`	Defines the encapsulation format as ISL, and specifies the VLAN identifier.

Configuring XNS Routing over ISL

XNS can be routed over VLAN subinterfaces using the ISL VLAN encapsulation protocol. The XNS Routing over ISL Virtual LANs feature provides full-feature Cisco IOS software XNS support on a per-VLAN basis, allowing standard XNS capabilities to be configured on VLANs.

To route XNS over ISL VLANs, you need to configure ISL encapsulation on the subinterface. Perform the steps described in the following task in the order in which they appear.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **xns routing** [*address*]
4. **interface** *type slot* / *port* **.** *subinterface-number*
5. **encapsulation isl** *vlan-identifier*
6. **xns network** [*number*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**xns routing** [*address*] **Example:** `Router(config)# xns routing 0123.4567.adcb`	Enables XNS routing globally.
Step 4	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface fastethernet 1/0.1`	Specifies the subinterface on which ISL will be used and enters interface configuration mode.
Step 5	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 100`	Defines the encapsulation format as ISL (**isl**), and specifies the VLAN identifier.
Step 6	**xns network** [*number*] **Example:** `Router(config-if)# xns network 20`	Enables XNS routing on the subinterface.

Configuring CLNS Routing over ISL

CLNS can be routed over VLAN subinterfaces using the ISL VLAN encapsulation protocol. The CLNS Routing over ISL Virtual LANs feature provides full-feature Cisco IOS software CLNS support on a per-VLAN basis, allowing standard CLNS capabilities to be configured on VLANs.

To route CLNS over ISL VLANs, you need to configure ISL encapsulation on the subinterface. Perform the steps described in the following task in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **clns routing**
4. **interface** *type slot* / *port* **.** *subinterface-number*
5. **encapsulation isl** *vlan-identifier*
6. **clns enable**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**clns routing** **Example:** `Router(config)# clns routing`	Enables CLNS routing globally.
Step 4	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config-if)# interface fastethernet 1/0.1`	Specifies the subinterface on which ISL will be used and enters interface configuration mode.
Step 5	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 100`	Defines the encapsulation format as ISL (**isl**), and specifies the VLAN identifier.
Step 6	**clns enable** **Example:** `Router(config-if)# clns enable`	Enables CLNS routing on the subinterface.

Configuring IS-IS Routing over ISL

IS-IS routing can be enabled over VLAN subinterfaces using the ISL VLAN encapsulation protocol. The IS-IS Routing over ISL Virtual LANs feature provides full-feature Cisco IOS software IS-IS support on a per-VLAN basis, allowing standard IS-IS capabilities to be configured on VLANs.

To enable IS-IS over ISL VLANs, you need to configure ISL encapsulation on the subinterface. Perform the steps described in the following task in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router isis** [*tag*]
4. **net** *network-entity-title*
5. **interface** *type slot* / *port* **.** *subinterface-number*
6. **encapsulation isl** *vlan-identifier*
7. **clns router isis network** [*tag*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router isis** [*tag*] **Example:** `Router(config)# isis routing test-proc2`	Enables IS-IS routing, and enters router configuration mode.
Step 4	**net** *network-entity-title* **Example:** `Router(config)# net 49.0001.0002.aaaa.aaaa.aaaa.00`	Configures the NET for the routing process.
Step 5	**interface** *type slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface fastethernet 2.`	Specifies the subinterface on which ISL will be used and enters interface configuration mode.

	Command or Action	Purpose
Step 6	**encapsulation isl** *vlan-identifier* **Example:** `Router(config-if)# encapsulation isl 101`	Defines the encapsulation format as ISL (**isl**), and specifies the VLAN identifier.
Step 7	**clns router isis network** [*tag*] **Example:** `Router(config-if)# clns router is-is network test-proc2`	Specifies the interfaces that should be actively routing IS-IS.

Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation

This section describes the required and optional tasks for configuring routing between VLANs with IEEE 802.1Q encapsulation. The IEEE 802.1Q protocol is used to interconnect multiple switches and routers, and for defining VLAN topologies.

Prerequisites

Configuring routing between VLANs with IEEE 802.1Q encapsulation assumes the presence of a single spanning tree and of an explicit tagging scheme with one-level tagging.

You can configure routing between any number of VLANs in your network.

Restrictions

The IEEE 802.1Q standard is extremely restrictive to untagged frames. The standard provides only a per-port VLANs solution for untagged frames. For example, assigning untagged frames to VLANs takes into consideration only the port from which they have been received. Each port has a parameter called a *permanent virtual identification* (Native VLAN) that specifies the VLAN assigned to receive untagged frames.

The main characteristics of the IEEE 802.1Q are that it assigns frames to VLANs by filtering and that the standard assumes the presence of a single spanning tree and of an explicit tagging scheme with one-level tagging.

This section contains the configuration tasks for each protocol supported with IEEE 802.1Q encapsulation. The basic process is the same, regardless of the protocol being routed. It involves the following tasks:

- Enabling the protocol on the router
- Enabling the protocol on the interface
- Defining the encapsulation format as IEEE 802.1Q
- Customizing the protocol according to the requirements for your environment

To configure IEEE 802.1Q on your network, perform the following tasks. One of the following tasks is required depending on the protocol being used.

- Configuring AppleTalk Routing over IEEE 802.1Q, on page 83 (required)
- Configuring IP Routing over IEEE 802.1Q, on page 84 (required)

- Configuring IPX Routing over IEEE 802.1Q, on page 85 (required)

The following tasks are optional. Perform the following tasks to connect a network of hosts over a simple bridging-access device to a remote access concentrator bridge between IEEE 802.1Q VLANs. The following sections contain configuration tasks for the Integrated Routing and Bridging, Transparent Bridging, and PVST+ Between VLANs with IEEE 802.1Q Encapsulation:

- Configuring a VLAN for a Bridge Group with Default VLAN1, on page 86 (optional)
- Configuring a VLAN for a Bridge Group as a Native VLAN, on page 87 (optional)

Configuring AppleTalk Routing over IEEE 802.1Q

AppleTalk can be routed over virtual LAN (VLAN) subinterfaces using the IEEE 802.1Q VLAN encapsulation protocol. AppleTalk Routing provides full-feature Cisco IOS software AppleTalk support on a per-VLAN basis, allowing standard AppleTalk capabilities to be configured on VLANs.

To route AppleTalk over IEEE 802.1Q between VLANs, you need to customize the subinterface to create the environment in which it will be used. Perform the steps in the order in which they appear.

Use the following task to enable AppleTalk routing on IEEE 802.1Q interfaces.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **appletalk routing** [**eigrp** *router-number*]
4. **interface fastethernet** *slot* / *port* **.** subinterface-number
5. **encapsulation dot1q** *vlan-identifier*
6. **appletalk cable-range** *cable-range* [*network* **.** *node*]
7. **appletalk zone** *zone-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**appletalk routing** [**eigrp** *router-number*] **Example:**	Enables AppleTalk routing globally.

	Command or Action	Purpose
	`Router(config)# appletalk routing`	
Step 4	**interface fastethernet** *slot* / *port* **.** subinterface-number **Example:** `Router(config)# interface fastethernet 4/1.00`	Specifies the subinterface the VLAN will use and enters interface configuration mode.
Step 5	**encapsulation dot1q** *vlan-identifier* **Example:** `Router(config-if)# encapsulation dot1q 100`	Defines the encapsulation format as IEEE 802.1Q (**dot1q**), and specifies the VLAN identifier.
Step 6	**appletalk cable-range** *cable-range* [*network* **.** *node*] **Example:** `Router(config-if)# appletalk cable-range 100-100 100.1`	Assigns the AppleTalk cable range and zone for the subinterface.
Step 7	**appletalk zone** *zone-name* **Example:** `Router(config-if)# appletalk zone eng`	Assigns the AppleTalk zone for the subinterface.

What to do next

Note For more information on configuring AppleTalk, see the "Configuring AppleTalk" module in the *Cisco IOS AppleTalk Configuration Guide* .

Configuring IP Routing over IEEE 802.1Q

IP routing over IEEE 802.1Q extends IP routing capabilities to include support for routing IP frame types in VLAN configurations using the IEEE 802.1Q encapsulation.

To route IP over IEEE 802.1Q between VLANs, you need to customize the subinterface to create the environment in which it will be used. Perform the tasks described in the following sections in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. ip routing
4. **interface fastethernet** *slot* / *port* **.** subinterface-number
5. **encapsulation dot1q** vlanid
6. **ip address** *ip-address mask*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	ip routing **Example:** `Router(config)# ip routing`	Enables IP routing on the router.
Step 4	**interface fastethernet** *slot* / *port* **.** subinterface-number **Example:** `Router(config)# interface fastethernet 4/1.101`	Specifies the subinterface on which IEEE 802.1Q will be used and enters interface configuration mode.
Step 5	**encapsulation dot1q** vlanid **Example:** `Router(config-if)# encapsulation dot1q 101`	Defines the encapsulation format at IEEE.802.1Q (dot1q) and specifies the VLAN identifier.
Step 6	**ip address** *ip-address mask* **Example:** `Router(config-if)# ip addr 10.0.0.11 255.0.0.0`	Sets a primary IP address and mask for the interface.

What to do next

Once you have IP routing enabled on the router, you can customize the characteristics to suit your environment. See the appropriate *Cisco IOS IP Routing Configuration Guide* for the version of Cisco IOS you are using.

Configuring IPX Routing over IEEE 802.1Q

IPX routing over IEEE 802.1Q VLANs extends Novell NetWare routing capabilities to include support for routing Novell Ethernet_802.3 encapsulation frame types in VLAN configurations. Users with Novell NetWare environments can configure Novell Ethernet_802.3 encapsulation frames to be routed using IEEE 802.1Q encapsulation across VLAN boundaries.

To configure Cisco IOS software on a router with connected VLANs to exchange IPX Novell Ethernet_802.3 encapsulated frames, perform the steps described in the following task in the order in which they appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipx routing** [*node*]
4. **interface fastethernet** *slot* / *port* **.** subinterface-number
5. **encapsulation dot1q** vlanid
6. **ipx network** *network*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ipx routing** [*node*] **Example:** `Router(config)# ipx routing`	Enables IPX routing globally.
Step 4	**interface fastethernet** *slot* / *port* **.** subinterface-number **Example:** `Router(config)# interface fastethernet 4/1.102`	Specifies the subinterface on which IEEE 802.1Q will be used and enters interface configuration mode.
Step 5	**encapsulation dot1q** vlanid **Example:** `Router(config-if)# encapsulation dot1q 102`	Defines the encapsulation format at IEEE.802.1Q (**dot1q**) and specifies the VLAN identifier.
Step 6	**ipx network** *network* **Example:** `Router(config-if)# ipx network 100`	Specifies the IPX network number.

Configuring a VLAN for a Bridge Group with Default VLAN1

Use the following task to configure a VLAN associated with a bridge group with a default native VLAN.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface fastethernet** *slot* / *port* **.** subinterface-number
4. **encapsulation dot1q** vlanid
5. **bridge-group** *bridge-group*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface fastethernet** *slot* / *port* **.** subinterface-number **Example:** `Router(config)# interface fastethernet 4/1.100`	Selects a particular interface to configure and enters interface configuration mode.
Step 4	**encapsulation dot1q** vlanid **Example:** `Router(config-subif)# encapsulation dot1q 1`	Defines the encapsulation format at IEEE.802.1Q (dot1q) and specifies the VLAN identifier. • The specified VLAN is by default the native VLAN. **Note** If there is no explicitly defined native VLAN, the default VLAN1 becomes the native VLAN.
Step 5	**bridge-group** *bridge-group* **Example:** `Router(config-subif)# bridge-group 1`	Assigns the bridge group to the interface.

Configuring a VLAN for a Bridge Group as a Native VLAN

Use the following task to configure a VLAN associated to a bridge group as a native VLAN.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **interface fastethernet** *slot* / *port* **.** subinterface-number
4. **encapsulation dot1q** *vlanid* **native**
5. **bridge-group** *bridge-group*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface fastethernet** *slot* / *port* **.** subinterface-number **Example:** `Router(config)# interface fastethernet 4/1.100`	Selects a particular interface to configure and enters interface configuration mode.
Step 4	**encapsulation dot1q** *vlanid* **native** **Example:** `Router(config-subif)# encapsulation dot1q 20 native`	Defines the encapsulation format at IEEE.802.1Q (**dot1q**) and specifies the VLAN identifier. VLAN 20 is specified as the native VLAN. **Note** If there is no explicitly defined native VLAN, the default VLAN1 becomes the native VLAN.
Step 5	**bridge-group** *bridge-group* **Example:** `Router(config-subif)# bridge-group 1`	Assigns the bridge group to the interface.

What to do next

Note If there is an explicitly defined native VLAN, VLAN1 will only be used to process CST.

Configuring IEEE 802.1Q-in-Q VLAN Tag Termination

Encapsulating IEEE 802.1Q VLAN tags within 802.1Q enables service providers to use a single VLAN to support customers who have multiple VLANs. The IEEE 802.1Q-in-Q VLAN Tag Termination feature on the subinterface level preserves VLAN IDs and keeps traffic in different customer VLANs segregated.

You must have checked Feature Navigator to verify that your Cisco device and software image support this feature.

You must be connected to an Ethernet device that supports double VLAN tag imposition/disposition or switching.

The following restrictions apply to the Cisco 10000 series Internet router for configuring IEEE 802.1Q-in-Q VLAN tag termination:

- Supported on Ethernet, FastEthernet, or Gigabit Ethernet interfaces.
- Supports only Point-to-Point Protocol over Ethernet (PPPoE) packets that are double-tagged for Q-in-Q VLAN tag termination.
- IP and Multiprotocol Label Switching (MPLS) packets are not supported.
- Modular QoS can be applied to unambiguous subinterfaces only.
- Limited ACL support.

Perform these tasks to configure the main interface used for the Q-in-Q double tagging and to configure the subinterfaces.

Configuring EtherType Field for Outer VLAN Tag Termination

The following restrictions are applicable for the Cisco 10000 series Internet router:

- PPPoE is already configured.
- Virtual private dial-up network (VPDN) is enabled.

The first task is optional. A step in this task shows you how to configure the EtherType field to be 0x9100 for the outer VLAN tag, if that is required.

After the subinterface is defined, the 802.1Q encapsulation is configured to use the double tagging.

To configure the EtherType field for Outer VLAN Tag Termination, use the following steps. This task is optional.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **dot1q tunneling ethertype** *ethertype*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface gigabitethernet 1/0/0`	Configures an interface and enters interface configuration mode.
Step 4	**dot1q tunneling ethertype** *ethertype* **Example:** `Router(config-if)# dot1q tunneling ethertype 0x9100`	(Optional) Defines the Ethertype field type used by peer devices when implementing Q-in-Q VLAN tagging. • Use this command if the Ethertype of peer devices is 0x9100 or 0x9200 (0x9200 is only supported on the Cisco 10000 series Internet router). • Cisco 10000 series Internet router supports both the 0x9100 and 0x9200 Ethertype field types.

Configuring the Q-in-Q Subinterface

Use the following steps to configure Q-in-Q subinterfaces. This task is required.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* **.** *subinterface-number*
4. **encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id*| *vlan-id* **-** *vlan-id* [**,** *vlan-id* **-** *vlan-id*]}
5. **pppoe enable** [**group** *group-name*]
6. **exit**
7. Repeat Step 3 to configure another subinterface.
8. Repeat Step 4 and Step 5 to specify the VLAN tags to be terminated on the subinterface.
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **.** *subinterface-number* **Example:** `Router(config)# interface gigabitethernet 1/0/0.1`	Configures a subinterface and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id*| *vlan-id* **-** *vlan-id* [**,** *vlan-id* **-** *vlan-id*]} **Example:** `Router(config-subif)# encapsulation dot1q 100 second-dot1q 200`	(Required) Enables the 802.1Q encapsulation of traffic on a specified subinterface in a VLAN. • Use the **second-dot1q** keyword and the *vlan-id*argument to specify the VLAN tags to be terminated on the subinterface. • In this example, an unambiguous Q-in-Q subinterface is configured because only one inner VLAN ID is specified. • Q-in-Q frames with an outer VLAN ID of 100 and an inner VLAN ID of 200 will be terminated.
Step 5	**pppoe enable** [**group** *group-name*] **Example:** `Router(config-subif)# pppoe enable group vpn1`	Enables PPPoE sessions on a subinterface. • The example specifies that the PPPoE profile, vpn1, will be used by PPPoE sessions on the subinterface.
Step 6	**exit** **Example:** `Router(config-subif)# exit`	Exits subinterface configuration mode and returns to interface configuration mode. • Repeat this step one more time to exit interface configuration mode.
Step 7	Repeat Step 3 to configure another subinterface. **Example:** `Router(config-if)# interface gigabitethernet 1/0/0.2`	(Optional) Configures a subinterface and enters subinterface configuration mode.

	Command or Action	Purpose

Step 8	Repeat Step 4 and Step 5 to specify the VLAN tags to be terminated on the subinterface.
	Example:
	`Router(config-subif)# encapsulation dot1q 100 second-dot1q 100-199,201-600`
	Example:
	Example:
	`Router(config-subif)# pppoe enable group vpn1`
	Example:
		Step 4 enables the 802.1Q encapsulation of traffic on a specified subinterface in a VLAN.
		• Use the **second-dot1q** keyword and the *vlan-id*argument to specify the VLAN tags to be terminated on the subinterface.
		• In the example, an ambiguous Q-in-Q subinterface is configured because a range of inner VLAN IDs is specified.
		• Q-in-Q frames with an outer VLAN ID of 100 and an inner VLAN ID in the range of 100 to 199 or 201 to 600 will be terminated.
		Step 5 enables PPPoE sessions on the subinterface. The example specifies that the PPPoE profile, vpn1, will be used by PPPoE sessions on the subinterface.
		Note
		Step 5 is required for the Cisco 10000 series Internet router because it only supports PPPoEoQinQ traffic.

Step 9	**end** **Example:** `Router(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Verifying the IEEE 802.1Q-in-Q VLAN Tag Termination

Perform this optional task to verify the configuration of the IEEE 802.1Q-in-Q VLAN Tag Termination feature.

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **show vlans dot1q** [**internal** | *interface-type interface-number* **.***subinterface-number*[**detail**] | *outer-id*[*interface-type interface-number* | **second-dot1q** [*inner-id*| **any**]] [**detail**]]

DETAILED STEPS

Procedure

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Router> enable
```

Step 2 **show running-config**

Use this command to show the currently running configuration on the device. You can use delimiting characters to display only the relevant parts of the configuration.

The following shows the currently running configuration on a Cisco 7300 series router:

Example:

```
Router# show running-config
.
.
.
interface FastEthernet0/0.201
 encapsulation dot1Q 201
 ip address 10.7.7.5 255.255.255.252
!
interface FastEthernet0/0.401
 encapsulation dot1Q 401
 ip address 10.7.7.13 255.255.255.252
!
interface FastEthernet0/0.201999
 encapsulation dot1Q 201 second-dot1q any
 pppoe enable
!
interface FastEthernet0/0.2012001
 encapsulation dot1Q 201 second-dot1q 2001
 ip address 10.8.8.9 255.255.255.252
!
interface FastEthernet0/0.2012002
 encapsulation dot1Q 201 second-dot1q 2002
 ip address 10.8.8.13 255.255.255.252
!
interface FastEthernet0/0.4019999
 encapsulation dot1Q 401 second-dot1q 100-900,1001-2000
 pppoe enable
!
interface GigabitEthernet5/0.101
 encapsulation dot1Q 101
 ip address 10.7.7.1 255.255.255.252
!
interface GigabitEthernet5/0.301
 encapsulation dot1Q 301
 ip address 10.7.7.9 255.255.255.252
!
interface GigabitEthernet5/0.301999
 encapsulation dot1Q 301 second-dot1q any
 pppoe enable
!
interface GigabitEthernet5/0.1011001
 encapsulation dot1Q 101 second-dot1q 1001
 ip address 10.8.8.1 255.255.255.252
!
interface GigabitEthernet5/0.1011002
 encapsulation dot1Q 101 second-dot1q 1002
 ip address 10.8.8.5 255.255.255.252
!
interface GigabitEthernet5/0.1019999
 encapsulation dot1Q 101 second-dot1q 1-1000,1003-2000
 pppoe enable
.
```

```
.
.
```

The following shows the currently running configuration on a Cisco 10000 series Internet router:

Example:

```
Router# show running-config
.
.
.
interface FastEthernet1/0/0.201
 encapsulation dot1Q 201
 ip address 10.7.7.5 255.255.255.252
!
interface FastEthernet1/0/0.401
 encapsulation dot1Q 401
 ip address 10.7.7.13 255.255.255.252
!
interface FastEthernet1/0/0.201999
 encapsulation dot1Q 201 second-dot1q any
 pppoe enable
!
interface FastEthernet1/0/0.4019999
 encapsulation dot1Q 401 second-dot1q 100-900,1001-2000
 pppoe enable
!
interface GigabitEthernet5/0/0.101
 encapsulation dot1Q 101
 ip address 10.7.7.1 255.255.255.252
!
interface GigabitEthernet5/0/0.301
 encapsulation dot1Q 301
 ip address 10.7.7.9 255.255.255.252
!
interface GigabitEthernet5/0/0.301999
 encapsulation dot1Q 301 second-dot1q any
 pppoe enable
!
interface GigabitEthernet5/0/0.1019999
 encapsulation dot1Q 101 second-dot1q 1-1000,1003-2000
 pppoe enable
.
.
.
```

Step 3 **show vlans dot1q** [**internal** | *interface-type interface-number* **.***subinterface-number*[**detail**] | *outer-id*[*interface-type interface-number* | **second-dot1q** [*inner-id*| **any**]] [**detail**]]

Use this command to show the statistics for all the 802.1Q VLAN IDs. In this example, only the outer VLAN ID is displayed.

Note
The **show vlans dot1q**command is not supported on the Cisco 10000 series Internet router.

Example:

```
Router# show vlans dot1q
Total statistics for 802.1Q VLAN 1:
   441 packets, 85825 bytes input
   1028 packets, 69082 bytes output
Total statistics for 802.1Q VLAN 101:
```

```
   5173 packets, 510384 bytes input
   3042 packets, 369567 bytes output
Total statistics for 802.1Q VLAN 201:
   1012 packets, 119254 bytes input
   1018 packets, 120393 bytes output
Total statistics for 802.1Q VLAN 301:
   3163 packets, 265272 bytes input
   1011 packets, 120750 bytes output
Total statistics for 802.1Q VLAN 401:
   1012 packets, 119254 bytes input
   1010 packets, 119108 bytes output
```

Monitoring and Maintaining VLAN Subinterfaces

Use the following task to determine whether a VLAN is a native VLAN.

SUMMARY STEPS

1. **enable**
2. **show vlans**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show vlans** **Example:** `Router# show vlans`	Displays VLAN subinterfaces.

Monitoring and Maintaining VLAN Subinterfaces Example

The following is sample output from the **show vlans**command indicating a native VLAN and a bridged group:

```
Router# show vlans
Virtual LAN ID:  1 (IEEE 802.1Q Encapsulation)
   vLAN Trunk Interface:   FastEthernet1/0/2
 This is configured as native Vlan for the following interface(s) :
FastEthernet1/0/2
   Protocols Configured:   Address: Received:        Transmitted:
Virtual LAN ID:  100 (IEEE 802.1Q Encapsulation)
   vLAN Trunk Interface:   FastEthernet1/0/2.1
   Protocols Configured:   Address: Received:        Transmitted:
        Bridging        Bridge Group 1 0                  0
```

The following is sample output from the **show vlans**command that shows the traffic count on Fast Ethernet subinterfaces:

```
Router# show vlans
Virtual LAN ID:  2 (IEEE 802.1Q Encapsulation)
    vLAN Trunk Interface:   FastEthernet5/0.1

    Protocols Configured:   Address:              Received:        Transmitted:
           IP               172.16.0.3                   16               92129

Virtual LAN ID:  3 (IEEE 802.1Q Encapsulation)

    vLAN Trunk Interface:   Ethernet6/0/1.1

    Protocols Configured:   Address:              Received:        Transmitted:
           IP               172.20.0.3                 1558               1521

Virtual LAN ID:  4 (Inter Switch Link Encapsulation)

    vLAN Trunk Interface:   FastEthernet5/0.2

    Protocols Configured:   Address:              Received:        Transmitted:
           IP               172.30.0.3                    0                  7
```

Configuration Examples for Configuring Routing Between VLANs

Single Range Configuration Example

The following example configures the Fast Ethernet subinterfaces within the range 5/1.1 and 5/1.4 and applies the following VLAN IDs to those subinterfaces:

Fast Ethernet5/1.1 = VLAN ID 301 (vlan-id)

Fast Ethernet5/1.2 = VLAN ID 302 (vlan-id = 301 + 2 - 1 = 302)

Fast Ethernet5/1.3 = VLAN ID 303 (vlan-id = 301 + 3 - 1 = 303)

Fast Ethernet5/1.4 = VLAN ID 304 (vlan-id = 301 + 4 - 1 = 304)

```
Router(config)# interface range fastethernet5/1.1 - fastethernet5/1.4

Router(config-if)# encapsulation dot1Q 301
Router(config-if)# no shutdown

Router(config-if)#
*Oct  6 08:24:35: %LINK-3-UPDOWN: Interface FastEthernet5/1.1, changed state to up
*Oct  6 08:24:35: %LINK-3-UPDOWN: Interface FastEthernet5/1.2, changed state to up
*Oct  6 08:24:35: %LINK-3-UPDOWN: Interface FastEthernet5/1.3, changed state to up
*Oct  6 08:24:35: %LINK-3-UPDOWN: Interface FastEthernet5/1.4, changed state to up
*Oct  6 08:24:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet5/1.1, changed
 state to up
*Oct  6 08:24:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet5/1.2, changed
 state to up
*Oct  6 08:24:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet5/1.3, changed
 state to up
*Oct  6 08:24:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet5/1.4, changed
 state to up
```

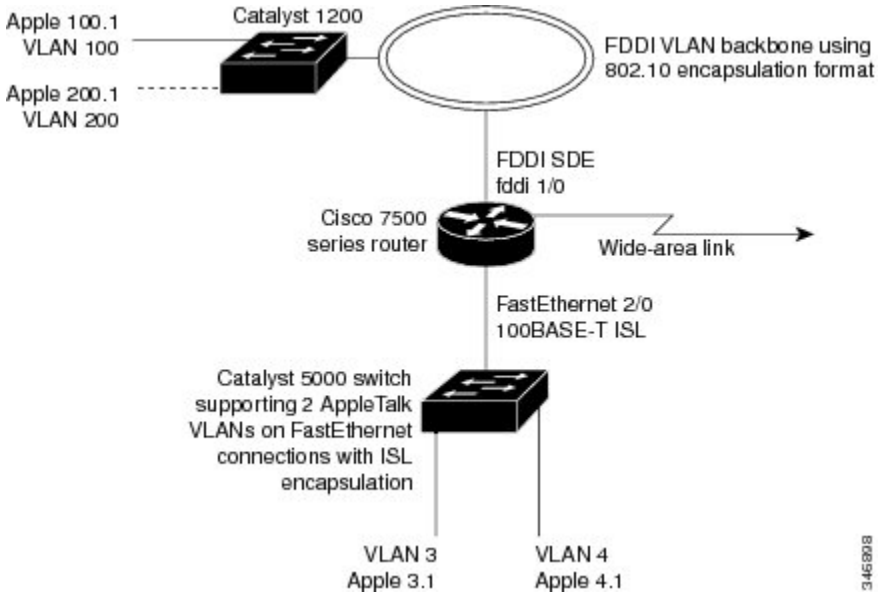
ISL Encapsulation Configuration Examples

This section provides the following configuration examples for each of the protocols described in this module:

AppleTalk Routing over ISL Configuration Example

The configuration example illustrated in the figure below shows AppleTalk being routed between different ISL and IEEE 802.10 VLAN encapsulating subinterfaces.

Figure 15: Routing AppleTalk over VLAN Encapsulations



As shown in the figure above, AppleTalk traffic is routed to and from switched VLAN domains 3, 4, 100, and 200 to any other AppleTalk routing interface. This example shows a sample configuration file for the Cisco 7500 series router with the commands entered to configure the network shown in the figure above.

Cisco 7500 Router Configuration

```
!
appletalk routing
interface Fddi 1/0.100
 encapsulation sde 100
 appletalk cable-range 100-100 100.2
 appletalk zone 100
!
interface Fddi 1/0.200
 encapsulation sde 200
 appletalk cable-range 200-200 200.2
 appletalk zone 200
!
interface FastEthernet 2/0.3
 encapsulation isl 3
 appletalk cable-range 3-3 3.2
 appletalk zone 3
!
interface FastEthernet 2/0.4
 encapsulation isl 4
```

```
 appletalk cable-range 4-4 4.2
 appletalk zone 4
!
```

Banyan VINES Routing over ISL Configuration Example

To configure routing of the Banyan VINES protocol over ISL trunks, you need to define ISL as the encapsulation type. This example shows Banyan VINES configured to be routed over an ISL trunk:

```
vines routing
interface fastethernet 0.1
 encapsulation isl 100
 vines metric 2
```

DECnet Routing over ISL Configuration Example

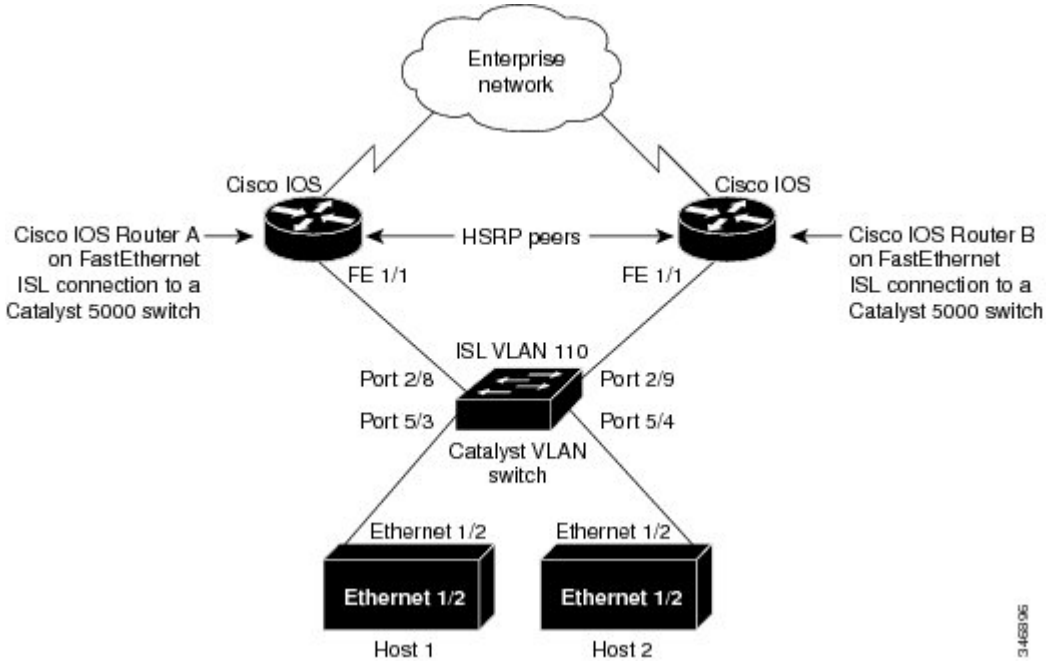
To configure routing the DECnet protocol over ISL trunks, you need to define ISL as the encapsulation type. This example shows DECnet configured to be routed over an ISL trunk:

```
decnet routing 2.1
interface fastethernet 1/0.1
 encapsulation isl 200
 decnet cost 4
```

HSRP over ISL Configuration Example

The configuration example shown in the figure below shows HSRP being used on two VLAN routers sending traffic to and from ISL VLANs through a Catalyst 5000 switch. Each router forwards its own traffic and acts as a standby for the other.

Figure 16: Hot Standby Router Protocol Sample Configuration



The topology shown in the figure above shows a Catalyst VLAN switch supporting Fast Ethernet connections to two routers running HSRP. Both routers are configured to route HSRP over ISLs.

The standby conditions are determined by the standby commands used in the configuration. Traffic from Host 1 is forwarded through Router A. Because the priority for the group is higher, Router A is the active router for Host 1. Because the priority for the group serviced by Host 2 is higher in Router B, traffic from Host 2 is forwarded through Router B, making Router B its active router.

In the configuration shown in the figure above, if the active router becomes unavailable, the standby router assumes active status for the additional traffic and automatically routes the traffic normally handled by the router that has become unavailable.

Host 1 Configuration

```
interface Ethernet 1/2
 ip address 10.1.1.25 255.255.255.0
 ip route 0.0.0.0 0.0.0.0 10.1.1.101
```

Host 2 Configuration

```
interface Ethernet 1/2
 ip address 10.1.1.27 255.255.255.0
 ip route 0.0.0.0 0.0.0.0 10.1.1.102
!
```

Router A Configuration

```
interface FastEthernet 1/1.110
 encapsulation isl 110
 ip address 10.1.1.2 255.255.255.0
 standby 1 ip 10.1.1.101
 standby 1 preempt
 standby 1 priority 105
 standby 2 ip 10.1.1.102
 standby 2 preempt
!
end
!
```

Router B Configuration

```
interface FastEthernet 1/1.110
 encapsulation isl 110
 ip address 10.1.1.3 255.255.255.0
 standby 1 ip 10.1.1.101
 standby 1 preempt
 standby 2 ip 10.1.1.102
 standby 2 preempt
 standby 2 priority 105
router igrp 1
!
network 10.1.0.0
network 10.2.0.0
!
```

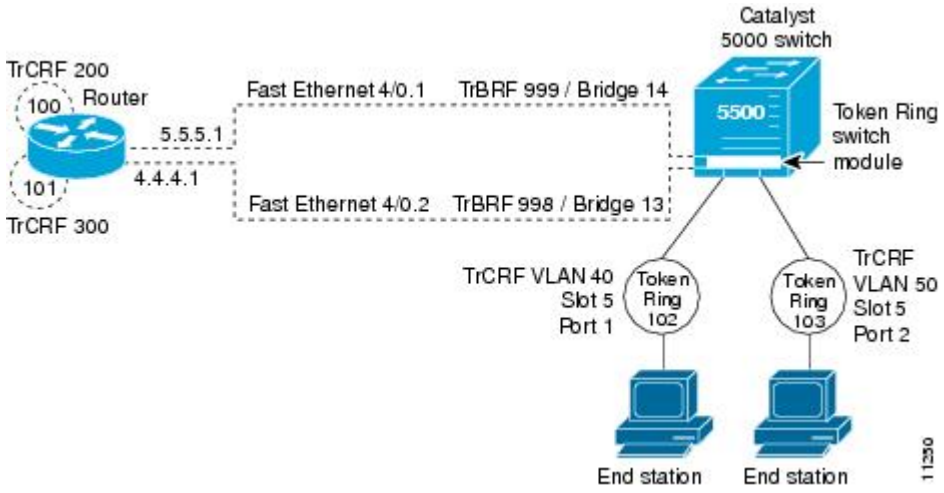
VLAN Switch Configuration

```
set vlan 110 5/4
set vlan 110 5/3
set trunk 2/8 110
set trunk 2/9 110
```

IP Routing with RIF Between TrBRF VLANs Example

The figure below shows IP routing with RIF between two TrBRF VLANs.

Figure 17: IP Routing with RIF Between TrBRF VLANs



The following is the configuration for the router:

```
interface FastEthernet4/0.1
 ip address 10.5.5.1 255.255.255.0
 encapsulation tr-isl trbrf-vlan 999 bridge-num 14
 multiring trcrf-vlan 200 ring 100
 multiring all
!
 interface FastEthernet4/0.2
 ip address 10.4.4.1 255.255.255.0
 encapsulation tr-isl trbrf-vlan 998 bridge-num 13
 multiring trcrf-vlan 300 ring 101
 multiring all
```

The following is the configuration for the Catalyst 5000 switch with the Token Ring switch module in slot 5. In this configuration, the Token Ring port 102 is assigned with TrCRF VLAN 40 and the Token Ring port 103 is assigned with TrCRF VLAN 50:

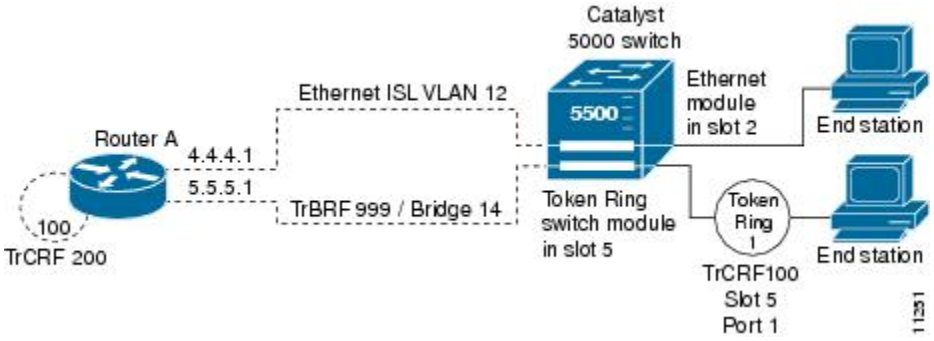
```
#vtp
set vtp domain trisl
set vtp mode server
set vtp v2 enable
#drip
set set tokenring reduction enable
set tokenring distrib-crf disable
#vlans
set vlan 999 name trbrf type trbrf bridge 0xe stp ieee
set vlan 200 name trcrf200 type trcrf parent 999 ring 0x64 mode srb
```

```
set vlan 40 name trcrf40 type trcrf parent 999 ring 0x66 mode srb
set vlan 998 name trbrf type trbrf bridge 0xd stp ieee
set vlan 300 name trcrf300 type trcrf parent 998 ring 0x65 mode srb
set vlan 50 name trcrf50 type trcrf parent 998 ring 0x67 mode srb
#add token port to trcrf 40
set vlan 40   5/1
#add token port to trcrf 50
set vlan 50   5/2
set trunk 1/2 on
```

IP Routing Between a TRISL VLAN and an Ethernet ISL VLAN Example

The figure below shows IP routing between a TRISL VLAN and an Ethernet ISL VLAN.

Figure 18: IP Routing Between a TRISL VLAN and an Ethernet ISL VLAN



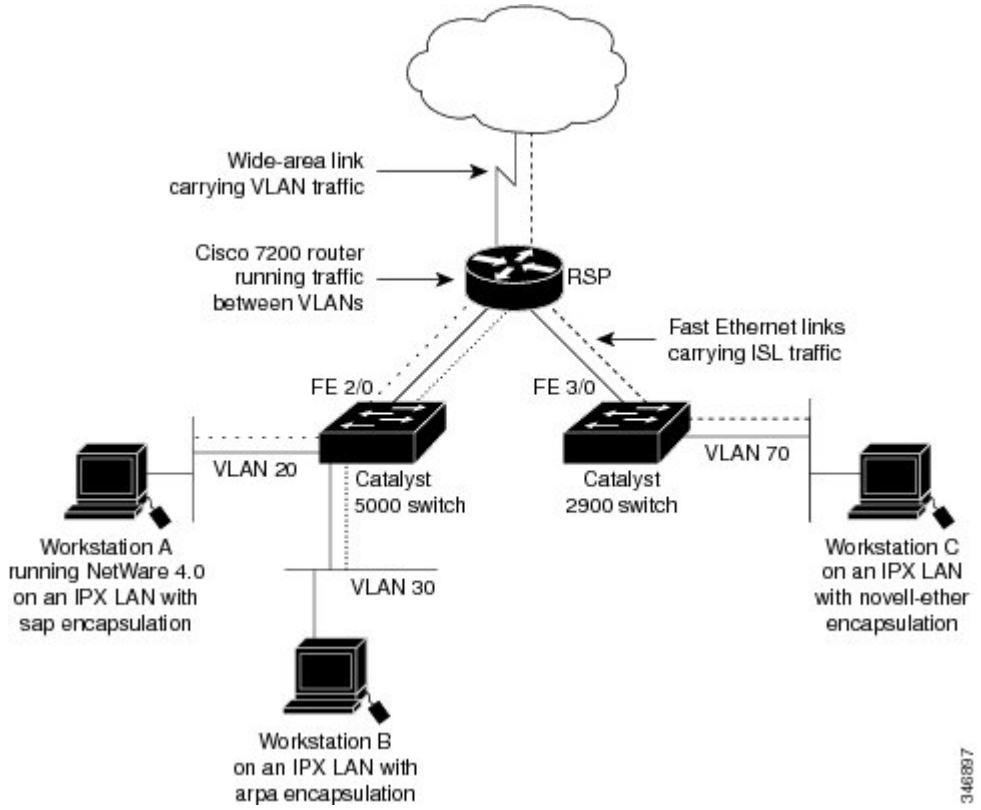
The following is the configuration for the router:

```
interface FastEthernet4/0.1
 ip address 10.5.5.1 255.255.255.0
 encapsulation tr-isl trbrf-vlan 999 bridge-num 14
 multiring trcrf-vlan 20 ring 100
 multiring all
!
interface FastEthernet4/0.2
 ip address 10.4.4.1 255.255.255.0
 encapsulation isl 12
```

IPX Routing over ISL Configuration Example

The figure below shows IPX interior encapsulations configured over ISL encapsulation in VLAN configurations. Note that three different IPX encapsulation formats are used. VLAN 20 uses SAP encapsulation, VLAN 30 uses ARPA, and VLAN 70 uses novell-ether encapsulation. Prior to the introduction of this feature, only the default encapsulation format, “novell-ether,” was available for routing IPX over ISL links in VLANs.

Figure 19: Configurable IPX Encapsulations Routed over ISL in VLAN Configurations



VLAN 20 Configuration

```
ipx routing
interface FastEthernet 2/0
 no shutdown
interface FastEthernet 2/0.20
 encapsulation isl 20
 ipx network 20 encapsulation sap
```

VLAN 30 Configuration

```
ipx routing
interface FastEthernet 2/0
 no shutdown
interface FastEthernet 2/0.30
 encapsulation isl 30
 ipx network 30 encapsulation arpa
```

VLAN 70 Configuration

```
ipx routing
interface FastEthernet 3/0
 no shutdown
interface Fast3/0.70
```

```
 encapsulation isl 70
 ipx network 70 encapsulation novell-ether
```

IPX Routing on FDDI Interfaces with SDE Example

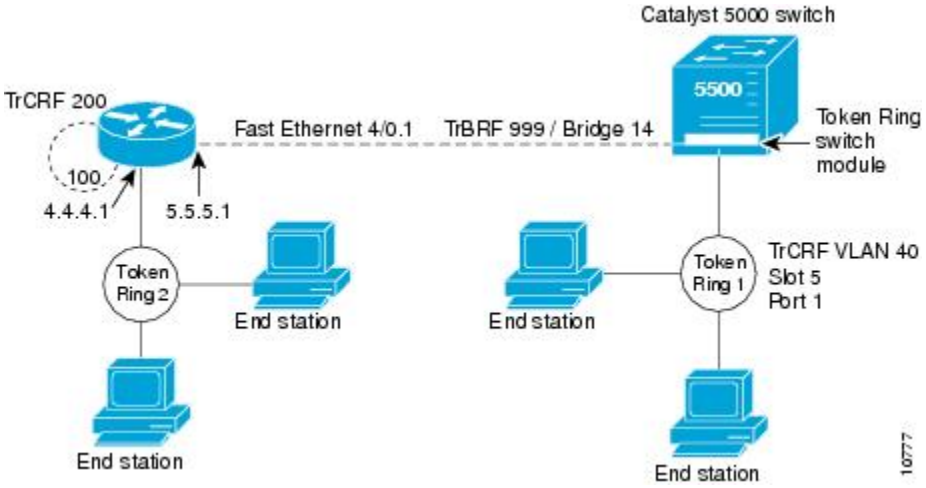
The following example enables IPX routing on FDDI interfaces 0.2 and 0.3 with SDE. On FDDI interface 0.2, the encapsulation type is SNAP. On FDDI interface 0.3, the encapsulation type is Novell's FDDI_RAW.

```
ipx routing
interface fddi 0.2 enc sde 2
 ipx network f02 encapsulation snap
interface fddi 0.3 enc sde 3
 ipx network f03 encapsulation novell-fddi
```

Routing with RIF Between a TRISL VLAN and a Token Ring Interface Example

The figure below shows routing with RIF between a TRISL VLAN and a Token Ring interface.

Figure 20: Routing with RIF Between a TRISL VLAN and a Token Ring Interface



The following is the configuration for the router:

```
source-bridge ring-group 100
!
interface TokenRing 3/1
 ip address 10.4.4.1 255.255.255.0
!
interface FastEthernet4/0.1
 ip address 10.5.5.1 255.255.255.0
 encapsulation tr-isl trbrf 999 bridge-num 14
 multiring trcrf-vlan 200 ring-group 100
 multiring all
```

The following is the configuration for the Catalyst 5000 switch with the Token Ring switch module in slot 5. In this configuration, the Token Ring port 1 is assigned to the TrCRF VLAN 40:

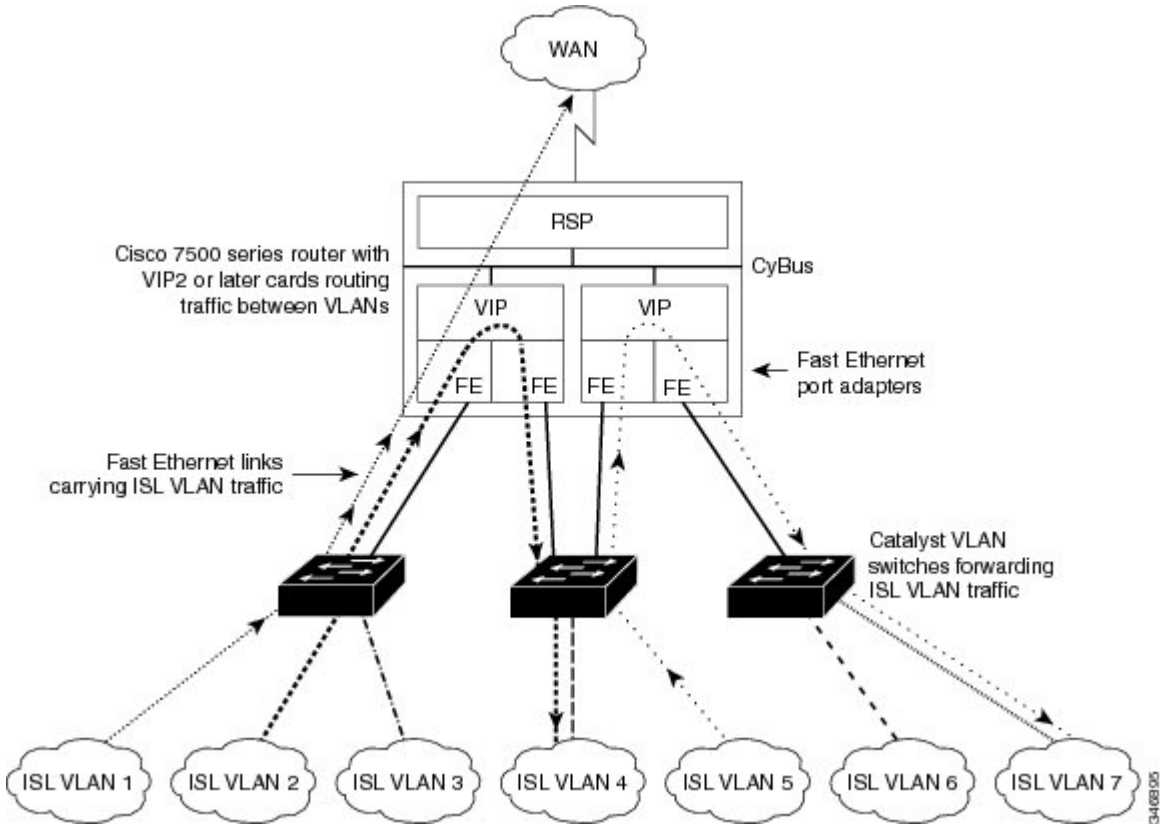
```
#vtp
set vtp domain trisl
set vtp mode server
set vtp v2 enable
#drip
```

```
set set tokenring reduction enable
set tokenring distrib-crf disable
#vlans
set vlan 999 name trbrf type trbrf bridge 0xe stp ieee
set vlan 200 name trcrf200 type trcrf parent 999 ring 0x64 mode srt
set vlan 40 name trcrf40 type trcrf parent 999 ring 0x1 mode srt
#add token port to trcrf 40
set vlan 40   5/1
set trunk 1/2 on
```

VIP Distributed Switching over ISL Configuration Example

The figure below shows a topology in which Catalyst VLAN switches are connected to routers forwarding traffic from a number of ISL VLANs. With the VIP distributed ISL capability in the Cisco 7500 series router, each VIP card can route ISL-encapsulated VLAN IP traffic. The inter-VLAN routing capacity is increased linearly by the packet-forwarding capability of each VIP card.

Figure 21: VIP Distributed ISL VLAN Traffic



In the figure above, the VIP cards forward the traffic between ISL VLANs or any other routing interface. Traffic from any VLAN can be routed to any of the other VLANs, regardless of which VIP card receives the traffic.

These commands show the configuration for each of the VLANs shown in the figure above:

```
interface FastEthernet1/0/0
 ip address 10.1.1.1 255.255.255.0
 ip route-cache distributed
 full-duplex
```

```
interface FastEthernet1/0/0.1
 ip address 10.1.1.1 255.255.255.0
 encapsulation isl 1
interface FastEthernet1/0/0.2
 ip address 10.1.2.1 255.255.255.0
 encapsulation isl 2
interface FastEthernet1/0/0.3
 ip address 10.1.3.1 255.255.255.0
 encapsulation isl 3
interface FastEthernet1/1/0
 ip route-cache distributed
 full-duplex
interface FastEthernet1/1/0.1
 ip address 172.16.1.1 255.255.255.0
 encapsulation isl 4
interface Fast Ethernet 2/0/0
 ip address 10.1.1.1 255.255.255.0
 ip route-cache distributed
 full-duplex
interface FastEthernet2/0/0.5
 ip address 10.2.1.1 255.255.255.0
 encapsulation isl 5
interface FastEthernet2/1/0
 ip address 10.3.1.1 255.255.255.0
 ip route-cache distributed
 full-duplex
interface FastEthernet2/1/0.6
 ip address 10.4.6.1 255.255.255.0
 encapsulation isl 6
interface FastEthernet2/1/0.7
 ip address 10.4.7.1 255.255.255.0
 encapsulation isl 7
```

XNS Routing over ISL Configuration Example

To configure routing of the XNS protocol over ISL trunks, you need to define ISL as the encapsulation type. This example shows XNS configured to be routed over an ISL trunk:

```
xns routing 0123.4567.adcb
interface fastethernet 1/0.1
 encapsulation isl 100
 xns network 20
```

CLNS Routing over ISL Configuration Example

To configure routing of the CLNS protocol over ISL trunks, you need to define ISL as the encapsulation type. This example shows CLNS configured to be routed over an ISL trunk:

```
clns routing
interface fastethernet 1/0.1
 encapsulation isl 100
 clns enable
```

IS-IS Routing over ISL Configuration Example

To configure IS-IS routing over ISL trunks, you need to define ISL as the encapsulation type. This example shows IS-IS configured over an ISL trunk:

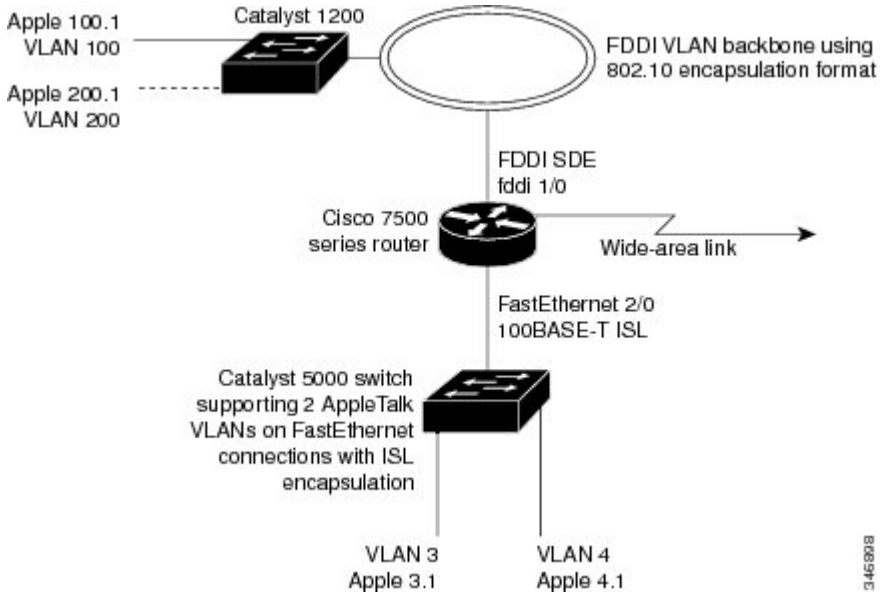
```
isis routing test-proc2
```

```
net 49.0001.0002.aaaa.aaaa.aaaa.00
interface fastethernet 2.0
 encapsulation isl 101
 clns router is-is test-proc2
```

Routing IEEE 802.10 Configuration Example

The figure below shows AppleTalk being routed between different ISL and IEEE 802.10 VLAN encapsulating subinterfaces.

Figure 22: Routing AppleTalk over VLAN encapsulations



As shown in the figure above, AppleTalk traffic is routed to and from switched VLAN domains 3, 4, 100, and 200 to any other AppleTalk routing interface. This example shows a sample configuration file for the Cisco 7500 series router with the commands entered to configure the network shown in the figure above.

Cisco 7500 Router Configuration

```
!
interface Fddi 1/0.100
 encapsulation sde 100
 appletalk cable-range 100-100 100.2
 appletalk zone 100
!
interface Fddi 1/0.200
 encapsulation sde 200
 appletalk cable-range 200-200 200.2
 appletalk zone 200
!
interface FastEthernet 2/0.3
 encapsulation isl 3
 appletalk cable-range 3-3 3.2
 appletalk zone 3
!
interface FastEthernet 2/0.4
 encapsulation isl 4
```

```
 appletalk cable-range 4-4 4.2
 appletalk zone 4
!
```

IEEE 802.1Q Encapsulation Configuration Examples

Configuration examples for each protocols are provided in the following sections:

Configuring AppleTalk over IEEE 802.1Q Example

This configuration example shows AppleTalk being routed on VLAN 100:

```
!
appletalk routing
!
interface fastethernet 4/1.100
  encapsulation dot1q 100
  appletalk cable-range 100-100 100.1
  appletalk zone eng
!
```

Configuring IP Routing over IEEE 802.1Q Example

This configuration example shows IP being routed on VLAN 101:

```
!
ip routing
!
interface fastethernet 4/1.101
  encapsulation dot1q 101
  ip addr 10.0.0.11 255.0.0.0
!
```

Configuring IPX Routing over IEEE 802.1Q Example

This configuration example shows IPX being routed on VLAN 102:

```
!
ipx routing
!
interface fastethernet 4/1.102
  encapsulation dot1q 102
  ipx network 100
!
```

VLAN 100 for Bridge Group 1 with Default VLAN1 Example

The following example configures VLAN 100 for bridge group 1 with a default VLAN1:

```
interface FastEthernet 4/1.100
encapsulation dot1q 1
bridge-group 1
```

VLAN 20 for Bridge Group 1 with Native VLAN Example

The following example configures VLAN 20 for bridge group 1 as a native VLAN:

```
interface FastEthernet 4/1.100
encapsulation dot1q 20 native
bridge-group 1
```

VLAN ISL or IEEE 802.1Q Routing Example

The following example configures VLAN ISL or IEEE 802.10 routing:

```
ipx routing
appletalk routing
!
interface Ethernet 1
ip address 10.1.1.1 255.255.255.0
appletalk cable-range 1-1 1.1
appletalk zone 1
ipx network 10 encapsulation snap
!
router igrp 1
network 10.1.0.0
!
end
!
#Catalyst5000
!
set VLAN 110 2/1
set VLAN 120 2/2
!
set trunk 1/1 110,120
# if 802.1Q, set trunk 1/1 nonegotiate 110, 120
!
end
!
ipx routing
appletalk routing
!
interface FastEthernet 1/1.110
encapsulation isl 110
!if 802.1Q, encapsulation dot1Q 110
ip address 10.1.1.2 255.255.255.0
appletalk cable-range 1.1 1.2
appletalk zone 1
ipx network 110 encapsulation snap
!
interface FastEthernet 1/1.120
encapsulation isl 120
!if 802.1Q, encapsulation dot1Q 120
ip address 10.2.1.2 255.255.255.0
appletalk cable-range 2-2 2.2
appletalk zone 2
ipx network 120 encapsulation snap
!
router igrp 1
network 10.1.0.0
network 10.2.1.0.0
!
end
!
ipx routing
appletalk routing
!
interface Ethernet 1
ip address 10.2.1.3 255.255.255.0
```

```
appletalk cable-range 2-2 2.3
appletalk zone 2
ipx network 120 encapsulation snap
!
router igrp 1
network 10.2.0.0
!
end
```

VLAN IEEE 802.1Q Bridging Example

The following examples configures IEEE 802.1Q bridging:

```
interface FastEthernet4/0
 no ip address
 no ip route-cache
 half-duplex
!
interface FastEthernet4/0.100
 encapsulation dot1Q 100
 no ip route-cache
 bridge-group 1
!
interface FastEthernet4/0.200
 encapsulation dot1Q 200 native
 no ip route-cache
 bridge-group 2
!
interface FastEthernet4/0.300
 encapsulation dot1Q 1
 no ip route-cache
 bridge-group 3
!
interface FastEthernet10/0
 no ip address
 no ip route-cache
 half-duplex
!
interface FastEthernet10/0.100
 encapsulation dot1Q 100
 no ip route-cache
 bridge-group 1
!
interface Ethernet11/3
 no ip address
 no ip route-cache
 bridge-group 2
!
interface Ethernet11/4
 no ip address
 no ip route-cache
 bridge-group 3
!
bridge 1 protocol ieee
bridge 2 protocol ieee
bridge 3 protocol ieee
```

VLAN IEEE 802.1Q IRB Example

The following examples configures IEEE 802.1Q integrated routing and bridging:

```
ip cef
appletalk routing
ipx routing 0060.2f27.5980
!
bridge irb
!
interface TokenRing3/1
 no ip address
 ring-speed 16
 bridge-group 2
!
interface FastEthernet4/0
 no ip address
 half-duplex
!
interface FastEthernet4/0.100
 encapsulation dot1Q 100
 bridge-group 1
!
interface FastEthernet4/0.200
 encapsulation dot1Q 200
 bridge-group 2
!
interface FastEthernet10/0
ip address 10.3.1.10 255.255.255.0
 half-duplex
 appletalk cable-range 200-200 200.10
 appletalk zone irb
 ipx network 200
!
interface Ethernet11/3
 no ip address
 bridge-group 1
!
interface BVI 1
 ip address 10.1.1.11 255.255.255.0
 appletalk cable-range 100-100 100.11
 appletalk zone bridging
 ipx network 100
!
router rip
 network 10.0.0.0
 network 10.3.0.0
!
bridge 1 protocol ieee
 bridge 1 route appletalk
 bridge 1 route ip
 bridge 1 route ipx
bridge 2 protocol ieee
!
```

Configuring IEEE 802.1Q-in-Q VLAN Tag Termination Example

Some ambiguous subinterfaces can use the **any** keyword for the inner VLAN ID specification. The **any** keyword represents any inner VLAN ID that is not explicitly configured on any other interface. In the following example, seven subinterfaces are configured with various outer and inner VLAN IDs.

Note The **any** keyword can be configured on only one subinterface of a specified physical interface and outer VLAN ID.

```
interface GigabitEthernet1/0/0.1
 encapsulation dot1q 100 second-dot1q 100
interface GigabitEthernet1/0/0.2
 encapsulation dot1q 100 second-dot1q 200
interface GigabitEthernet1/0/0.3
 encapsulation dot1q 100 second-dot1q 300-400,500-600
interface GigabitEthernet1/0/0.4
 encapsulation dot1q 100 second-dot1q any
interface GigabitEthernet1/0/0.5
 encapsulation dot1q 200 second-dot1q 50
interface GigabitEthernet1/0/0.6
 encapsulation dot1q 200 second-dot1q 1000-2000,3000-4000
interface GigabitEthernet1/0/0.7
 encapsulation dot1q 200 second-dot1q any
```

The table below shows which subinterfaces are mapped to different values of the outer and inner VLAN ID on Q-in-Q frames that come in on Gigabit Ethernet interface 1/0/0.

Table 9: Subinterfaces Mapped to Outer and Inner VLAN IDs for GE Interface 1/0/0

Outer VLAN ID	Inner VLAN ID	Subinterface mapped to
100	1 through 99	GigabitEthernet1/0/0.4
100	100	GigabitEthernet1/0/0.1
100	101 through 199	GigabitEthernet1/0/0.4
100	200	GigabitEthernet1/0/0.2
100	201 through 299	GigabitEthernet1/0/0.4
100	300 through 400	GigabitEthernet1/0/0.3
100	401 through 499	GigabitEthernet1/0/0.4
100	500 through 600	GigabitEthernet1/0/0.3
100	601 through 4095	GigabitEthernet1/0/0.4
200	1 through 49	GigabitEthernet1/0/0.7
200	50	GigabitEthernet1/0/0.5
200	51 through 999	GigabitEthernet1/0/0.7
200	1000 through 2000	GigabitEthernet1/0/0.6
200	2001 through 2999	GigabitEthernet1/0/0.7
200	3000 through 4000	GigabitEthernet1/0/0.6

Outer VLAN ID	Inner VLAN ID	Subinterface mapped to
200	4001 through 4095	GigabitEthernet1/0/0.7

A new subinterface is now configured:

```
interface GigabitEthernet1/0/0.8
 encapsulation dot1q 200 second-dot1q 200-600,900-999
```

The table below shows the changes made to the table for the outer VLAN ID of 200. Notice that subinterface 1/0/0.7 configured with the **any** keyword now has new inner VLAN ID mappings.

Table 10: Subinterfaces Mapped to Outer and Inner VLAN IDs for GE Interface 1/0/0--Changes Resulting from Configuring GE Subinterface 1/0/0.8

Outer VLAN ID	Inner VLAN ID	Subinterface mapped to
200	1 through 49	GigabitEthernet1/0/0.7
200	50	GigabitEthernet1/0/0.5
200	51 through 199	GigabitEthernet1/0/0.7
200	200 through 600	GigabitEthernet1/0/0.8
200	601 through 899	GigabitEthernet1/0/0.7
200	900 through 999	GigabitEthernet1/0/0.8
200	1000 through 2000	GigabitEthernet1/0/0.6
200	2001 through 2999	GigabitEthernet1/0/0.7
200	3000 through 4000	GigabitEthernet1/0/0.6
200	4001 through 4095	GigabitEthernet1/0/0.7

Additional References

The following sections provide references related to the Managed LAN Switch feature.

Related Documents

Related Topic	Document Title
IP LAN switching commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS LAN Switching Services Command Reference
LAN switching	“LAN Switching” module of the *Internetworking Technology Handbook*

Standards

Standards	Title
No new or modified RFCs are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing standards has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Routing Between VLANs

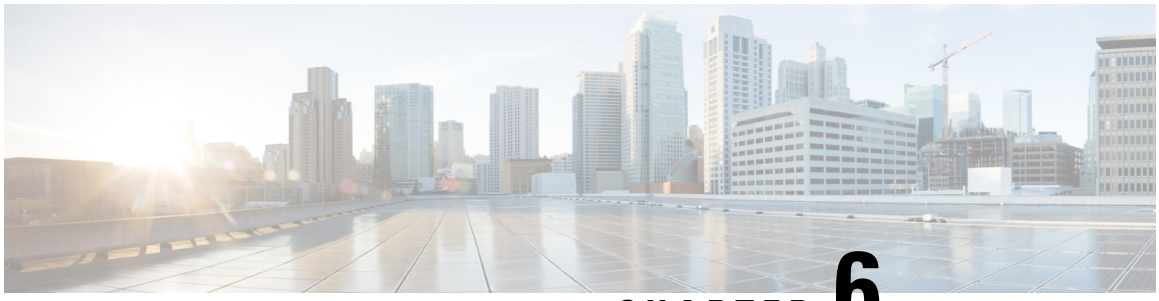
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 11: Feature Information for Routing Between VLANs

Feature Name	Releases	Feature Information
IEEE 802.1Q-in-Q VLAN Tag Termination	12.0(28)S, 12.3(7)(X17) 12.0(32)S1, 12.2(31)SB 12.3(7)T 12.3((7)XI1	Encapsulating IEEE 802.1Q VLAN tags within 802.1Q enables service providers to use a single VLAN to support customers who have multiple VLANs. The IEEE 802.1Q-in-Q VLAN Tag Termination feature on the subinterface level preserves VLAN IDs and keeps traffic in different customer VLANs segregated.
Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation	12.0(7)XE 12.1(5)T 12.2(2)DD 12.2(4)B 12.2(8)T 12.2(13)T Cisco IOS XE 3.8(S) Cisco IOS XE 3.9(S)	The IEEE 802.1Q protocol is used to interconnect multiple switches and routers, and for defining VLAN topologies. The IEEE 802.1Q standard is extremely restrictive to untagged frames. The standard provides only a per-port VLANs solution for untagged frames. For example, assigning untagged frames to VLANs takes into consideration only the port from which they have been received. Each port has a parameter called a *permanent virtual identification* (Native VLAN) that specifies the VLAN assigned to receive untagged frames. In Cisco IOS XE Release 3.8(S), support was added for the Cisco ISR 4400 Series Routers. In Cisco IOS XE Release 3.9(S), support was added for the Cisco CSR 1000V Series Routers.
Configuring Routing Between VLANs with Inter-Switch Link Encapsulation	12.0(7)XE 12.1(5)T 12.2(2)DD 12.2(4)B 12.2(8)T 12.2(13)T	ISL is a Cisco protocol for interconnecting multiple switches and maintaining VLAN information as traffic goes between switches. ISL provides VLAN capabilities while maintaining full wire speed performance on Fast Ethernet links in full- or half-duplex mode. ISL operates in a point-to-point environment and will support up to 1000 VLANs. You can define virtually as many logical networks as are necessary for your environment.
Configuring Routing Between VLANs with IEEE 802.10 Encapsulation	12.0(7)XE 12.1(5)T 12.2(2)DD 12.2(4)B 12.2(8)T 12.2(13)T	AppleTalk can be routed over VLAN subinterfaces using the ISL or IEEE 802.10 VLANs feature that provides full-feature Cisco IOS software AppleTalk support on a per-VLAN basis, allowing standard AppleTalk capabilities to be configured on VLANs.

Feature Name	Releases	Feature Information
VLAN Range	12.0(7)XE 12.1(5)T 12.2(2)DD 12.2(4)B 12.2(8)T 12.2(13)T	Using the VLAN Range feature, you can group VLAN subinterfaces together so that any command entered in a group applies to every subinterface within the group. This capability simplifies configurations and reduces command parsing. In Cisco IOS Release 12.0(7)XE, the **interface range** command was introduced. The **interface range** command was integrated into Cisco IOS Release 12.1(5)T. In Cisco IOS Release 12.2(2)DD, the **interface range** command was expanded to enable configuration of subinterfaces. The **interface range** command was integrated into Cisco IOS Release 12.2(4)B. The VLAN Range feature was integrated into Cisco IOS Release 12.2(8)T. This VLAN Range feature was integrated into Cisco IOS Release 12.2(13)T.
256+ VLANS	12.1(2)E, 12.2(8)T Cisco IOS XE 3.8(S) Cisco IOS XE 3.9(S)	The 256+ VLAN feature enables a device to route more than 256 VLAN interfaces. This feature requires the MSFC2. The routed VLAN interfaces can be chosen from any of the VLANs supported on the device. Catalyst switches can support up to 4096 VLANs. If MSFC is used, up to 256 VLANs can be routed, but this can be selected from any VLANs supported on the device. In Cisco IOS XE Release 3.8(S), support was added for the Cisco ISR 4400 Series Routers. In Cisco IOS XE Release 3.9(S), support was added for the Cisco CSR 1000V Series Routers.



CHAPTER 6

EtherChannel Flow-Based Limited 1 1 Redundancy

EtherChannel flow-based limited 1:1 redundancy provides MAC, or layer 2, traffic protection to avoid higher layer protocols from reacting to single link failures and re-converging. To use EtherChannel flow-based limited 1:1 redundancy, you configure an EtherChannel with two ports (one active and one standby). If the active link goes down, the EtherChannel stays up and the system performs fast switchover to the hot-standby link. Depending on how you have the priorities set, when the failed link becomes operational again, the EtherChannel performs another fast switchover to revert to the original active link. if all port-priorities are the same, it will not revert, but remain on the current active link.

With 1:1 redundancy configured, only one link is active at any given time so all flows are directed over the active link.

- Restrictions for EtherChannel Flow-based Limited 1:1 Redundancy, on page 117
- Information About EtherChannel Flow-Based Limited 1 1 Redundancy, on page 118
- How to Configure EtherChannel Flow-Based Limited 1 1 Redundancy, on page 118
- Configuration Examples for EtherChannel Flow-Based Limited1 1 Redundancy, on page 123
- Additional References, on page 124
- Feature Information for EtherChannel Flow-based Limited 1 1 Redundancy, on page 125

Restrictions for EtherChannel Flow-based Limited 1:1 Redundancy

When you are using the Cisco ASR 1001-X, the following restrictions apply for collecting traffic statistics for VLAN egress on sub-interfaces. Obtaining input/output counters using SNMP is unsupported. This is because the Cisco ASR 1001-X has a built-in SPA.

Restrictions that apply when obtaining traffic statistics for two types of interfaces are shown below:

- **Physical sub-interfaces**

  For the Cisco ASR 1001-X, statistics for the VLAN egress are available for physical sub-interfaces. The output counter is used from cpp, not from the built-in SPA hardware. To show VLAN egress statistics, use the **show vlans** *vlan id* command.

  Example

```
# show vlans 10
VLAN ID: 10 (IEEE 802.1Q Encapsulation)
>
>    Protocols Configured:          Received:        Transmitted:
>                     IP             133                104
```

- **Port Channel sub-interfaces**

  For the Cisco ASR 1001-X, showing traffic statistics for the VLAN egress is not supported for port channel sub-interfaces.

  cpp or the built-in SPA can not be used to give an output counter value for port channel sub-interfaces.

Information About EtherChannel Flow-Based Limited 1 1 Redundancy

EtherChannel Flow-Based Limited 1 1 Redundancy

EtherChannel flow-based limited 1:1 redundancy provides an EtherChannel configuration with one active link and fast switchover to a hot standby link. To use EtherChannel flow-based limited 1:1 redundancy, you configure a Link Aggregation Control Protocol (LACP) EtherChannel with two ports (one active and one standby). If the active link goes down, the EtherChannel stays up and the system performs fast switchover to the hot standby link. Depending on how the priorities of the links are set, when the failed link becomes operational again, the EtherChannel performs another fast switchover to revert to the original active link, or to the link with the higher priority.

For EtherChannel flow-based limited 1:1 redundancy to work correctly (especially the fast switchover capability) the feature must be enabled at both ends of the link.

How to Configure EtherChannel Flow-Based Limited 1 1 Redundancy

Configuring EtherChannel Flow-Based Limited 1 1 Redundancy with Fast-Switchover

To configure an LACP EtherChannel with two ports (one active and one standby), perform the following steps. This feature must be enabled at both ends of the link.

You can control which link is the primary active link by setting the port priority on the links used for the redundancy. To configure a primary link and enable the EtherChannel to revert to the original link, one link must have a higher port priority than the other and the LACP max-bundle must be set to 1. This configuration results in link 1 being active and link 2 being in hot standby state.

To prevent the switchover to revert, you can assign both links the same priority.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *channel -number*
4. **lacp fast-switchover**
5. **lacp max-bundle 1**
6. **exit**
7. **interface tengigabitethernet** *slot / port / number*
8. **channel-group 1 mode** *mode*
9. **lacp port-priority** *priority*
10. **exit**
11. **interface tengigabitethernet** *slot / port / number*
12. **channel-group 1 mode** *mode*
13. **lacp port-priority** *priority*
14. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *channel -number* **Example:** `Router(config)# interface port-channel 1`	Selects an LACP port channel interface.
Step 4	**lacp fast-switchover** **Example:** `Router(config-if)# lacp fast-switchover`	Enables the fast switchover feature for this EtherChannel.
Step 5	**lacp max-bundle 1** **Example:** `Router(config-if)# lacp max-bundle 14`	Sets the maximum number of active member ports to 14. **Note** For Cisco ASR 1000 Series Aggregation Services Routers, the minimum number of active member ports is 1 and the maximum number is 14.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**interface tengigabitethernet** *slot / port / number* **Example:** `Router(config)# interface tengigabitethernet 0/0/0`	Selects the first interface to add to the port channel.
Step 8	**channel-group 1 mode** *mode* **Example:** `Router(config-if)# channel-group 1 mode active`	Adds the member link to the port-channel and actively participates in LACP negotiation.
Step 9	**lacp port-priority** *priority* **Example:** `Router(config-if)# lacp port-priority 32768`	Sets the priority on the port-channel. This priority is set to the default value.
Step 10	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 11	**interface tengigabitethernet** *slot / port / number* **Example:** `Router(config)# interface tengigabitethernet 1/0/0`	Selects the interface to add to the port channel.
Step 12	**channel-group 1 mode** *mode* **Example:** `Router(config-if)# channel-group 1 mode active`	Adds the member link to the port-channel and actively participates in LACP negotiation.
Step 13	**lacp port-priority** *priority* **Example:** `Router(config-if)# lacp port-priority 32767`	Sets the port priority higher than the other link by using a value lower than the default value of 32768. This forces this link to be the active link whenever it is capable of carrying traffic.
Step 14	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode.

Setting the Switchover Rate with Carrier Delay

Optionally, you can control the speed of the switchover between the active and standby links by setting the carrier delay on each link. The **carrier-delay** command controls how long it takes for Cisco IOS to propagate the information about the links status to other modules.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tengigabitethernet** *slot* / *port* / *number*
4. **carrier-delay msec** *msec*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface tengigabitethernet** *slot* / *port* / *number* **Example:** `Router(config)# interface tengigabitethernet 0/1/0`	Enters interface configuration mode and opens the configuration for the specified interface.
Step 4	**carrier-delay msec** *msec* **Example:** `Router(config-if)# carrier-delay msec 11`	Sets how long it takes to propagate the link status to other modules.
Step 5	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode.

Verifying EtherChannel Flow-Based Limited 1 1 Redundancy

Use these show commands to verify the configuration and to display information about the port channel.

SUMMARY STEPS

1. **enable**
2. **show running-config interface** *type slot / port / number*
3. **show interfaces port-channel** *channel-number* **etherchannel**
4. **show etherchannel** *channel-number* **port-channel**
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show running-config interface** *type slot / port / number* **Example:** `Router# show running-config interface tengigabitethernet 0/0/0`	Verifies the configuration. • *type* --**gigabitethernet** or **tengigabitethernet**.
Step 3	**show interfaces port-channel** *channel-number* **etherchannel** **Example:** `Router# show interfaces port-channel 1 etherchannel`	Displays the bucket distribution currently in use.
Step 4	**show etherchannel** *channel-number* **port-channel** **Example:** `Router# show etherchannel 1 port-channel`	Displays the port channel fast-switchover feature capability.
Step 5	**end** **Example:** `Router# end`	Exits privileged EXEC mode.

Configuration Examples for EtherChannel Flow-Based Limited1 1 Redundancy

EtherChannel 1 1 Active Standby Example

This example shows how to configure a port channel for 1:1 link redundancy for equal priority ports so there is no preference which port is active.

```
Router# enable
Router# configure terminal
Router(config)# interface port-channel 2
Router(config-if)# ip address 10.1.1.1 255.255.0.0
Router(config-if)# negotiation auto
Router(config-if)# lacp max-bundle 1
Router(config-if)# lacp fast-switchover
Router(config)# interface Tengigabitethernet0/1/0
Router(config-if)# channel-group 2 mode active
Router(config-if)# negotiation auto
Router(config)# interface Tengigabitethernet 2/1/0
Router(config-if)# channel-group 2 mode active
Router(config-if)# negotiation auto
Router(config)# interface GigabitEthernet0/1/6
Router(config-if)# negotiation auto
Router(config-if)# channel-group 19 mode active
Router(config)# interface GigabitEthernet0/1/7
Router(config-if)# negotiation auto
Router(config-if)# channel-group 19 mode active
Router(config-if)# interface Port-channel19
Router(config-if)# ip address 10.19.1.1 255.255.255.0
Router(config-if)# no negotiation auto
Router(config-if)# lacp fast-switchover
Router(config-if)# lacp max-bundle 1
Router(config-if)# end
```

Notice in the **show** command display the priorities are the same value.

```
Router# show lacp internal
Flags: S - Device is requesting Slow LACPDUs
 F - Device is requesting Fast LACPDUs
 A - Device is in Active mode P - Device is in Passive mode
Channel group 19
LACP port Admin Oper Port Port
Port Flags State Priority Key Key Number State
Gi0/1/6 SA bndl 32768 0x13 0x13 0x47 0x3D
Gi0/1/7 FA hot-sby 32768 0x13 0x13 0x48 0x7
```

Setting Priority for 1 1 Redundancy Using LACP Example

This example shows how to configure an LACP EtherChannel with 1:1 redundancy. GigabitEthernet 0/1/7 is the active link, because it is configured with a lower number which give it a higher port priority.

```
Router# configure terminal
Router(config)# interface GigabitEthernet0/1/6
```

```
Router(config-if)# lacp port-priority 32767
Router(config-if)# exit
Router(config)# interface GigabitEthernet0/1/7
Router(config-if)# lacp fast-switchover
Router(config-if)# lacp max-bundle 1
Router(config-if)# negotiation auto
Router(config-if)# channel-group 19 mode active
```

In this show display, notice that the bundled link is set at a higher priority. This will ensure that the bundled link is used as the first active link in the standby configuration.

```
Router# show lacp internal

Flags: S - Device is requesting Slow LACPDUs
 F - Device is requesting Fast LACPDUs
 A - Device is in Active mode P - Device is in Passive mode
Channel group 19
LACP port Admin Oper Port Port
Port Flags State Priority Key Key Number State
Gi0/1/6 FA hot-sby 32768 0x13 0x13 0x47 0x7
Gi0/1/7 SA bndl 32767 0x13 0x13 0x48 0x3D
```

Additional References

The following sections provide references related to the EtherChannel Flow-based Limited1:1 Redundancy feature.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
LAN Switching commands	*Cisco IOS LAN Switching Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing standards has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

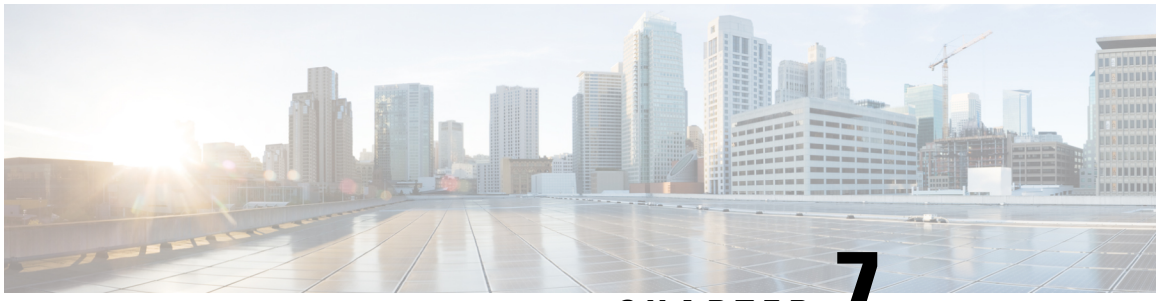
Feature Information for EtherChannel Flow-based Limited 1 1 Redundancy

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 12: Feature Information for EtherChannel Flow-based Limited 1:1 Redundancy

Feature Name	Releases	Feature Information
EtherChannel Flow-Based Limited 1:1 Redundancy	Cisco IOS XE Release 2.4	EtherChannel flow-based limited 1:1 redundancy provides MAC, or layer 2, traffic protection to avoid higher layer protocols from reacting to single link failures and re-converging. To use EtherChannel flow-based limited 1:1 redundancy, you configure an EtherChannel with two ports (one active and one standby). If the active link goes down, the EtherChannel stays up and the system performs fast switchover to the hot-standby link. Depending on how you have the priorities set, when the failed link becomes operational again, the EtherChannel performs another fast switchover to revert to the original active link. if all port-priorities are the same, it will not revert, but remain on the current active link. No commands were modified or created to support this feature.



CHAPTER 7

Flow-Based per Port-Channel Load Balancing

The Flow-Based per Port-Channel Load Balancing feature allows different flows of traffic over a Gigabit EtherChannel (GEC) interface to be identified based on the packet header and then mapped to the different member links of the port channel. This feature enables you to apply flow-based load balancing and VLAN-manual load balancing to specific port channels.

- Restrictions for Flow-Based per Port-Channel Load Balancing, on page 127
- Information About Flow-Based per Port-Channel Load Balancing, on page 127
- How to Enable Flow-Based per Port-Channel Load Balancing, on page 130
- Configuration Examples for Flow-Based per Port-Channel Load Balancing, on page 133
- Information About Five-Tuple Hash Support for GEC Flow-based Load Balancing, on page 134
- Additional References, on page 134
- Feature Information for Flow-Based per Port-Channel Load Balancing, on page 136

Restrictions for Flow-Based per Port-Channel Load Balancing

- Supports up to 64 GEC interfaces.
- Supports up to 14 member links per GEC interface.



Note This feature achieves load balancing of MPLS traffic only by using source IP address and destination IP address. The MPLS label is not considered for load balancing.

Information About Flow-Based per Port-Channel Load Balancing

Flow-Based Load Balancing

Flow-based load balancing identifies different flows of traffic based on the key fields in the data packet. For example, IPv4 source and destination IP addressees can be used to identify a flow. The various data traffic flows are then mapped to the different member links of a port channel. After the mapping is done, the data traffic for a flow is transmitted through the assigned member link. The flow mapping is dynamic and changes when there is any change in the state of a member link to which a flow is assigned. The flow mappings can

also change if member links are added to or removed from the GEC interface. Multiple flows can be mapped to each member link.

Buckets for Flow-Based Load Balancing

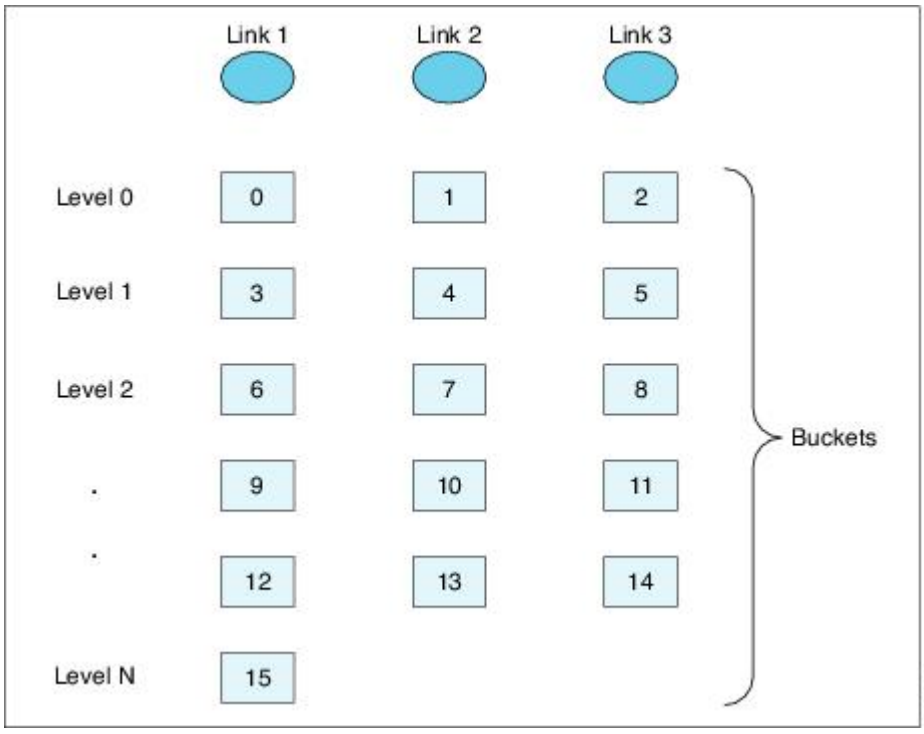
Load balancing dynamically maps traffic flows to the member links of a GEC interface through the concept of buckets. The various defined traffic flows are mapped to the buckets and the buckets are evenly distributed among the member links. Each port channel maintains 16 buckets, with one active member link associated with each bucket. All traffic flows mapped to a bucket use the member link to which the bucket is assigned.

The router creates the buckets-to-member links mappings when you apply flow-based load balancing to a port channel and the port channel has at least one active member link. The mappings are also created when the first member link is added, or comes up, and the load-balancing method is set to flow-based.

When a member link goes down or is removed from a port channel, the buckets associated with that member link are redistributed among the other active member links in a round-robin fashion. When a member link comes up or is added to a port channel, some of the buckets associated with other links are assigned to this link.

The figure below illustrates an example of 16 buckets distributed among three member links. The numbers shown in the buckets are the bucket IDs. Note that the first member link has an extra bucket.

Figure 23: Example of 16 Buckets Mapped to Three Member Links



If you change the load-balancing method, the bucket-to-member link mappings for flow-based load balancing are deleted. The mappings are also deleted if the port channel is deleted or the last member link in the port channel is deleted or goes down.

Load Balancing on Port Channels

GEC interfaces can use either dynamic flow-based load balancing or VLAN-manual load balancing. You can configure the load-balancing method globally for all port channels or directly on specific port channels. The global configuration applies only to those port channels for which you have not explicitly configured load balancing. The port-channel configuration overrides the global configuration.

Flow-based load balancing is enabled by default at the global level. You must explicitly configure VLAN load balancing or the load-balancing method is flow-based.

For more information about configuring VLAN load balancing, see the module VLAN Mapping to Gigabit EtherChannel (GEC) Member Links.

The table below lists the load-balancing method that is applied to port channels based on the configuration:

Table 13: Flow-Based Load Balancing Configuration Options

Global Configuration	Port-Channel Configuration	Load Balancing Applied
Not configured	Not configured	Flow-based
	Flow-based	Flow-based
	VLAN-manual	VLAN-manual
VLAN-manual	Not configured	VLAN-manual
	Flow-based	Flow-based
	VLAN-manual	VLAN-manual

The table below lists the configuration that results if you change the global load-balancing method.

Table 14: Results When Global Configuration Changes

Port-Channel Configuration	Global Configuration	Action Taken at Port Channel	
–	From	To	–
Not configured	Not configured	VLAN-manual	Changed from flow-based to VLAN-manual
	VLAN-manual	Not configured	Changed from VLAN-manual to flow-based
Configured	Any	Any	No change

The table below lists the configuration that results if you change the port-channel load-balancing method.

Table 15: Results When Port-Channel Configuration Changes

Global Configuration	Port-Channel Configuration	Action Taken at Port Channel	
–	From	To	–

Global Configuration	Port-Channel Configuration	Action Taken at Port Channel	
Not configured	Not configured	VLAN-manual	Changed from flow-based to VLAN-manual
	Not configured	Flow-based	No action taken
	VLAN-manual	Flow-based	Changed from VLAN-manual to flow-based
	VLAN-manual	Not configured	Changed from VLAN-manual to flow-based
	Flow-based	VLAN-manual	Changed from flow-based to VLAN-manual
	Flow-based	Not configured	No action taken
VLAN-manual	Not configured	VLAN-manual	No action taken
	Not configured	Flow-based	Changed from VLAN-manual to flow-based
	VLAN-manual	Flow-based	Changed from VLAN-manual to flow-based
	VLAN-manual	Not configured	No action taken
	Flow-based	VLAN-manual	Changed from flow-based to VLAN-manual
	Flow-based	Not configured	Changed from flow-based to VLAN-manual

How to Enable Flow-Based per Port-Channel Load Balancing

Configuring Load Balancing on a Port Channel

To configure load balancing on a port channel, perform the following steps. Repeat these steps for each GEC interface.

Before you begin

If you have already configured your desired load-balancing method globally and want to use that method for all port channels, you need not perform this task. To configure load balancing globally, use the **port-channel load-balancing vlan-manual** command. If you do not configure the global command, flow-based load balancing is applied to all port channels.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *channel-number*
4. **load-balancing** {**flow** | **vlan**}
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *channel-number* **Example:** `Router(config)# interface port-channel 1`	Enters interface configuration mode and defines the interface as a port channel.
Step 4	**load-balancing** {**flow**	**vlan**} **Example:** `Router(config-if)# load-balancing flow`	Applies a load-balancing method to the specific port channel. • If you do not configure this command, the port channel uses the global load-balancing method configured with the **port-channel load-balancing vlan-manual** command. The global default is flow-based.
Step 5	**end** **Example:** `Router(config-if)# end`	Exits configuration mode.

Verifying Load-Balancing Configuration on a GEC Interface

Use these show commands to verify the load-balancing configuration and to display information about the bucket distribution on the port channel. You can use these commands in any order.

SUMMARY STEPS

1. **show running-config interface port-channel** *channel-number*
2. **show etherchannel load-balancing**
3. **show interfaces port-channel** *channel-number* **etherchannel**

DETAILED STEPS

Procedure

Step 1 **show running-config interface port-channel** *channel-number*

Use this command to verify the configuration of the port channel.

Example:

```
Router# show running-config interface port-channel 1
Building configuration...

Current configuration : 88 bytes
!
interface Port-channel1
 ip address 10.1.1.1 255.0.0.0
 no negotiation auto
 load-balancing flow
end
```

Step 2 **show etherchannel load-balancing**

Use this command to display the load-balancing method applied to each port channel. The following example shows output for a configuration with load balancing set globally to VLAN-manual and set to flow-based on port channel 1:

Example:

```
Router# show etherchannel load-balancing


EtherChannel Load-Balancing Method:
Global LB Method: vlan-manual

  Port-Channel:                      LB Method
    Port-channel1                  :  flow-based
```

Step 3 **show interfaces port-channel** *channel-number* **etherchannel**

Use this command to display the bucket distribution currently in use. The following example shows output for an interface with load balancing set to flow-based:

Example:

```
Router(config)# show interface port-channel 2 etherchannel

 All IDBs List contains 3 configured interfaces
  Port: GigabitEthernet2/1/6 (index: 0)
  Port: GigabitEthernet2/1/7 (index: 1)
  Port: GigabitEthernet2/1/0 (index: 2)
```

```
Active Member List contains 1 interfaces
 Port: GigabitEthernet2/1/0

Passive Member List contains 2 interfaces
 Port: GigabitEthernet2/1/6

 Port: GigabitEthernet2/1/7

Load-Balancing method applied: flow-based

Bucket Information for Flow-Based LB:
Interface:                                Buckets
   GigabitEthernet2/1/0:
                          Bucket 0 , Bucket 1 , Bucket 2 , Bucket 3
                          Bucket 4 , Bucket 5 , Bucket 6 , Bucket 7
                          Bucket 8 , Bucket 9 , Bucket 10, Bucket 11
                          Bucket 12, Bucket 13, Bucket 14, Bucket 15
```

Configuration Examples for Flow-Based per Port-Channel Load Balancing

Flow-Based Load Balancing Example

The following example shows a configuration where flow-based load balancing is configured on port-channel 2 while the VLAN-manual method is configured globally:

```
!
no aaa new-model
port-channel load-balancing vlan-manual
ip source-route
.
.
.
interface Port-channel2
 ip address 10.0.0.1 255.255.255.0
 no negotiation auto
 load-balancing flow
!
interface Port-channel2.10
 ip rsvp authentication key 11223344
 ip rsvp authentication
!
interface Port-channel2.50
 encapsulation dot1Q 50
!
interface GigabitEthernet2/1/0
 no ip address
 negotiation auto
 cdp enable
 channel-group 2
!
```

Information About Five-Tuple Hash Support for GEC Flow-based Load Balancing

The five-tuple hash support for gigabit etherchannel (GEC) flow-based load balancing feature decides which member link to use for routing traffic based on the following five parameters:

- Source IP address
- Destination IP address
- Source Port
- Destination Port
- Protocol ID (type of protocol: TCP/UDP)

Earlier, the GEC flow-based load balancing feature was applicable only for layer 3 (network layer). With the five-tuple hash support, it's applicable for layer 4 (TCP/IP layer) also. But it is supported only for the TCP and UDP, layer 4 protocols.

Restrictions for Five-Tuple Hash Support for GEC Flow-based Load Balancing

The five-tuple hash support for GEC flow-based load balancing feature is not supported for MPLS traffic.

Configuring Five-Tuple Hash Support for GEC Flow-based Load Balancing

Use the **port-channel load-balance-hash-algo** command to enable the five-tuple hash support for GEC flow-based load balancing feature.

The following example shows how to configure a five-tuple hash support for GEC flow-based load balancing feature:

```
Device (config)# port-channel load-balance-hash-algo ?

src-dst-ip Source XOR Destination IP Addr
src-dst-mixed-ip-port Source XOR Destination Port, IP addr
```

The **src-dst-mixed-ip-port** option specifies load distribution based on the hash value obtained from the calculation of five parameters: source ip address, destination ip address, source port, destination port, and L4 protocol.

Example

Additional References

The following sections provide references related to the Flow-Based per Port-Channel Load Balancing feature.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS LAN switching commands	*Cisco IOS LAN Switching Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing standards has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

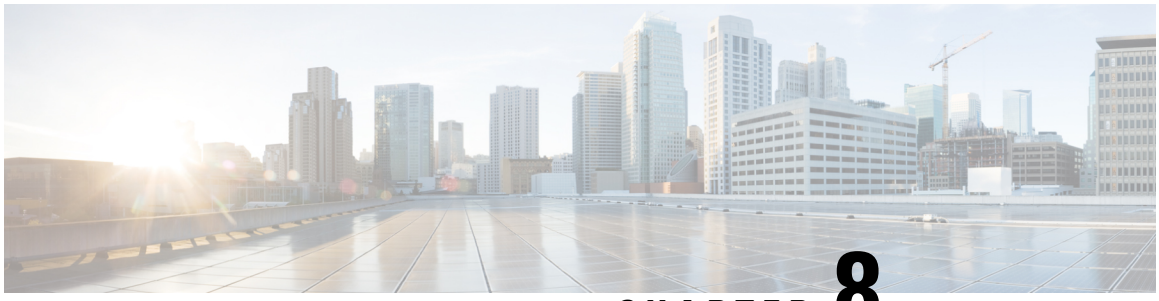
Feature Information for Flow-Based per Port-Channel Load Balancing

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 16: Feature Information for Flow-Based per Port-Channel Load Balancing

Feature Name	Releases	Feature Information
Flow-Based per Port-Channel Load Balancing	Cisco IOS XE Release 2.5	This feature allows different flows of traffic over a GEC interface to be identified and mapped to the different member links. It also enables you to apply load balancing to specific port channels. The following commands were introduced or modified: **load-balancing**, **port-channel load-balancing vlan-manual**, **show etherchannel load-balancing, show interfaces port-channel etherchannel**.
IPv6 Loadbalancing on GEC	Cisco IOS XE Release 3.4S	The IPv6 Loadbalancing on GEC feature provides load balancing for IPv6 traffic on Gigabit EtherChannel.
Five-Tuple Hash Support for GEC Flow-based Load Balancing	Cisco IOS XE Everest 16.4.1	The five-tuple hash support for gigabit etherchannel (GEC) flow-based load balancing feature decides which member link to use for routing traffic based on the hash value obtained from the calculation of 5 parameters: source ip address, destination ip address, source port, destination port, and L4 protocol.



CHAPTER 8

VLANs over IP Unnumbered SubInterfaces

The VLANs over IP Unnumbered Subinterfaces feature allows IP unnumbered interface support to be configured on Ethernet VLAN subinterfaces. This feature also provides support for DHCP on VLAN subinterfaces. Configuring Ethernet VLANs on IP unnumbered subinterfaces can save IPv4 address space and simplify configuration management, address management, and migration for DSL providers from ATM networks to IP.

- Prerequisites for VLANs over IP Unnumbered Subinterfaces, on page 137
- Restrictions for VLANs over IP Unnumbered Subinterfaces, on page 137
- Information About VLANs over IP Unnumbered Subinterfaces, on page 138
- How to Configure VLANs over IP Unnumbered Subinterfaces, on page 140
- Configuration Examples for VLANs over IP Unnumbered Subinterfaces, on page 143
- Additional References for VLANs over IP Unnumbered Subinterfaces, on page 143
- Feature Information for VLANs over IP Unnumbered Subinterfaces, on page 144

Prerequisites for VLANs over IP Unnumbered Subinterfaces

Configure DHCP and ensure that it is operational.

Restrictions for VLANs over IP Unnumbered Subinterfaces

- Only Ethernet VLAN subinterfaces, in addition to serial interfaces, can be configured as IP unnumbered interfaces.
- Interface ranges (the **interface range** command) are not supported in Cisco IOS Release 12.2(18)SXE.
- A physical interface cannot be used for Layer 3 services (no IP address configurations are supported on the physical interface) if one of the subinterface is configured as native.

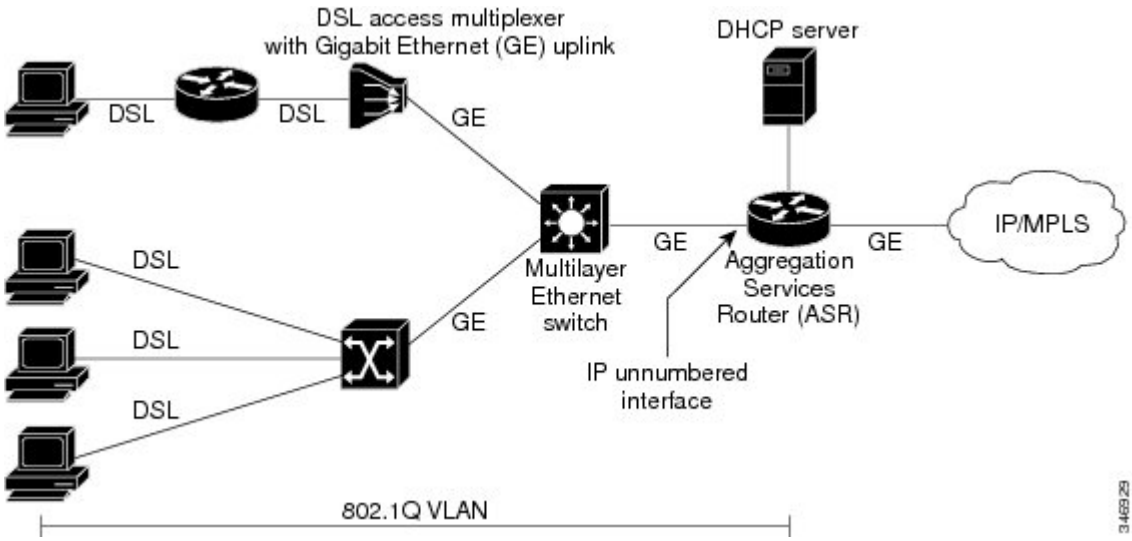
Information About VLANs over IP Unnumbered Subinterfaces

Support for VLANs over IP Unnumbered Subinterfaces

The VLANs over IP Unnumbered Subinterfaces feature enables Ethernet VLANs to be configured on IP unnumbered subinterfaces. The IP unnumbered interface configuration enables IP processing on an interface without assigning an IP address to the interface. The IP unnumbered interface borrows an IP address from another interface that is already configured on the device to conserve network and address space.

Figure 1 shows the implementation of the VLANs over IP Unnumbered Subinterfaces feature in a sample network topology. In this topology, the aggregation services routers dynamically establish IP routes when the DHCP server assigns IP addresses to hosts.

Figure 24: Sample Network Topology Using VLANs over IP Unnumbered Subinterfaces Feature



The VLANs over IP Unnumbered Subinterfaces feature supports the following functions:

- Allocating peer IP address through DHCP.
- Configuring IP unnumbered interface support for a range of VLAN subinterfaces.
- Configuring service selection gateway support for VLANs over IP unnumbered subinterfaces.
- Supporting DHCP relay agent information feature (Option 82).

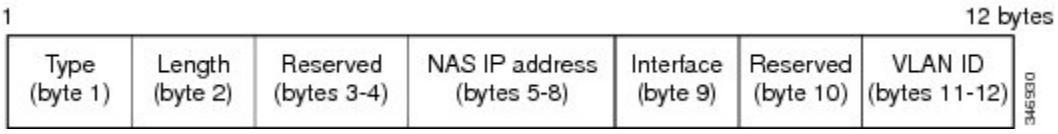
DHCP Option 82

DHCP provides a framework for passing configuration information to hosts on a TCP/IP network. Configuration parameters and other control information are carried in tagged data items (also called options) that are stored in the options field of the DHCP message. Option 82 is organized as a single DHCP option that contains information known by the relay agent.

The DHCP Relay Agent Information feature communicates information to the DHCP server using a suboption of the DHCP relay agent information option called agent remote ID. The information sent in the agent remote ID includes an IP address identifying the relay agent and information about the interface and the connection over which the DHCP request was received. The DHCP server uses this information to assign IP addresses to interfaces and to form security policies.

Figure 2 shows the agent remote ID suboption format that is used with the VLANs over IP Unnumbered Subinterfaces feature.

Figure 25: Format of the Agent Remote ID Suboption



Field	Description
Type	Format type (1 byte). Value 2 specifies the format for use with this feature.
Length	Length of the agent remote ID suboption (1 byte). The type field and the remaining bytes of the length field are not included.
Reserved	Reserved (2 bytes).
NAS IP Address	Network-attached storage (NAS) IP address (4 bytes) of the interface specified by the **ip unnumbered** command.
Interface	Physical interface (1 byte). This field has the following format: slot (4 bits) | module (1 bit) | port (3 bits). For example, if the interface is Ethernet 2/1/1, the slot is 2, the module is 1, and the port is 1.
Reserved	Reserved (1 byte).
VLAN ID	VLAN identifier (2 bytes) for the Ethernet subinterface.

Benefits of VLANs over IP Unnumbered Subinterfaces

The VLANs over IP Unnumbered Subinterfaces feature provides the following benefits:

- Migration from other interfaces to Gigabit Ethernet uplinks and IP core becomes easier for DSL providers.
- All ports share the same subnet, therefore saving the IPv4 address space.
- Each user is on a separate VLAN. DHCP communicates routing information, and there is no Address Resolution Protocol (ARP) or MAC address spoofing, which leads to enhancement in security layers.
- IP address management with DHCP becomes simpler.

- Configuring interface ranges with Ethernet VLAN subinterfaces leads to easier NVRAM configuration and saves overall memory.

How to Configure VLANs over IP Unnumbered Subinterfaces

Configuring IP Unnumbered Interface Support on an Ethernet VLAN Subinterface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **encapsulation dot1q** *vlan-id* [**native**]
5. **ip unnumbered** *type number*
6. **end**
7. **show running-config**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Device(config)# interface fastethernet 1/0.1`	Configures an interface type and enters interface or subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* [**native**] **Example:** `Device(config-subif)# encapsulation dot1q 10`	Enables IEEE 802.1Q encapsulation of traffic on a specified subinterface in a VLAN.

	Command or Action	Purpose
Step 5	**ip unnumbered** *type number* **Example:** `Device(config-subif)# ip unnumbered ethernet 3/0`	Enables IP processing on an interface without assigning an explicit IP address to the interface. • The *type* and *number* arguments specify an interface with a predefined IP address on the device. Do not specify an unnumbered interface, if one already exists.
Step 6	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Displays contents of the current running configuration file on the device including the configuration of the IP unnumbered support feature.

Configuring IP Unnumbered Interface Support on a Range of Ethernet VLAN Subinterfaces

Note The **interface range** command is not supported in Cisco IOS Release 12.2(18)SXE.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface range** {{**ethernet** | **fastethernet** | **gigabitethernet** | **vlan** *vlan*} *slot*/*interface.subinterface* - {**ethernet** | **fastethernet** | **gigabitethernet** | **vlan** *vlan*} *slot*/*interface.subinterface* | **macro** *macro-name*}
4. **encapsulation dot1q** *vlan-id* [**native**]
5. **ip unnumbered** *type number*
6. **end**
7. **show running-config**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface range** {{**ethernet** | **fastethernet** | **gigabitethernet** | **vlan** *vlan*} *slot*/*interface.subinterface* - {**ethernet** | **fastethernet** | **gigabitethernet** | **vlan** *vlan*} *slot*/*interface.subinterface* | **macro** *macro-name*} **Example:** `Device(config)# interface range fastethernet 1/0.1 - fastethernet 1/0.100`	Executes commands on multiple subinterfaces simultaneously. The device prompt changes to configuration interface range mode after the commands are executed. • Separate the interface range with a hyphen and space as shown in the example.
Step 4	**encapsulation dot1q** *vlan-id* [**native**] **Example:** `Device(config-if-range)# encapsulation dot1q 10`	Applies a unique VLAN ID to each subinterface within the range. • The VLAN ID specified by the *vlan-id* argument is applied to the first subinterface in the range. Each subsequent interface is assigned a VLAN ID, which is the specified *vlan-id* including the subinterface number and excluding the first subinterface number (VLAN ID + subinterface number - first subinterface number).
Step 5	**ip unnumbered** *type number* **Example:** `Device(config-if-range)# ip unnumbered ethernet 3/0`	Enables IP processing on an interface without assigning an explicit IP address to the interface. • The *type* and *number* arguments specify an interface with a predefined IP address on the device. Do not specify an unnumbered interface, if one already exists.
Step 6	**end** **Example:** `Device(config-if-range)# end`	Exits interface-range configuration mode and returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Displays contents of the current running configuration file on the device including the configuration of the IP unnumbered support feature.

Configuration Examples for VLANs over IP Unnumbered Subinterfaces

Example: VLAN Configuration on a Single IP Unnumbered Subinterface

The following example shows how to configure IP unnumbered subinterface using Ethernet VLAN subinterface 3/0.2:

```
interface ethernet 3/0.2
 encapsulation dot1q 200
 ip unnumbered ethernet 3/1
```

Example: VLAN Configuration on a Range of IP Unnumbered Subinterfaces

The following example shows how to configure IP unnumbered subinterfaces using Fast Ethernet subinterfaces in the range from 5/1.1 to 5/1.4:

```
interface range fastethernet 5/1.1 - fastethernet 5/1.4
 ip unnumbered ethernet 3/1
```

Additional References for VLANs over IP Unnumbered Subinterfaces

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
IP Addressing commands	Cisco IOS IP Addressing Services Command Reference
IP Addressing Services configuration tasks	Cisco IOS IP Addressing Services Configuration Guide
VLAN configuration tasks	Cisco IOS LAN Switching Configuration Guide
VLAN configuration commands	Cisco IOS LAN Switching Command Reference

RFCs

RFCs	Title
RFC 1812	*Requirements for IP Version 4 Routers*, June 1995

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

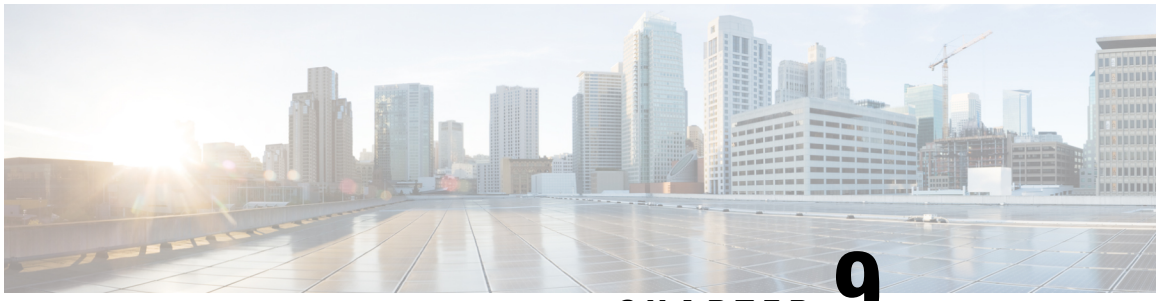
Feature Information for VLANs over IP Unnumbered Subinterfaces

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 17: Feature Information for VLANs over IP Unnumbered Subinterfaces

Feature Name	Releases	Feature Information
VLANs over IP Unnumbered Subinterfaces	Cisco IOS XE Release 3.9S	The VLANs over IP Unnumbered Subinterfaces feature allows IP unnumbered interface support to be configured on Ethernet VLAN subinterfaces. This feature also provides support for DHCP on VLAN subinterfaces. Configuring Ethernet VLANs on IP unnumbered subinterfaces can save IPv4 address space and simplify configuration management, address management, and migration for DSL providers from ATM networks to IP. The following command was modified: **ip unnumbered**



CHAPTER 9

Spanning Tree Protocol

For conceptual information about Spanning Tree Protocol, see the "Using the Spanning Tree Protocol with the EtherSwitch Network Module" section of the EtherSwitch Network feature module.

- Information About Spanning Tree Protocol, on page 145
- How to Configure Spanning Tree Protocol, on page 154
- Configuration Examples for Spanning Tree Protocol, on page 162
- Additional References, on page 165
- Feature Information for Spanning Tree Protocol, on page 165

Information About Spanning Tree Protocol

Using the Spanning Tree Protocol with the EtherSwitch Network Module

The EtherSwitch Network Module uses Spanning Tree Protocol (STP) (the IEEE 802.1D bridge protocol) on all VLANs. By default, a single instance of STP runs on each configured VLAN (provided that you do not manually disable STP). You can enable and disable STP on a per-VLAN basis.

When you create fault-tolerant internetworks, you must have a loop-free path between all nodes in a network. The spanning tree algorithm calculates the best loop-free path throughout a switched Layer 2 network. Switches send and receive spanning tree frames at regular intervals. The switches do not forward these frames but use the frames to construct a loop-free path.

Multiple active paths between end stations cause loops in the network. If a loop exists in the network, end stations might receive duplicate messages and switches might learn endstation MAC addresses on multiple Layer 2 interfaces. These conditions result in an unstable network.

STP defines a tree with a root switch and a loop-free path from the root to all switches in the Layer 2 network. STP forces redundant data paths into a standby (blocked) state. If a network segment in the spanning tree fails and a redundant path exists, the spanning tree algorithm recalculates the spanning tree topology and activates the standby path.

When two ports on a switch are part of a loop, the spanning tree port priority and port path cost setting determine which port is put in the forwarding state and which port is put in the blocking state. The spanning tree port priority value represents the location of an interface in the network topology and how well located it is to pass traffic. The spanning tree port path cost value represents media speed.

Spanning Tree Port States

Propagation delays occur when protocol information passes through a switched LAN. As a result, topology changes take place at different times and at different places in a switched network. When a Layer 2 interface changes from nonparticipation in the spanning tree topology to the forwarding state, it creates temporary data loops. Ports must wait for new topology information to propagate through the switched LAN before starting to forward frames. They must allow the frame lifetime to expire for frames that are forwarded using the old topology.

Each Layer 2 interface on a switch using Spanning Tree Protocol (STP) exists in one of the following states:

- Blocking—The Layer 2 interface does not participate in frame forwarding.
- Disabled—The Layer 2 interface does not participate in spanning tree and is not forwarding frames.
- Forwarding—The Layer 2 interface forwards frames.
- Learning—The Layer 2 interface prepares to participate in frame forwarding.
- Listening—First transitional state after the blocking state when spanning tree determines that the Layer 2 interface must participate in frame forwarding.

A Layer 2 interface moves through the following states:

- From blocking state to listening or disabled state.
- From forwarding state to disabled state.
- From initialization to blocking state.
- From learning state to forwarding or disabled state.
- From listening state to learning or disabled state.

The figure below illustrates how a port moves through these five states.

Boot-up Initialization

When you enable Spanning Tree Protocol (STP), every port in the switch, VLAN, or network goes through the blocking state and transitory states of listening and learning at power up. If properly configured, each Layer 2 interface stabilizes to the forwarding or blocking state.

When the spanning tree algorithm places a Layer 2 interface in the forwarding state, the following process occurs:

1. The Layer 2 interface is put into the listening state while it waits for protocol information to go to the blocking state.
2. The Layer 2 interface waits for the forward delay timer to expire, moves the Layer 2 interface to the learning state, and resets the forward delay timer.
3. The Layer 2 interface continues to block frame forwarding in the learning state as it learns end station location information for the forwarding database.
4. The Layer 2 interface waits for the forward delay timer to expire and then moves the Layer 2 interface to the forwarding state, where both learning and frame forwarding are enabled.

Blocking State

A Layer 2 interface in the blocking state does not participate in frame forwarding, as shown in the figure below. After initialization, a bridge protocol data unit (BPDU) is sent out to each Layer 2 interface in the switch. The switch initially assumes it is the root until it exchanges BPDUs with other switches. This exchange establishes which switch in the network is the root or root bridge. If only one switch is in the network, no exchange occurs, the forward delay timer expires, and the ports move to the listening state. A port enters the blocking state following switch initialization.

A Layer 2 interface in the blocking state performs as follows:

- Discards frames received from the attached segment.
- Discards frames switched from another interface for forwarding.
- Does not incorporate end station location into its address database. (There is no learning on a blocking Layer 2 interface, so there is no address database update.)
- Does not transmit BPDUs received from the system module.
- Receives BPDUs and directs them to the system module.
- Receives and responds to network management messages.

Listening State

The listening state is the first transitional state a Layer 2 interface enters after the blocking state. The Layer 2 interface enters this state when STP determines that the Layer 2 interface must participate in frame forwarding. The figure below shows a Layer 2 interface in the listening state.

A Layer 2 interface in the listening state performs as follows:

- Discards frames received from the attached segment.
- Discards frames switched from another interface for forwarding.
- Does not incorporate end station location into its address database. (There is no learning on a blocking Layer 2 interface, so there is no address database update.)
- Receives and directs BPDUs to the system module.
- Receives, processes, and transmits BPDUs received from the system module.
- Receives and responds to network management messages.

Learning State

The learning state prepares a Layer 2 interface to participate in frame forwarding. The Layer 2 interface enters the learning state from the listening state. The figure below shows a Layer 2 interface in the learning state.

A Layer 2 interface in the learning state performs as follows:

- Discards frames received from the attached segment.
- Discards frames switched from another interface for forwarding.
- Incorporates end station location into its address database.
- Receives BPDUs and directs them to the system module.

- Receives, processes, and transmits BPDUs received from the system module.
- Receives and responds to network management messages.

Forwarding State

A Layer 2 interface in the forwarding state forwards frames, as shown in the figure below. The Layer 2 interface enters the forwarding state from the learning state.

A Layer 2 interface in the forwarding state performs as follows:

- Forwards frames received from the attached segment.
- Forwards frames switched from another Layer 2 interface for forwarding.
- Incorporates end station location information into its address database.
- Receives BPDUs and directs them to the system module.
- Processes BPDUs received from the system module.
- Receives and responds to network management messages.

Disabled State

A Layer 2 interface in the disabled state does not participate in frame forwarding or spanning tree, as shown in the figure below. A Layer 2 interface in the disabled state is virtually nonoperational.

A Layer 2 interface in the disabled state performs as follows:

- Discards frames received from the attached segment.
- Discards frames switched from another Layer 2 interface for forwarding.
- Does not incorporate end station location into its address database. (There is no learning on a blocking Layer 2 interface, so there is no address database update.)
- Does not receive BPDUs for transmission from the system module.

Default Spanning Tree Configuration

The table below shows the default Spanning Tree Protocol (STP) configuration values.

Table 18: SPT Default Configuration Values

Feature	Default Value
Bridge priority	32768
Enable state	Spanning tree enabled for all VLANs
Forward delay time	15 seconds
Hello time	2 seconds
Maximum aging time	20 seconds

Feature	Default Value
Spanning tree port cost (configurable on a per-interface basis; used on interfaces configured as Layer 2 access ports)	Fast Ethernet: 19 Ethernet: 100 Gigabit Ethernet: 19 when operated in 100 Mb mode, and 4 when operated in 1000 Mb mode
Spanning tree port priority (configurable on a per-interface basis; used on interfaces configured as Layer 2 access ports)	128
Spanning tree VLAN port cost (configurable on a per-VLAN basis; used on interfaces configured as Layer 2 trunk ports)	Fast Ethernet: 10 Ethernet: 10
Spanning tree VLAN port priority (configurable on a per-VLAN basis; used on interfaces configured as Layer 2 trunk ports)	128

Bridge Protocol Data Units

The stable active spanning tree topology of a switched network is determined by the following:

- Port identifier (port priority and MAC address) associated with each Layer 2 interface.
- Spanning tree path cost to the root bridge.
- Unique bridge ID (bridge priority and MAC address) associated with each VLAN on each switch.

The bridge protocol data units (BPDUs) are transmitted in one direction from the root switch and each switch sends configuration BPDUs to communicate and compute the spanning tree topology. Each configuration BPDU contains the following minimal information:

- Bridge ID of the transmitting bridge
- Message age
- Port identifier of the transmitting port
- Spanning tree path cost to the root
- Unique bridge ID of the switch that the transmitting switch believes to be the root switch
- Values for the hello, forward delay, and max-age protocol timers

When a switch transmits a BPDU frame, all switches connected to the LAN on which the frame is transmitted receive the BPDU. When a switch receives a BPDU, it does not forward the frame but uses the information in the frame to calculate a BPDU, and, if the topology changes, begin a BPDU transmission.

A BPDU exchange results in the following:

- A designated bridge for each LAN segment is selected. This is the switch closest to the root bridge through which frames are forwarded to the root.
- A root port is selected. This is the port providing the best path from the bridge to the root bridge.
- One switch is elected as the root switch.

- Ports included in the spanning tree are selected.
- The shortest distance to the root switch is calculated for each switch based on the path cost.

For each VLAN, the switch with the highest bridge priority (the lowest numerical priority value) is elected as the root switch. If all switches are configured with the default priority (32768), the switch with the lowest MAC address in the VLAN becomes the root switch.

The spanning tree root switch is the logical center of the spanning tree topology in a switched network. All paths that are not needed to reach the root switch from anywhere in the switched network are placed in spanning tree blocking mode.

BPDUs contain information about the transmitting bridge and its ports, including bridge and MAC addresses, bridge priority, port priority, and path cost. Spanning tree uses this information to elect the root bridge and root port for the switched network, as well as the root port and designated port for each switched segment.

MAC Address Allocation

MAC addresses are allocated sequentially, with the first MAC address in the range assigned to VLAN 1, the second MAC address in the range assigned to VLAN 2, and so forth. For example, if the MAC address range is 00-e0-1e-9b-2e-00 to 00-e0-1e-9b-31-ff, the VLAN 1 bridge ID is 00-e0-1e-9b-2e-00, the VLAN 2 bridge ID is 00-e0-1e-9b-2e-01, the VLAN 3 bridge ID is 00-e0-1e-9b-2e-02, and so forth.

BackboneFast

BackboneFast is started when a root port or blocked port on a switch receives inferior bridge protocol data units (BPDUs) from its designated bridge. An inferior BPDU identifies one switch as both the root bridge and the designated bridge. When a switch receives an inferior BPDU, it means that a link to which the switch is not directly connected is failed. That is, the designated bridge has lost its connection to the root switch. Under Spanning Tree Protocol (STP) rules, the switch ignores inferior BPDUs for the configured maximum aging time specified by the **spanning-tree max-age** command.

The switch determines if it has an alternate path to the root switch. If the inferior BPDU arrives on a blocked port, the root port and other blocked ports on the switch become alternate paths to the root switch. If the inferior BPDU arrives on the root port, all blocked ports become alternate paths to the root switch. If the inferior BPDU arrives on the root port and there are no blocked ports, the switch assumes that it lost connectivity to the root switch, causes the maximum aging time on the root to expire, and becomes the root switch according to normal STP rules.

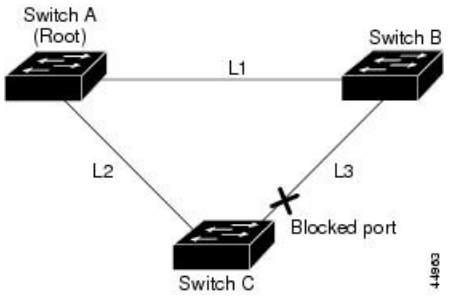


Note Self-looped ports are not considered as alternate paths to the root switch.

If the switch possesses alternate paths to the root switch, it uses these alternate paths to transmit the protocol data unit (PDU) that is called the root link query PDU. The switch sends the root link query PDU on all alternate paths to the root switch. If the switch determines that it has an alternate path to the root, it causes the maximum aging time on ports on which it received the inferior BPDU to expire. If all the alternate paths to the root switch indicate that the switch has lost connectivity to the root switch, the switch causes the maximum aging time on the ports on which it received an inferior BPDU to expire. If one or more alternate paths connect to the root switch, the switch makes all ports on which it received an inferior BPDU its designated ports and moves them out of the blocking state (if they were in the blocking state), through the listening and learning states, and into the forwarding state.

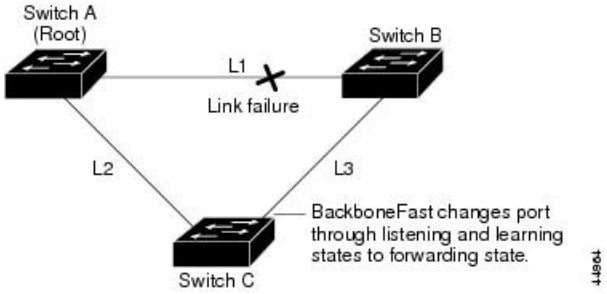
The figure below shows an example topology with no link failures. Switch A, the root switch, connects directly to Switch B over link L1 and to Switch C over link L2. The interface on Switch C that connects directly to Switch B is in the blocking state.

Figure 26: BackboneFast Example Before Indirect Link Failure



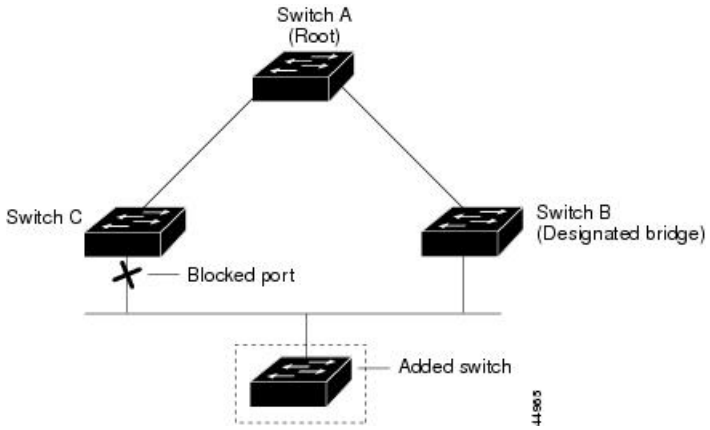
If link L1 fails, Switch C cannot detect this failure because it is not connected directly to link L1. However, Switch B is directly connected to the root switch over L1 and it detects the failure, elects itself as the root switch, and begins sending BPDUs to Switch C. When Switch C receives the inferior BPDUs from Switch B, Switch C assumes that an indirect failure has occurred. At that point, BackboneFast allows the blocked port on Switch C to move to the listening state without waiting for the maximum aging time for the port to expire. BackboneFast then changes the interface on Switch C to the forwarding state, providing a path from Switch B to Switch A. This switchover takes 30 seconds, twice the forward delay time, if the default forward delay time of 15 seconds is set. The figure below shows how BackboneFast reconfigures the topology to account for the failure of link L1.

Figure 27: BackboneFast Example After Indirect Link Failure



If a new switch is introduced into a shared-medium topology as shown in the figure below, BackboneFast is not activated because inferior BPDUs did not come from the designated bridge (Switch B). The new switch begins sending inferior BPDUs that say it is the root switch. However, the other switches ignore these inferior BPDUs, and the new switch learns that Switch B is the designated bridge to Switch A, the root switch.

Figure 28: Adding a Switch in a Shared-Medium Topology



STP Timers

The table below describes the Spanning Tree Protocol (STP) timers that affect the entire spanning tree performance.

Table 19: STP Timers

Timer	Purpose
Forward delay timer	Determines how long listening state and learning state last before the port begins forwarding.
Hello timer	Determines how often the switch broadcasts hello messages to other switches.
Maximum age timer	Determines how long a switch can store the protocol information received on a port.

Spanning Tree Port Priority

Spanning tree considers port priority when selecting an interface to put into the forwarding state if there is a loop. You can assign higher priority values to interfaces that you want spanning tree to select first, and lower priority values to interfaces that you want spanning tree to select last. If all interfaces possess the same priority value, spanning tree puts the interface with the lowest interface number in the forwarding state and blocks other interfaces. The spanning tree port priority range is from 0 to 255, configurable in increments of 4. The default value is 128.

Cisco software uses the port priority value when an interface is configured as an access port and uses VLAN port priority values when an interface is configured as a trunk port.

Spanning Tree Port Cost

The spanning tree port path cost default value is derived from the media speed of an interface. if there is a loop, spanning tree considers port cost value when moving an interface to the forwarding state. You can assign lower port cost values to interfaces that you want spanning tree to select first and higher port cost values to

interfaces that you want spanning tree to select last. If all interfaces have the same port cost value, spanning tree puts the interface with the lowest interface number to the forwarding state and blocks other interfaces.

The port cost range is from 0 to 65535. The default value is media-specific.

Spanning tree uses the port cost value when an interface is configured as an access port and uses VLAN port cost value when an interface is configured as a trunk port.

Spanning tree port cost value calculations are based on the bandwidth of the port. There are two classes of port cost values. Short (16-bit) values are specified by the IEEE 802.1D specification and the range is from 1 to 65535. Long (32-bit) values are specified by the IEEE 802.1t specification and the range is from 1 to 200,000,000.

Assigning Short Port Cost Values

You can manually assign port cost values in the range of 1 to 65535. Default port cost values are listed in Table 2.

Table 20: Default Port Cost Values

Port Speed	Default Port Cost Value
10 Mbps	100
100 Mbps	19

Assigning Long Port Cost Values

You can manually assign port cost values in the range of 1 to 200,000,000. Default port cost values are listed in Table 3.

Table 21: Default Port Cost Values

Port Speed	Recommended Value	Recommended Range
10 Mbps	2,000,000	200,000 to 20,000,000
100 Mbps	200,000	20,000 to 2,000,000

Spanning Tree Root Bridge

The EtherSwitch HWIC maintains a separate instance of spanning tree for each active VLAN configured on the device. A bridge ID, consisting of the bridge priority and the bridge MAC address, is associated with each instance. For each VLAN, the device with the lowest bridge ID will become the root bridge for that VLAN.

To configure a VLAN instance to become the root bridge, the bridge priority can be modified from the default value (32768) to a lower value so that the bridge becomes the root bridge for the specified VLAN. Use the **spanning-tree vlan root** command to alter the bridge priority.

The device checks the bridge priority of current root bridges for each VLAN. The bridge priority for specified VLANs is set to 8192, if this value is caused the device to become the root for specified VLANs.

If any root device for specified VLANs has a bridge priority lower than 8192, the device sets the bridge priority for specified VLANs to 1 less than the lowest bridge priority.

For example, if all devices in a network have the bridge priority for VLAN 100 set to the default value of 32768, entering the **spanning-tree vlan 100 root primary** command on a device sets the bridge priority for VLAN 100 to 8192, causing the device to become the root bridge for VLAN 100.

Note The root device for each instance of spanning tree must be a backbone or distribution device. Do not configure an access device as the spanning tree primary root.

Use the **diameter** keyword to specify the Layer 2 network diameter. That is, the maximum number of bridge hops between any two end stations in the Layer 2 network. When you specify the network diameter, the device automatically picks an optimal hello time, a forward delay time, and a maximum age time for a network of that diameter, which reduces the spanning tree convergence time. You can use the **hello** keyword to override the automatically calculated hello time.

Note We recommend that you do not configure the hello time, forward delay time, and maximum age time manually after you configure the device as the root bridge.

How to Configure Spanning Tree Protocol

Enabling Spanning Tree Protocol

You can enable spanning tree protocol on a per-VLAN basis. The device maintains a separate instance of spanning tree for each VLAN except for which you disable spanning tree.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **spanning-tree vlan** *vlan-id*
4. **end**
5. **show spanning-tree vlan** *vlan-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**spanning-tree vlan** *vlan-id* **Example:** `Device(config)# spanning-tree vlan 200`	Enables spanning tree on a per-VLAN basis.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.
Step 5	**show spanning-tree vlan** *vlan-id* **Example:** `Device# show spanning-tree vlan 200`	Verifies spanning tree configuration.

Configuring the Bridge Priority of a VLAN

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **spanning-tree vlan** *vlan-id* **priority** *bridge-priority*
4. **show spanning-tree vlan bridge**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**spanning-tree vlan** *vlan-id* **priority** *bridge-priority* **Example:** `Device(config)# spanning-tree vlan 200 priority 2`	Configures the bridge priority of a VLAN. The bridge priority value ranges from 0 to 65535. **Caution** Use the **spanning-tree vlan** *vlan-id* **root primary** command and the **spanning-tree vlan** *vlan-id* **root secondary** command to modify the bridge priority.

	Command or Action	Purpose
Step 4	**show spanning-tree vlan bridge** **Example:** `Device(config-if)# spanning-tree cost 200`	Verifies the bridge priority.

Configuring STP Timers

Configuring Hello Time

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **spanning-tree vlan** *vlan-id* **hello-time** *hello-time*
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**spanning-tree vlan** *vlan-id* **hello-time** *hello-time* **Example:** `Device(config)# spanning-tree vlan 200 hello-time 5`	Configures the hello time for a VLAN.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.

Configuring the Forward Delay Time for a VLAN

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **spanning-tree vlan** *vlan-id* **forward-time** *forward-time*
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**spanning-tree vlan** *vlan-id* **forward-time** *forward-time* **Example:** `Device(config)# spanning-tree vlan 20 forward-time 5`	Configures the forward delay time for a VLAN.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.

Configuring the Maximum Aging Time for a VLAN

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **spanning-tree vlan** *vlan-id* **max-age** *max-age*
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**spanning-tree vlan** *vlan-id* **max-age** *max-age* **Example:** `Device(config)# spanning-tree vlan 200 max-age 30`	Configures the maximum aging time for a VLAN.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.

Configuring Spanning Tree Port Priority

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **spanning-tree port-priority** *port-priority*
5. **spanning-tree vlan** *vlan-id* **port-priority** *port-priority*
6. **end**
7. **show spanning-tree interface fastethernet** *interface-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface fastethernet 0/1/6`	Configures an interface and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**spanning-tree port-priority** *port-priority* **Example:** `Device(config-if)# spanning-tree port-priority 8`	Configures the port priority for an interface.
Step 5	**spanning-tree vlan** *vlan-id* **port-priority** *port-priority* **Example:** `Device (config-if)# spanning-tree vlan vlan1 port-priority 12`	Configures the port priority for a VLAN.
Step 6	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.
Step 7	**show spanning-tree interface fastethernet** *interface-id* **Example:** `Device# show spanning-tree interface fastethernet 0/1/6`	(Optional) Saves your entries in the configuration file.

Configuring Spanning Tree Port Cost

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **spanning-tree cost** *port-cost*
5. **spanning-tree vlan** *vlan-id* **cost** *port-cost*
6. **end**
7. **show spanning-tree interface fastethernet** *interface-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Device(config)# interface fastethernet 0/1/6`	Configures an interface and enters interface configuration mode.
Step 4	**spanning-tree cost** *port-cost* **Example:** `Device(config-if)# spanning-tree cost 2000`	Configures the port cost for an interface.
Step 5	**spanning-tree vlan** *vlan-id* **cost** *port-cost* **Example:** `Device(config-if)# spanning-tree vlan 200 cost 2000`	Configures the VLAN port cost for an interface.
Step 6	**end** **Example:** `Device(config)# end`	Exits interface configuration mode and enters privileged EXEC mode.
Step 7	**show spanning-tree interface fastethernet** *interface-id* **Example:** `Device# show spanning-tree interface fastethernet 0/1/6`	(Optional) Saves your entries in the configuration file.

Configuring Spanning Tree Root Bridge

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **spanning-tree vlan** *vlanid* **root primary** [**diameter** *hops* [**hello-time** *seconds*]]
4. **no spanning-tree vlan** *vlan-id*
5. **show spanning-tree vlan** *vlan-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**spanning-tree vlan** *vlanid* **root primary** [**diameter** *hops* [**hello-time** *seconds*]] **Example:** `Device(config)# spanning-tree vlan 200 root primary`	Configures a device as the root device.
Step 4	**no spanning-tree vlan** *vlan-id* **Example:** `Device(config)# no spanning-tree vlan 200 root primary`	Disables spanning tree on a per-VLAN basis.
Step 5	**show spanning-tree vlan** *vlan-id* **Example:** `Device(config)# show spanning-tree vlan 200`	Verifies spanning tree on a per-VLAN basis.

Verifying Spanning Tree on a VLAN

SUMMARY STEPS

1. **enable**
2. **show spanning-tree** [*bridge-group*] [**active** | **backbonefast** | **blockedports** | **bridge** | **brief** | **inconsistentports** | **interface** *interface-type interface-number* | **pathcost method** | **root** | **summary** [**totals**] | **uplinkfast** | **vlan** *vlan-id*]

DETAILED STEPS

Procedure

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Device> enable
```

Step 2 **show spanning-tree** [*bridge-group*] [**active** | **backbonefast** | **blockedports** | **bridge** | **brief** | **inconsistentports** | **interface** *interface-type interface-number* | **pathcost method** | **root** | **summary** [**totals**] | **uplinkfast** | **vlan** *vlan-id*]

Use this command with the **vlan** keyword to display the spanning tree information about a specified VLAN.

Example:

```
Device# show spanning-tree vlan 200
VLAN200 is executing the ieee compatible Spanning Tree protocol
  Bridge Identifier has priority 32768, address 0050.3e8d.6401
  Configured hello time 2, max age 20, forward delay 15
  Current root has priority 16384, address 0060.704c.7000
  Root port is 264 (FastEthernet5/8), cost of root path is 38
  Topology change flag not set, detected flag not set
```

```
  Number of topology changes 0 last change occurred 01:53:48 ago

 Times:  hold 1, topology change 24, notification 2
          hello 2, max age 14, forward delay 10
  Timers: hello 0, topology change 0, notification 0
```

Example:

```
Port 264 (FastEthernet5/8) of VLAN200 is forwarding
   Port path cost 19, Port priority 128, Port Identifier 129.9.
   Designated root has priority 16384, address 0060.704c.7000
   Designated bridge has priority 32768, address 00e0.4fac.b000
   Designated port id is 128.2, designated path cost 19
   Timers: message age 3, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   BPDU: sent 3, received 3417
```

Use this command with the **interface** keyword to display spanning tree information about a specified interface.

Example:

```
Device# show spanning-tree interface fastethernet 5/8
Port 264 (FastEthernet5/8) of VLAN200 is forwarding
   Port path cost 19, Port priority 100, Port Identifier 129.8.
   Designated root has priority 32768, address 0010.0d40.34c7
   Designated bridge has priority 32768, address 0010.0d40.34c7
   Designated port id is 128.1, designated path cost 0
   Timers: message age 2, forward delay 0, hold 0
   Number of transitions to forwarding state: 1
   BPDU: sent 0, received 13513
```

Use this command with the **bridge**, **brief**, and **vlan** keywords to display the bridge priority information.

Example:

```
Device# show spanning-tree bridge brief vlan 200
 Hello Max  Fwd
Vlan                 Bridge ID      Time  Age Delay  Protocol
---------------- -------------------- ---- ---- -----  --------
VLAN200          33792 0050.3e8d.64c8    2   20    15  ieee
```

Configuration Examples for Spanning Tree Protocol

Example: Enabling Spanning Tree Protocol

The following example shows how to enable spanning tree protocol on VLAN 20:

```
Device# configure terminal
Device(config)# spanning-tree vlan 20
Device(config)# end
Device#
```

Note Because spanning tree is enabled by default, the **show running** command will not display the command you entered to enable spanning tree protocol.

The following example shows how to disable spanning tree protocol on VLAN 20:

```
Device# configure terminal
Device(config)# no spanning-tree vlan 20
Device(config)# end
Device#
```

Example: Configuring the Bridge Priority of a VLAN

The following example shows how to configure the bridge priority of VLAN 20 to 33792:

```
Device# configure terminal
Device(config)# spanning-tree vlan 20 priority 33792
Device(config)# end
```

Example: Configuring STP Timers

Example: Configuring Hello Time

The following example shows how to configure the hello time for VLAN 20 to 7 seconds:

```
Device# configure terminal
Device(config)# spanning-tree vlan 20 hello-time 7
Device(config)# end
```

Example: Configuring the Forward Delay Time for a VLAN

The following example shows how to configure the forward delay time for VLAN 20 to 21 seconds:

```
Device#configure terminal
Device(config)#spanning-tree vlan 20 forward-time 21
Device(config)#end
```

Example: Configuring the Maximum Aging Time for a VLAN

The following example shows how to configure the maximum aging time for VLAN 20 to 36 seconds:

```
Device#configure terminal
Device(config)#spanning-tree vlan 20 max-age 36
Device(config)#end
```

Example: Configuring Spanning Tree Port Priority

The following example shows how to configure VLAN port priority on an interface:

```
Device# configure terminal
Device(config)# interface fastethernet 0/3/2
Device(config-if)# spanning-tree vlan 20 port priority 64
Device(config-if)# end
```

The following example shows how to verify the configuration of VLAN 20 on an interface when it is configured as a trunk port:

```
Device#show spanning-tree vlan 20

 VLAN20 is executing the ieee compatible Spanning Tree protocol
  Bridge Identifier has priority 32768, address 00ff.ff90.3f54
  Configured hello time 2, max age 20, forward delay 15
  Current root has priority 32768, address 00ff.ff10.37b7
  Root port is 33 (FastEthernet0/3/2), cost of root path is 19
  Topology change flag not set, detected flag not set
  Number of topology flags 0 last change occurred 00:05:50 ago
  Times: hold 1, topology change 35, notification 2
     hello 2, max age 20, forward delay 15
  Timers: hello 0, topology change 0, notification 0, aging 0
 Port 33 (FastEthernet0/3/2) of VLAN20 is forwarding
  Port path cost 18, Port priority 64, Port Identifier 64.33
  Designated root has priority 32768, address 00ff.ff10.37b7
  Designated bridge has priority 32768, address 00ff.ff10.37b7
  Designated port id is 128.13, designated path cost 0
  Timers: message age 2, forward delay 0, hold 0
  Number of transitions to forwarding state: 1
  BPDU: sent 1, received 175
```

Example: Configuring Spanning Tree Port Cost

The following example shows how to change the spanning tree port cost of a Fast Ethernet interface:

```
Device# configure terminal
Device(config)# interface fastethernet0/3/2
Device(config-if)# spanning-tree cost 18
Device(config-if)# end
Device#
Device# show run interface fastethernet0/3/2
Building configuration...
Current configuration: 140 bytes
!
interface FastEthernet0/3/2
 switchport access vlan 20
  no ip address
  spanning-tree vlan 20 port-priority 64
  spanning-tree cost 18
end
```

The following example shows how to verify the configuration of a Fast Ethernet interface when it is configured as an access port:

```
Device# show spanning-tree interface fastethernet0/3/2

 Port 33 (FastEthernet0/3/2) of VLAN20 is forwarding
  Port path cost 18, Port priority 64, Port Identifier 64.33
  Designated root has priority 32768, address 00ff.ff10.37b7
  Designated bridge has priority 32768, address 00ff.ff10.37b7
  Designated port id is 128.13, designated path cost 0
  Timers: message age 2, forward delay 0, hold 0
  Number of transitions to forwarding state: 1
  BPDU: sent 1, received 175
```

Example: Configuring Spanning Tree Root Bridge

The following example shows how to configure the spanning tree root bridge for VLAN 10, with a network diameter of 4:

```
Device# configure terminal
Device(config)# spanning-tree vlan 10 root primary diameter 4
Device(config)# exit
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
LAN switching commands	

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Spanning Tree Protocol

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 22: Feature Information for Spanning Tree Protocol

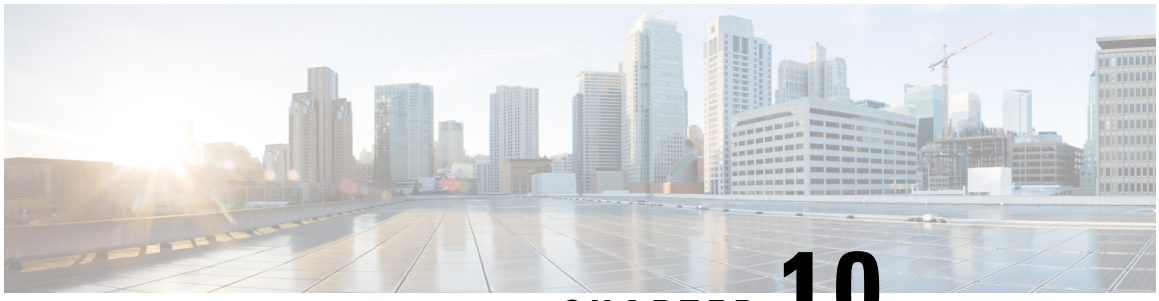
Feature Name	Releases	Feature Information
Spanning Tree Protocol	12.1(1)E	Spanning Tree Protocol (STP) is a Layer 2 link management protocol that provides path redundancy while preventing undesirable loops in the network. The following commands were introduced or modified: **spanning-tree vlan**, **spanning-tree port-priority**, and **spanning-tree cost**.



PART II

Wide Area Networking

- Wide-Area Networking Overview, on page 169
- Layer 2 Tunneling Protocol Version 3, on page 175
- L2VPN Pseudowire Redundancy, on page 231
- Layer 2 Local Switching, on page 255



CHAPTER 10

Wide-Area Networking Overview

Cisco IOS software provides a range of wide-area networking capabilities to fit almost every network environment need. Cisco offers cell relay via the Switched Multimegabit Data Service (SMDS), circuit switching via ISDN, packet switching via Frame Relay, and the benefits of both circuit and packet switching via Asynchronous Transfer Mode (ATM). LAN emulation (LANE) provides connectivity between ATM and other LAN types. The *Cisco IOS Wide-Area Networking Configuration Guide* presents a set of general guidelines for configuring the following software components:

This module gives a high-level description of each technology. For specific configuration information, see the appropriate module.

- Frame Relay, on page 169
- Layer 2 Virtual Private Network, on page 172

Frame Relay

The Cisco Frame Relay implementation currently supports routing on IP, DECnet, AppleTalk, XNS, Novell IPX, CLNS, Banyan VINES, and transparent bridging.

Although Frame Relay access was originally restricted to leased lines, dialup access is now supported. For more information, for dialer profiles or for legacy dial-on-demand routing (DDR) see the see the module Dial-on-Demand Routing Configuration.

To install software on a new router or access server by downloading software from a central server over an interface that supports Frame Relay, see the module Loading and Maintaining System Images.

To configure access between Systems Network Architecture (SNA) devices over a Frame Relay network, see the module Configuring SNA Frame Relay Access Support.

The Frame Relay software provides the following capabilities:

- Support for the three generally implemented specifications of Frame Relay Local Management Interfaces (LMIs):
  - The Frame Relay Interface joint specification produced by Northern Telecom, Digital Equipment Corporation, StrataCom, and Cisco Systems
  - The ANSI-adopted Frame Relay signal specification, T1.617 Annex D
  - The ITU-T-adopted Frame Relay signal specification, Q.933 Annex A
- Conformity to ITU-T I-series (ISDN) recommendation as I122, "Framework for Additional Packet Mode Bearer Services":

  - The ANSI-adopted Frame Relay encapsulation specification, T1.618
  - The ITU-T-adopted Frame Relay encapsulation specification, Q.922 Annex A
- Conformity to Internet Engineering Task Force (IETF) encapsulation in accordance with RFC 2427, except bridging.
- Support for a keepalive mechanism, a multicast group, and a status message, as follows:
  - The keepalive mechanism provides an exchange of information between the network server and the switch to verify that data is flowing.
  - The multicast mechanism provides the network server with a local data-link connection identifier (DLCI) and a multicast DLCI. This feature is specific to our implementation of the Frame Relay joint specification.
  - The status mechanism provides an ongoing status report on the DLCIs known by the switch.
- Support for both PVCs and SVCs in the same sites and routers.

SVCs allow access through a Frame Relay network by setting up a path to the destination endpoints only when the need arises and tearing down the path when it is no longer needed.

- Support for Frame Relay Traffic Shaping beginning with Cisco IOS Release 11.2. Traffic shaping provides the following:
  - Rate enforcement for individual circuits--The peak rate for outbound traffic can be set to the committed information rate (CIR) or some other user-configurable rate.
  - Dynamic traffic throttling on a per-virtual-circuit basis--When backward explicit congestion notification (BECN) packets indicate congestion on the network, the outbound traffic rate is automatically stepped down; when congestion eases, the outbound traffic rate is stepped up again.
  - Enhanced queueing support on a per-virtual circuit basis--Custom queueing, priority queueing, and weighted fair queueing can be configured for individual virtual circuits.
- Transmission of congestion information from Frame Relay to DECnet Phase IV and CLNS. This mechanism promotes forward explicit congestion notification (FECN) bits from the Frame Relay layer to upper-layer protocols after checking for the FECN bit on the incoming DLCI. Use this Frame Relay congestion information to adjust the sending rates of end hosts. FECN-bit promotion is enabled by default on any interface using Frame Relay encapsulation. No configuration is required.
- Support for Frame Relay Inverse ARP as described in RFC 1293 for the AppleTalk, Banyan VINES, DECnet, IP, and IPX protocols, and for native hello packets for DECnet, CLNP, and Banyan VINES. It allows a router running Frame Relay to discover the protocol address of a device associated with the virtual circuit.
- Support for Frame Relay switching, whereby packets are switched based on the DLCI--a Frame Relay equivalent of a Media Access Control (MAC)-level address. Routers are configured as a hybrid DTE switch or pure Frame Relay DCE access node in the Frame Relay network.

Frame Relay switching is used when all traffic arriving on one DLCI can be sent out on another DLCI to the same next-hop address. In such cases, the Cisco IOS software need not examine the frames individually to discover the destination address, and, as a result, the processing load on the router decreases.

The Cisco implementation of Frame Relay switching provides the following functionality:

-
  - Switching over an IP tunnel
  - Switching over Network-to-Network Interfaces (NNI) to other Frame Relay switches
  - Local serial-to-serial switching

- Switching over ISDN B channels
- Traffic shaping on switched PVCs
- Congestion management on switched PVCs
- Traffic policing on User-Network Interface (UNI) DCE
- FRF.12 fragmentation on switched PVCs

- Support for *subinterfaces* associated with a physical interface. The software groups one or more PVCs under separate subinterfaces, which in turn are located under a single physical interface. See the Configuring Frame Relay module.
- Support for fast-path transparent bridging, as described in RFC 1490, for Frame Relay encapsulated serial and High-Speed Serial Interfaces (HSSIs) on all platforms.
- Support of the Frame Relay DTE MIB specified in RFC 1315. However, the error table is not implemented. To use the Frame Relay MIB, refer to your MIB publications.
- Support for Frame Relay fragmentation. Cisco has developed the following three types of Frame Relay fragmentation:
  - End-to-End FRF.12 Fragmentation

FRF.12 fragmentation is defined by the FRF.12 Implementation Agreement. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. End-to-end FRF.12 fragmentation is recommended for use on PVCs that share links with other PVCs that are transporting voice and on PVCs transporting Voice over IP (VoIP).

- • Frame Relay Fragmentation Using FRF.11 Annex C

When VoFR (FRF.11) and fragmentation are both configured on a PVC, the Frame Relay fragments are sent in the FRF.11 Annex C format. This fragmentation is used when FRF.11 voice traffic is sent on the PVC, and it uses the FRF.11 Annex C format for data.

See the module Configuring Voice over Frame Relay in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Frame Relay fragmentation using FRF.11 Annex C.

- • Cisco Proprietary Fragmentation

Cisco proprietary fragmentation is used on data packets on a PVC that is also used for voice traffic.

See the module Configuring Voice over Frame Relay in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Cisco proprietary fragmentation.

Frame Relay-ATM Internetworking

Cisco IOS software supports the Frame Relay Forum implementation agreements for Frame Relay-ATM Interworking. Frame Relay-ATM Interworking enables Frame Relay and ATM networks to exchange data, despite differing network protocols. There are two types of Frame Relay-ATM Interworking.

FRF.5 Frame Relay-ATM Network Interworking

FRF.5 provides network interworking functionality that allows Frame Relay end users to communicate over an intermediate ATM network that supports FRF.5. Multiprotocol encapsulation and other higher-layer procedures are transported transparently, just as they would be over leased lines.

FRF.5 describes network interworking requirements between Frame Relay Bearer Services and Broadband ISDN (BISDN) permanent virtual circuit (PVC) services.

The FRF.5 standard is defined by the Frame Relay Forum Document Number FRF.5: *Frame Relay/ATM PVC Network Interworking Implementation Agreement.* For information about which sections of this implementation agreement are supported by Cisco IOS software, see Frame Relay-ATM Interworking Supported Standards.

FRF.8 Frame Relay-ATM Service Interworking

FRF.8 provides service interworking functionality that allows a Frame Relay end user to communicate with an ATM end user. Traffic is translated by a protocol converter that provides communication among dissimilar Frame Relay and ATM equipment.

FRF.8 describes a one-to-one mapping between a Frame Relay PVC and an ATM PVC.

The FRF.8 standard is defined by the Frame Relay Forum Document Number FRF.8: *Frame Relay/ATM PVC Network Service Interworking Implementation Agreement.* For information about which sections of this implementation agreement are supported by Cisco IOS software, see Frame Relay-ATM Interworking Supported Standards.

Layer 2 Virtual Private Network

L2VPN services are point-to-point. They provide Layer 2 point-to-point connectivity over either an MPLS or a pure IP (L2TPv3) core.

Layer 2 Tunneling Protocol Version 3

The Layer 2 Tunneling Protocol Version 3 feature expands Cisco's support of Layer 2 VPNs. Layer 2 Tunneling Protocol Version 3 (L2TPv3) is an IETF l2tpext working group draft that provides several enhancements to L2TP to tunnel any Layer 2 payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network by using Layer 2 VPNs.

L2VPN Pseudowire Redundancy

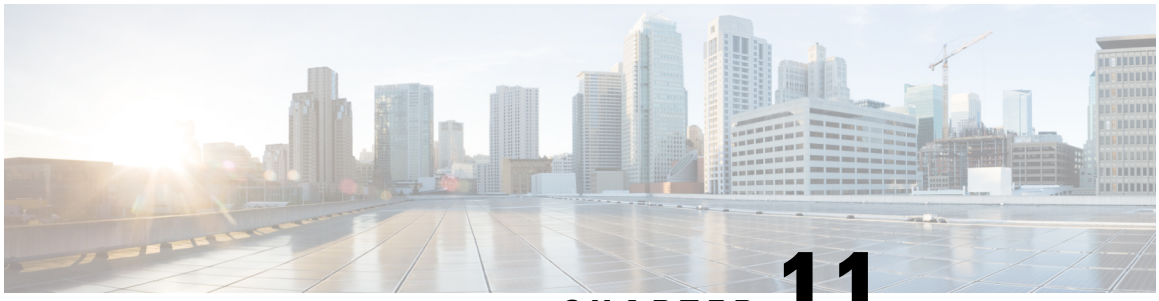
L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data can take over. However, there are some parts of the network where this rerouting mechanism does not protect against interruptions in service. The L2VPN Pseudowire Redundancy feature provides the ability to ensure that the CE2 router in can always maintain network connectivity, even if one or all the failures in the figure occur. The L2VPN Pseudowire Redundancy feature enables you to set up backup pseudowires. You can configure the network with redundant pseudowires (PWs) and redundant network elements.

Layer 2 Virtual Private Network Interworking

Layer 2 transport over MPLS and IP already exists for like-to-like attachment circuits, such as Ethernet-to-Ethernet or PPP-to-PPP. L2VPN Interworking builds on this functionality by allowing disparate attachment circuits to be connected. An interworking function facilitates the translation between the different Layer 2 encapsulations. The L2VPN Interworking feature supports Ethernet, 802.1Q (VLAN), Frame Relay, ATM AAL5, and PPP attachment circuits over MPLS and L2TPv3.

Layer 2 Local Switching

Local switching allows 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 can be on the same line card or on two different cards. During these kinds of switching, the Layer 2 address is used, not any Layer 3 address. Same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.



CHAPTER 11

Layer 2 Tunneling Protocol Version 3

The Layer 2 Tunneling Protocol Version 3 feature expands Cisco's support of Layer 2 VPNs. Layer 2 Tunneling Protocol Version 3 (L2TPv3) is an IETF l2tpext working group draft that provides several enhancements to L2TP to tunnel any Layer 2 payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network by using Layer 2 VPNs.

- Feature Information for Layer 2 Tunneling Protocol Version 3, on page 175
- Prerequisites for Layer 2 Tunneling Protocol Version 3, on page 176
- Restrictions for Layer 2 Tunneling Protocol Version 3, on page 177
- Information About Layer 2 Tunneling Protocol Version 3, on page 179
- How to Configure Layer 2 Tunneling Protocol Version 3, on page 193
- Configuration Examples for Layer 2 Tunneling Protocol Version 3, on page 219
- Additional References, on page 226
- Glossary, on page 227

Feature Information for Layer 2 Tunneling Protocol Version 3

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 23: Feature Information for Layer 2 Tunneling Protocol Version 3

Feature Name	Releases	Feature Information
Layer 2 TPv3 on Switched Virtual Interface	Cisco IOS XE Amsterdam 17.2.1r	The Layer 2 Tunneling Protocol Version 3 (L2TPv3) feature expands Cisco's support of Layer 2 VPNs on Switched Virtual Interface. For L2TPv3 enabled on SVI interfaces that have xconnect deployed on the devices, the ports are configured to function as LAN switching ports. There are no commands introduced or modified for this feature.

Feature Name	Releases	Feature Information
Layer 2Tunneling Protocol Version 3	XE 2.6 XE 2.6.2 XE 3.3S XE 3.11S	The Layer 2 Tunneling Protocol Version 3 (L2TPv3) feature expands Cisco's support of Layer 2 VPNs. In Cisco IOS XE Release 2.6, the following features were added: • Ethernet over L2TPv3 • Layer 2 VPN (L2VPN): Syslog, SNMP Trap, and show Command Enhancements for AToM and L2TPv3 • L2TPv3 Control Message Hashing • L2TPv3 Control Message Rate Limiting • L2TPv3 Digest Secret Graceful Switchover • L2TPv3 Protocol Demultiplexing • L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations In Cisco IOS XE Release 2.6.2, support was added for the **ip pmtu** command. In Cisco IOS XE Release 3.3S, support for HDLC over L2TPv3 was added. The following commands were introduced or modified: **clear l2tun**, **debug vpdn**, **ip pmtu**, **l2tp cookie local**, **l2tp cookie remote l2tp hello**, **l2tp id**, and **xconnect**.

Prerequisites for Layer 2 Tunneling Protocol Version 3

- Before you configure an xconnect attachment circuit for a provider edge (PE) device , the Cisco Express Forwarding (formerly known as CEF) feature must be enabled. To enable Cisco Express Forwarding on an interface, use the **ip cef** or **ip cef distributed** command.
- 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 PE device at the other end of an L2TPv3 control channel.
- 800 L2TPv3 sessions are supported on the Cisco 1000 Series Integrated Services in the below format:
- 800 L2TPv3 sessions ---- 800 loopbacks ---- 800 vlans ---- 800 SVIs



Note Recommended L2TPv3 sessions - 200

Restrictions for Layer 2 Tunneling Protocol Version 3

General L2TPv3 Restrictions

- Cisco Express Forwarding must be enabled for the L2TPv3 feature to function. The xconnect configuration mode is blocked until Cisco Express Forwarding is enabled. On distributed platforms, such as the Cisco 7500 series, if Cisco Express Forwarding is disabled while a session is established, the session is torn down. The session remains down until Cisco Express Forwarding is reenabled. To enable Cisco Express Forwarding, use the **ip cef** or **ip cef distributed** command.
- The number of sessions on PPP, High-Level Data Link Control (HDLC), Ethernet, or 802.1q VLAN ports is limited by the number of interface descriptor blocks (IDBs) that the router can support. For PPP, HDLC, Ethernet, and 802.1q VLAN circuit types, an IDB is required for each circuit.
- When L2TPv3 is used to tunnel Frame Relay D channel data-link connection identifiers (DLCIs), an IDB is not required for each circuit. As a result, the memory requirements are much lower. The scalability targets for the Engineering Field Test (EFT) program are 4000 L2TP session.
- To convert an interface with Any Transport over MPLS (AToM) xconnect to L2TPv3 xconnect, remove the AToM configuration from the interface and then configure L2TPv3. Some features may not work if L2TPv3 is configured before removing the AToM configuration.
- Frame Relay support includes only 10-bit DLCI addressing. The L2TPv3 feature does not support Frame Relay extended addressing.
- The interface keepalive feature is automatically disabled on the interface to which xconnect is applied, except for Frame Relay encapsulation, which is required for Local Management Interface (LMI).
- Static L2TPv3 sessions do not support Frame Relay LMI interworking.
- Static L2TPv3 sessions do not interoperate with Universal Tunnel Interface (UTI) using keepalives.
- Layer 2 fragmentation of IP packets and Intermediate System-to-Intermediate System (IS-IS) fragmentation through a static L2TPv3 session are not supported.
- Layer 3 fragmentation is not recommended because of performance degradation.
- The L2TPv3 Layer 2 (IP packet) fragmentation feature (see the Configuring the L2TPv3 Pseudowire task) is not supported when the customer edge (CE) router is running special Layer 2 options such as Layer 2 sequencing, compression, or encryption. Examples of these options are Frame Relay compression and fragmentation or PPP compression. In these scenarios, the IP payload is not in a format that is compatible with IP fragmentation.
- The Stateful Switchover (SSO), Route Processor Redundancy (RPR) and RPR+ components of the HA functions are supported only at the coexistence level. If you attempt a switchover using SSO, RPR, or RPR+, the tunnels will fail and then eventually recover after an undetermined time duration. This includes both IPv4 and IPv6 traffic.
- Interworking is not allowed when sequencing is enabled.
- Untagged packets (native VLAN) forwarding for xconnect that is configured on the dot1q subinterface is not supported.

- L2TPv3 xconnect is not supported on an EtherSwitch module. This limitation is also applicable to switch virtual interfaces (SVI) that are physically terminated on an EtherSwitch module interface.
- Only Ethernet, HDLC, Frame Relay and VLAN (802.1Q, QinQ, and QinAny) attachment circuits are supported; EVC is not supported.
- The IP local interface must be a loopback interface and the loopback interface cannot be in a VRF. Configuring any other interface with the "ip local interface" command results in a nonoperational setting.

VLAN-Specific Restrictions

- A PE device is responsible only for static VLAN membership entries that are configured manually on the device. Dynamic VLAN membership entries, entry aging, and membership discovery are not supported.
- Implicit tagging for VLAN memberships operating on other layers, such as membership by MAC address, protocol type at Layer 2, or membership by IP subnet at Layer 3, is not supported.
- Point-to-multipoint and multipoint-to-point configurations are not supported. There is a 1:1 relationship between an attachment circuit and an L2TPv3 session.

IPv6 Protocol Demultiplexing for L2TPv3 Restrictions

- IPv6 protocol demultiplexing is supported only for Ethernet traffic.
- IPv6 protocol demultiplexing is supported over noninterworking sessions.

L2TPv3 Control Message Hashing Restrictions

- L2TPv3 control channel authentication configured using the **digest** command requires bidirectional configuration on the peer devices. A shared secret must be configured on the communicating nodes.
- For a compatibility matrix of all the L2TPv3 authentication methods, see the Valid Configuration Scenarios table in the IPv6 Protocol Demultiplexing section.

L2TPv3 Digest Secret Graceful Switchover Restrictions

- This feature works only with authentication passwords configured using the L2TPv3 Control Message Hashing feature. L2TPv3 control channel authentication passwords configured with the older, Challenge Handshake Authentication Protocol (CHAP)-like authentication system cannot be updated without tearing down L2TPv3 tunnels and sessions.
- In Cisco IOS Release 12.0(30)S, a maximum of two passwords can be configured simultaneously using the **digest secret** command.

For more information about the L2TPv3 Control Message Hashing feature, see the L2TPv3 Control Message Hashing section.

Quality of Service Restrictions in L2TPv3 Tunneling

Quality of service (QoS) policies configured with the modular QoS command-line interface (MQC) are supported in L2TPv3 tunnel sessions with the following restrictions: Protocol demultiplexing requires a combination of an IP address and the xconnect command configured on the interface. The interface is then treated as a regular L3. To apply QoS on the Layer 2 IPv6 traffic, you must classify the IPv6 traffic into a separate class before applying any feature(s) to it. The following match criteria are used to classify Layer 2 IPv6 traffic on a protocol demultiplexing interface:

The following match criterion is used to classify Layer 2 IPv6 traffic on a protocol demultiplexing interface:

```
class-map match-ipv6
     match protocol ipv6
```

In the absence of a class to handle Layer 2 IPv6 traffic, the service policy is not accepted on a protocol demultiplexing interface.

For detailed information about QoS configuration tasks and command syntax, refer to:

- *Cisco IOS Quality of Service Solutions Configuration Guide*
- *Cisco IOS Quality of Service Solutions Command Reference*

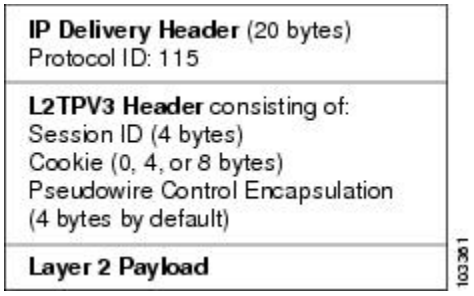
Information About Layer 2 Tunneling Protocol Version 3

L2TPv3 provides a method for delivering L2TP services over an IPv4 (non-UDP) backbone network. It encompasses the signaling protocol as well as the packet encapsulation specification.

L2TPv3 Header Description

The migration from UTI to L2TPv3 also requires the standardization of the UTI header. As a result, the L2TPv3 header has the new format shown in the figure below.

Figure 29: L2TPv3 Header Format



Each L2TPv3 packet contains an L2TPv3 header that includes a unique session ID representing one session and a variable cookie length. The L2TPv3 session ID and the Tunnel Cookie field length are assigned through the CLI. See the Configuration Examples for Layer 2 Tunneling Protocol Version 3 section for more information on the CLI commands for L2TPv3.

Session ID

The L2TPv3 session ID identifies the session context on the decapsulating system. For dynamic sessions, the value of the session ID is selected to optimize the context identification efficiency of the decapsulating system. A decapsulation implementation may, therefore, elect to support a smaller session ID bit field. In this L2TPv3 implementation, an upper value for the L2TPv3 session ID was set at 023. The L2TPv3 session ID value 0 is reserved for use by the protocol. For static sessions, the session ID is manually configured.



Note The local session ID must be unique on the decapsulating system and is restricted to the least significant ten bits.

Session Cookie

The L2TPv3 header contains a control channel cookie field. The control channel cookie field has a variable length of 0, 4, or 8 bytes according to the cookie length supported by a given platform for packet decapsulation. The control channel cookie length can be configured manually for static sessions or determined dynamically for dynamic sessions.

The variable cookie length does not present a problem when the same platform is at both ends of an L2TPv3 control channel. However, when different platforms interoperate across an L2TPv3 control channel, both platforms need to encapsulate packets with a 4-byte cookie length.

Pseudowire Control Encapsulation

The L2TPv3 pseudowire control encapsulation consists of 32 bits (4 bytes) and contains information used to sequence L2TP packets (see the Sequencing section). For the purposes of sequencing, only the first bit and bits 8 to 31 are relevant. Bit 1 indicates whether the Sequence Number field, bits 8 to 31, contains a valid sequence number and is to be updated.

L2TPv3 Operation

L2TPv3 includes the following features:

- Xconnect for Layer 2 tunneling through a pseudowire over an IP network
- Layer 2 VPNs for PE-to-PE device service using xconnect that supports Ethernet and VLAN, including both static and dynamic (using the new L2TPv3 signaling) forwarded sessions

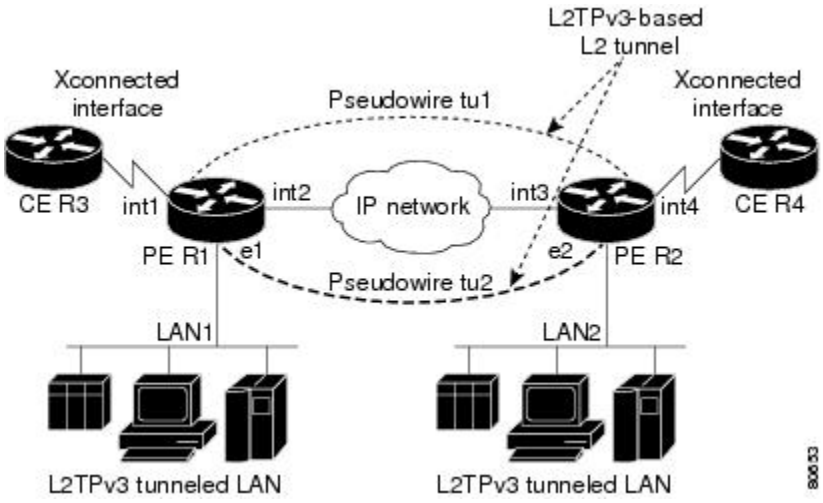
The initial Cisco IOS software supported only the following features:

- Layer 2 tunneling (as used in an L2TP access concentrator or LAC) to an attachment circuit, not Layer 3 tunneling
- L2TPv3 data encapsulation directly over IP (IP protocol number 115), not using the UDP
- Point-to-point sessions, not point-to-multipoint or multipoint-to-point sessions
- Sessions between the same Layer 2 protocols, such as Ethernet-to-Ethernet and VLAN-to-VLAN, but not VLAN-to-Ethernet

The attachment circuit is the physical interface or subinterface attached to the pseudowire.

The figure below shows how the L2TPv3 feature is used for setting 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 devices of the IP network treat the traffic as any other IP traffic and need not know anything about the customer networks.

Figure 30: L2TPv3 Operation



In the figure above, the PE devices R1 and R2 provide L2TPv3 services. The R1 and R2 devices communicate with each other using a pseudowire over the IP backbone network through a path comprising interfaces **int1** and **int2**, the IP network, and interfaces **int3** and **int4**.

Note When you configure SVI on the PE devices, the interfaces **int1** and **int4** act as LAN switching ports

The PE devices communicate with each other using pseudowires (tu1 and tu2) through a path comprising SVIs over an IP network, while the CE devices communicate through a pair of Xconnect Ethernet or VLAN interfaces using an L2TPv3 sessions.

The L2TPv3 session - tu1 is a pseudowire configured between interface **int1** on PE-R1 and interface **int4** on PE-R2. Any traffic arriving on interface **int1** on PE-R1 from CE-R3 is encapsulated and sent through the pseudowire, which is the control channel (tu1) to PE-R2, where the information is decapsulated and sent to CE-R4 from interface **int4** on P2-R2. When CE-R4 needs to send information to CE-R3, the traffic follows the same path, but, in reverse.

Note

- All packets received on interface int1 are forwarded to R4. R3 and R4 cannot detect the intervening network.
- For Ethernet interfaces, any packet received from LAN1 by R1 on Ethernet interface e1 is encapsulated directly in IP and sent through the pseudowire session tu2 to R2 interface e2, where it is sent on LAN2.
- A VLAN on an Ethernet interface can be mapped to an L2TPv3 session.

L2TPv3 Features

L2TPv3 provides xconnect support for Ethernet and VLAN using Static and Dynamic sessions.

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. You can set up static L2TPv3 sessions for a PE device by configuring fixed values for the fields in the L2TP data header. A static L2TPv3 session allows the PE device to tunnel Layer 2 traffic as soon as the attachment circuit to which the session is bound comes up.

Static configuration allows sessions to be established without dynamically negotiating control connection parameters. This means that although sessions are displayed in the **show l2tun session** command output, no control channel information is displayed in the **show l2tun tunnel** command output.



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.

If you use a static L2TPv3 session, you cannot perform circuit interworking, such as 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 (AV) pairs. Each AV pair contains information about the nature of the Layer 2 link being forwarded, including the payload type and virtual circuit (VC) ID.

Multiple L2TP sessions, one for each forwarded Layer 2 circuit, can exist between a pair of PE devices 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. Information such as sequencing configuration is also exchanged. Circuit state changes (UP/DOWN) are conveyed using the set link info (SLI) message.

Control Channel Parameters

The L2TP class configuration procedure creates a template of L2TP control channel parameters that can be inherited by different pseudowire classes. 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 device at each end of the control channel. Configuring L2TP control channel parameters is optional. However, the L2TP class must be configured before it is associated with a pseudowire class (see the Configuring the L2TPv3 Pseudowire task).

L2TPv3 Control Channel Authentication Parameters

Two methods of control channel message authentication are available: the L2TPv3 Control Message Hashing feature and CHAP-style L2TP control channel. The L2TPv3 Control Message Hashing feature introduces a more robust authentication method than the older, CHAP-style L2TP control channel method of authentication. You may choose to enable both the methods of authentication to ensure interoperability with peers that support only one of these methods of authentication, but this configuration will yield control of the authentication

method used on the peer PE device. Enabling both the methods of authentication should be considered as an interim solution to solve backward compatibility issues during software upgrades.

The principal difference between the two methods of authentication lies in the L2TPv3 Control Message Hashing feature using the entire message in the hash instead of computing the hash over selected contents of a received control message. In addition, instead of including the hash digest in only the start control channel replay (SCCRP) and start control channel connected (SCCCN) messages, it includes it in all messages.

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.

The table below shows a compatibility matrix for the different L2TPv3 authentication methods. PE1 is running the new authentication method. The possible authentication configurations for PE1 are shown in the first column. The other columns represent PE2 running software with different available authentication options. The tables cells in these columns indicate compatible configuration options for PE2. If any PE1/PE2 authentication configuration poses ambiguity about the authentication method used, the winning authentication method is indicated in bold. If both the old and new authentication methods are enabled on PE1 and PE2, both types of authentication occur.

Table 24: Compatibility Matrix for L2TPv3 Authentication Methods

PE1 Authentication Configuration	PE2 Supporting Old Authentication[1]	PE2 Supporting New Authentication[2]	PE2 Supporting Old and New Authentication[3]
None	None	None New integrity check	None New integrity check
Old authentication	Old authentication	—	Old authentication **Old authentication** and new authentication **Old authentication** and new integrity check
New authentication	—	New authentication	New authentication Old authentication and **new authentication**
New integrity check	None	None New integrity check	None New integrity check
Old and new authentication	Old authentication	New authentication	Old authentication New authentication **Old and new authentication** **Old authentication** and new integrity check

PE1 Authentication Configuration	PE2 Supporting Old Authentication[1]	PE2 Supporting New Authentication[2]	PE2 Supporting Old and New Authentication[3]
Old authentication and new integrity check	Old authentication	—	Old authentication **Old authentication** and new authentication **Old authentication** and new integrity check

1 Any PE software that supports only the old CHAP-like authentication system.
2 Any PE software that supports only the new message digest authentication and integrity checking authentication system, but does not understand the old CHAP-like authentication system. This type of software may be implemented by other vendors based on the latest L2TPv3 draft.
3 Any PE software that supports both the old CHAP-like authentication and the new message digest authentication and integrity checking authentication system.

Ethernet over L2TPv3

The Ethernet over L2TPv3 feature provides support for Ethernet-based Layer 2 payload tunneling over IP core networks using L2TPv3.

The Ethernet over L2TPv3 feature supports the following like-to-like switching modes:

- Ethernet port mode
- Ethernet VLAN mode
- Ethernet VLAN mode with VLAN rewrite
- Ethernet QinQ and QinAny mode

Note The QinQ over L2TPv3 support feature includes QinAny over L2TPv3, which has a fixed outer VLAN tag and a variable inner VLAN tag.

The Ethernet over L2TPv3 feature supports the following types of internetworking:

- Ethernet port to VLAN (routed)
- Ethernet port to VLAN (bridged)
- QinQ to Ethernet VLAN or Port Interworking (routed)
- QinQ to Ethernet VLAN or Port Interworking (bridged)

Note QinAny Interworking is not a valid configuration because the inner VLAN tag is undetermined.

GEC over L2TPv3

Gigabit EtherChannel (GEC) over Layer 2 Tunneling Protocol Version 3 (L2TPv3) provides support for GEC-based Layer 2 payload tunneling over IP core networks using L2TPv3. GEC also known as *port channel* is integrated with Ethernet and dot1q attachment circuits (ACs).

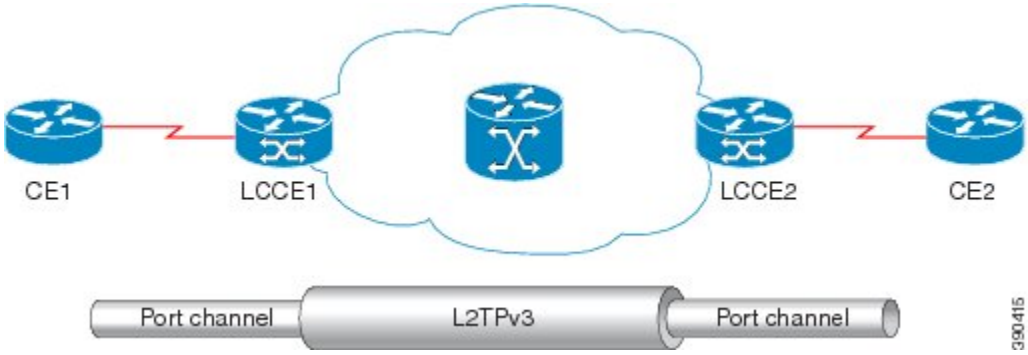
A port channel bundles physical links into a channel group to create a single logical link that provides the aggregate bandwidth of up to eight physical links. If a member port within a port channel fails, the traffic previously carried over the failed link switches to the remaining member ports within the port channel.

Interworking switching is supported in the following scenarios:

- The customer-edge-provider-edge (CE-PE) connecting interface on the local PE is a port-channel interface without dot1q encapsulation. The CE-PE connecting interface on the remote PE is a port-channel interface with dot1q encapsulation.
- The CE-PE connecting interface on the local PE is a port-channel interface with or without dot1q encapsulation. The CE-PE connecting interface on the remote PE is an Ethernet interface with or without dot1q encapsulation.

The figure below illustrates a port channel over IP core networks using L2TPv3. CE1 and CE2 are connected to L2TP Control Connection Endpoints (LCCE) and through port channels. The LCCE is connected to the IP core network using L2TPv3.

Figure 31: GEC over L2TPv3



Sequencing

Although the correct sequence of received Layer 2 frames is guaranteed by some Layer 2 technologies (by the nature of the link such as a serial line) or by the protocol itself, forwarded Layer 2 frames may be lost, duplicated, or reordered when they traverse a network as IP packets. If the Layer 2 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 AV pair when the session is being negotiated. A sender (or one that is manually configured to send sequenced packets) that receives this AV pair uses the Layer 2-specific pseudowire control encapsulation defined in L2TPv3.

You can configure L2TP to drop only out-of-order packets; you cannot configure L2TP to deliver the packets out-of-order. No reordering mechanism is available.

Interworking is not allowed when sequencing is enabled.

L2TPv3 Type of Service Marking

When Layer 2 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 Layer 2 frames themselves encapsulate IP packets, 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."
- You can specify the ToS byte value used by all packets sent across the pseudowire. This is known as "Static ToS byte configuration".

For more details on how to configure ToS, see the Example: Configuring a Negotiated L2TPv3 Session for Local HDLC Switching section.

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 configure sessions manually.

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), SCCRP, and SCCCN control messages. The control channel is responsible for maintaining only 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 peer has gone down through the keepalive mechanism, it sends a StopCCN control message and then notifies all the pseudowires to the peer about the event. This notification results in the teardown of both manually configured and dynamic sessions.

MTU Handling

It is important that you configure a Maximum Transmission Unit (MTU) appropriate for each L2TPv3 tunneled link. The configured MTU size ensures the following:

- The lengths of the tunneled Layer 2 frames fall below the MTU of the destination attachment circuit.
- 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.
- If you enable the **ip dfbit set** command in the pseudowire class, the default MTU behavior changes so that any packets that cannot fit within the tunnel MTU are dropped.
- If you enable the **ip pmtu** command in the pseudowire class, the L2TPv3 control channel participates in the path MTU (PMTU) discovery.

If you enable this feature, the following processing is performed:

- Internet Control Message Protocol (ICMP) unreachable messages sent back to the L2TPv3 device are deciphered and the tunnel MTU is updated accordingly. To receive ICMP unreachable messages for fragmentation errors, the Don’t Fragment (DF) bit in the tunnel header is either set according to the DF

bit value received from the CE device or set statically if the **ip dfbit set** option is enabled. The tunnel MTU is periodically reset to the default value based on a periodic timer.

- ICMP unreachable messages are sent back to the clients on the CE side. ICMP unreachable messages are sent to the CE whenever IP packets arrive on the CE-PE interface and have a packet size greater than the tunnel MTU. A Layer 2 header calculation is performed before the ICMP unreachable message is sent to the CE.

L2TPv3 Control Message Hashing

The L2TPv3 Control Message Hashing feature introduces a new and more secure authentication system that replaces the CHAP-like authentication system inherited from L2TPv2, which uses the Challenge and Challenge Response AV pairs in the SCCRQ, SCCRP, and SCCCN messages. The L2TPv3 Control Message Hashing feature incorporates an optional authentication or integrity check for all control messages.

The per-message authentication introduced by the L2TPv3 Control Message Hashing feature is designed to:

- Perform a mutual authentication between L2TP nodes.
- Check integrity of all control messages.
- Guard against control message spoofing and replay attacks that would otherwise be trivial to mount against the network.

The new authentication method uses the following:

- A computed, one-way hash over the header and body of the L2TP control message
- A preconfigured, shared secret that must be defined on the communicating L2TP nodes
- A local and remote random value exchanged using the Nonce AV pairs

Received control messages that lack any of the required security elements are dropped.

L2TPv3 control message integrity checking is a unidirectional mechanism that does not require the configuration of a shared secret. If integrity checking is enabled on the local PE device, control messages are sent with the message digest calculated without the shared secret or Nonce AV pairs and are verified by the remote PE device. If verification fails, the remote PE device drops the control message.

Enabling the L2TPv3 Control 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 provided 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 may choose to configure control channel authentication or control message integrity checking. Control channel authentication requires participation by both peers and a shared secret must be configured on both devices. Control message integrity check is unidirectional and requires configuration on only one of the peers.

L2TPv3 Control Message Rate Limiting

The L2TPv3 Control Message Rate Limiting feature was introduced to counter the possibility of a denial-of-service (DoS) attack on a device running L2TPv3. The L2TPv3 Control Message Rate Limiting feature limits the rate at which SCCRQ control packets arriving at the PE that terminates the L2TPv3 tunnel

can be processed. SCCRQ control packets initiate the process of bringing up the L2TPv3 tunnel and require a large amount of control plane resources of the PE device.

No configuration is required for the L2TPv3 Control Message Rate Limiting feature. This feature automatically runs in the background in supported releases.

L2TPv3 Digest Secret Graceful Switchover

Authentication of L2TPv3 control channel messages occurs using a password that is configured on all participating peer PE devices. Before the introduction of this feature, changing this password required removing of the old password from the configuration before adding the new password, causing an interruption in L2TPv3 services. The authentication password must be updated on all peer PE devices, which are often at different physical locations. It is difficult for all peer PE devices to be updated with the new password simultaneously to minimize interruptions in L2TPv3 services.

The L2TPv3 Digest Secret Graceful Switchover feature allows the password used to authenticate L2TPv3 control channel messages to be changed without tearing down the established L2TPv3 tunnels. This feature works only for authentication passwords configured with the L2TPv3 Control Message Hashing feature. Authentication passwords configured with the older, CHAP-like authentication system cannot be updated without tearing down L2TPv3 tunnels.

The L2TPv3 Digest Secret Graceful Switchover feature allows two control channel passwords to be configured simultaneously, so a new control channel password can be enabled without first removing the old password. Established tunnels are rapidly updated with the new password, but continue to use the old password until it is removed from the configuration. This allows authentication to continue normally with peer PE devices that have not yet been updated to use the new password. After all peer PE devices are configured with the new password, the old password can be removed from the configuration.

During the period when both a new and an old password are configured, authentication will occur only with the new password if the attempt to authenticate using the old password fails.

L2TPv3 Pseudowire

The pseudowire class configuration procedure creates a configuration template for the pseudowire. Use this template or class to configure session-level parameters for L2TPv3 sessions that are used to transport attachment circuit traffic over the pseudowire.

The pseudowire configuration specifies the characteristics of the L2TPv3 signaling mechanism, including the data encapsulation type, the control protocol, sequencing, Layer 3 fragmentation, payload-specific options, and IP properties. The setting that determines whether signaling is used to set up the pseudowire is also included.

If you specify the **encapsulation l2tpv3** command, you cannot remove it by using the **no encapsulation l2tpv3** command. You also cannot change the command setting by using the **encapsulation mpls** command. These methods result in the following error message:

```
Encapsulation changes are not allowed on an existing pw-class.
```

To remove the command, you must delete the pseudowire by using the **no pseudowire-class** command. To change the type of encapsulation, remove the pseudowire by using the **no pseudowire-class** command, reestablish the pseudowire, and specify the new encapsulation type.

Manual Clearing of L2TPv3 Tunnels

This feature lets you clear L2TPv3 tunnels manually. Before the introduction of this feature, there was no provision to clear a specific L2TPv3 tunnel manually. This functionality provides users more control over an L2TPv3 network.

L2TPv3 Tunnel Management

New and enhanced commands have been introduced to facilitate the management and diagnosis of problems with xconnect configurations. No specific configuration tasks are associated with these commands.

- **debug vpdn**--The output of this command includes authentication failure messages.
- **show l2tun session**--The **hostname** keyword allows the peer hostname to be displayed in the output.
- **show l2tun tunnel**--The **authentication** keyword allows the display of global information about L2TP control channel authentication AV pairs.
- **show xconnect**--The output of this command displays information about xconnect attachment circuits and pseudowires. This command also provides a sortable, single point of reference for information about all xconnect configurations.
- **xconnect logging pseudowire status**--This command enables syslog reporting of pseudowire status events.

For information about these Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the Cisco IOS Master Commands List, All Releases.

L2TPv3 Protocol Demultiplexing

The L2TPv3 Protocol Demultiplexing feature introduces the ability to provide native IPv6 support by utilizing a specialized IPv6 network to offload IPv6 traffic from the IPv4 network. The IPv6 traffic is tunneled to the IPv6 network transparently by using L2TPv3 pseudowires without affecting the configuration of the CE devices. IPv4 traffic is routed as usual within the IPv4 network, maintaining the existing performance and reliability of the IPv4 network.

The IPv4 PE devices must be configured to demultiplex the incoming IPv6 traffic from IPv4 traffic. The PE devices facing the IPv6 network do not require the IPv6 configuration. The configuration of the IPv6 network is beyond the scope of this document. For more information on configuring an IPv6 network, see the IPv6 Configuration Guide .

L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations

The L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations feature lets you configure an Ethertype other than 0x8100 on Gigabit Ethernet interfaces with the QinQ or Dot1Q encapsulation. You can set the custom Ethertype to 0x9100, 0x9200, or 0x88A8. This allows interoperability in a multivendor Gigabit Ethernet environment.

HDLC over L2TPv3

HDLC for Layer 2 Data Encapsulation provides encapsulation of port-to-port Layer 2 traffic. All HDLC traffic including IPv4, IPv6, and non-IP packet, such as IS-IS, is tunneled over L2TPv3. HDLC does not support interworking mode.

Note L2TPv3 supports the IPv4 tunnel only for HDLC. The IPv4 tunnel supports IPv4 and IPv6 packets.

L2TPv3 Benefits

Simplifies Deployment of VPNs

L2TPv3 is an industry-standard Layer 2 tunneling protocol that ensures interoperability among vendors, thus increasing customer flexibility and service availability.

Omits the Need for MPLS

Service providers need not deploy Multiprotocol Label Switching (MPLS) in the core IP backbone to set up VPNs using L2TPv3 over the IP backbone, resulting in operational savings and increased revenue.

Supports Layer 2 Tunneling over IP for Any Payload

L2TPv3 provides enhancements to L2TP to support Layer 2 tunneling of any payload over an IP core network. L2TPv3 defines the base L2TP protocol as being separate from the Layer 2 payload that is tunneled.

Other Benefits

- Provides cookies for authentication
- Provides session state updates and multiple sessions
- Supports interworking (Ethernet-VLAN, Ethernet-QinQ, and VLAN-QinQ)

Supported L2TPv3 Payloads

Note Each L2TPv3 tunneled packet includes the entire Layer 2 frame of the payloads described in this section. If sequencing is required (see the Sequencing section), a Layer 2-specific sublayer (see the Pseudowire Control Encapsulation section) is included in the L2TPv3 header to provide the Sequence Number field.

Ethernet

An Ethernet frame arriving at a PE device is simply encapsulated in its entirety with an L2TP data header. At the other end, a received L2TP data packet is stripped of its L2TP data header. The payload, an Ethernet frame, is then forwarded to the appropriate attachment circuit.

Because the L2TPv3 tunneling protocol serves essentially as a bridge, it need not examine any part of an Ethernet frame. Any Ethernet frame received on an interface is tunneled, and any L2TP-tunneled Ethernet frame is forwarded out of the interface.

Note Because of the way in which L2TPv3 handles Ethernet frames, an Ethernet interface must be configured to promiscuous mode to capture all traffic received on the Ethernet segment attached to the device. All frames are tunneled through the L2TP pseudowire.

VLAN

L2TPv3 supports VLAN memberships in the following ways:

- Port-based, in which undated Ethernet frames are received
- VLAN-based, in which tagged Ethernet frames are received

In L2TPv3, Ethernet xconnect supports port-based VLAN membership and the reception of tagged Ethernet frames. A tagged Ethernet frame contains a tag header (defined in 802.1Q), which is 4 bytes long and consists of a 2-byte tag protocol identifier (TPID) field and a 2-byte tag control information (TCI) field. The TPID indicates that a TCI follows. The TCI is further broken down into the following three fields:

- User priority field
- Canonical format indicator (CFI)
- A 12-bit VLAN ID (VID)

For L2TPv3, an Ethernet subinterface configured to support VLAN switching may be bound to an xconnect service so that all Ethernet traffic, tagged with a VID specified on the subinterface, is tunneled to another PE. The VLAN Ethernet frames are forwarded in their entirety. The receiving PE may rewrite the VID of the tunneled traffic to another value before forwarding the traffic onto an attachment circuit.

To successfully rewrite VLANs, it may be necessary to disable the Spanning Tree Protocol (STP). This can be done on a per-VLAN basis by using the **no spanning-tree vlan** command.

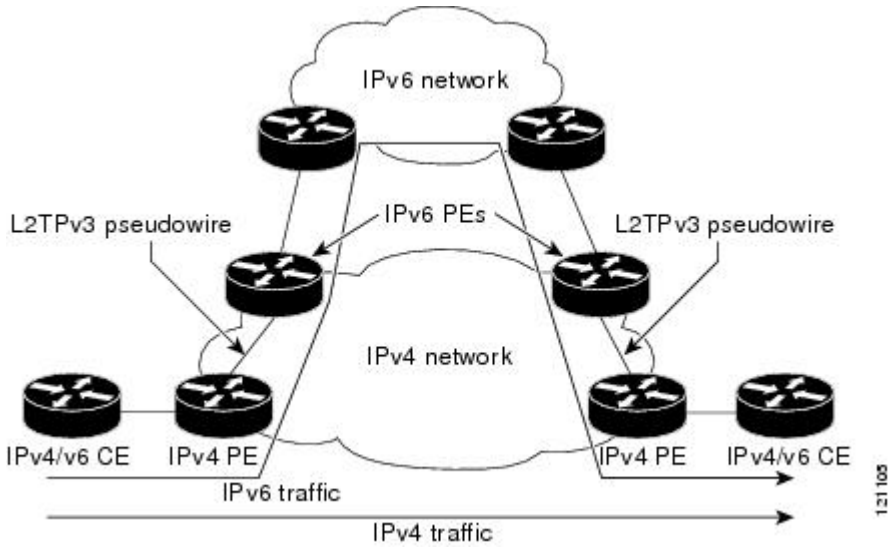
Note Because of the way in which L2TPv3 handles VLAN packets, the Ethernet interface must be configured in promiscuous mode to capture all traffic received on the Ethernet segment attached to the device. All frames are tunneled through the L2TP pseudowire.

IPv6 Protocol Demultiplexing

Upgrading a service provider network to support IPv6 is a long and expensive process. As an interim solution, the Protocol Demultiplexing for L2TPv3 feature introduces the ability to provide native IPv6 support by setting up a specialized IPv6 network and offloading IPv6 traffic from the IPv4 network. IPv6 traffic is tunneled transparently to the IPv6 network using L2TPv3 pseudowires without affecting the configuration of the CE devices. IPv4 traffic is routed as usual within the IPv4 network, maintaining the existing performance and reliability of the IPv4 network.

The figure below shows a network deployment that offloads IPv6 traffic from the IPv4 network to a specialized IPv6 network. The PE devices demultiplex the IPv6 traffic from the IPv4 traffic. IPv6 traffic is routed to the IPv6 network over an L2TPv3 pseudowire, while IPv4 traffic is routed normally. The IPv4 PE devices must be configured to demultiplex the incoming IPv6 traffic from the IPv4 traffic. The PE devices facing the IPv6 network do not require the IPv6 configuration.

Figure 32: Protocol Demultiplexing of IPv6 Traffic from IPv4 Traffic



If no IP address is configured, the protocol demultiplexing configuration is rejected. If an IP address is configured, the **xconnect** command configuration is rejected unless protocol demultiplexing is enabled in xconnect configuration mode before exiting that mode. If an IP address is configured with an **xconnect** command configuration and protocol demultiplexing is enabled, the IP address cannot be removed. To change or remove the configured IP address, the **xconnect** command configuration must first be disabled.

The table below shows the valid combinations of configurations.

Table 25: Valid Configuration Scenarios

Scenario	IP Address	xconnect Configuration	Protocol Demultiplexing Configuration
Routing	Yes	No	--
L2VPN	No	Yes	No
IPv6 Protocol Demultiplexing	Yes	Yes	Yes

Performance Impact of L2TPv3 on Cisco ASR 1000 Series Routers

L2TPv3 supports the following maximum number of attachment circuits and tunnels:

- First-generation Cisco ASR 1000 Series Route Processor (RP1) with Embedded Services Processor 10 (ESP10)
  - Attachment circuits for Ethernet: 8000 per system in a typical user environment. This includes 4000 per port and 8000 per SPA.
  - L2TPv3 tunnels: 1000 (in a typical user environment) and 2000 (maximum).
- Second-generation Cisco ASR 1000 Series Route Processor (RP2) with Embedded Services Processor 20 (ESP20)

- Attachment circuits for Ethernet: 16,000 per system in a typical user environment. This includes 4000 per port and 8000 per SPA.
- L2TPv3 tunnels: 2000 (in a typical user environment) and 4000 (maximum).

L2TPv3 adds tunnel encapsulation to TCP packets, which can cause fragmentation of big packets (packet size larger than the session MTU). Consider a scenario where a big TCP packet is followed by a small TCP packet (packet size smaller than the session MTU). After L2TPv3 encapsulation, the encapsulated big TCP packet will be fragmented, but the encapsulated small TCP packet will not be fragmented. On the Cisco ASR 1000 Series Routers, the fragmentation and reassembly of the big TCP packet requires an additional processor cycle. Because Cisco ASR 1000 Series Routers follow multithread processing, the small packet will need shorter processing time and may be forwarded ahead of the fragmented big packet. This process may result in packet sequence changes on the receiver's end.

As a workaround, you can enable the **ip pmtu** command to prevent the fragmentation of tunneled packets (see the "MTU Handling" section).

Layer 2 Protocol Tunneling and Forwarding

This feature introduces a new functionality for Layer 2 protocol tunneling on ISR platforms. Layer 2 protocol tunneling will tunnel more layer 2 protocols (mvrp/mmrp/elmi/link-oam/esmc/dtp) and forwards all 12 protocols (R4 R5 R6 R8 R9 RA RB RC RD RF stp vtp cdp pagp udld lacp dtp lldp ptppd mvrp mmrp elmi link-oam esmc).

How to Configure Layer 2 Tunneling Protocol Version 3

Configuring L2TP Control Channel Parameters

After 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 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

This task configures a set of timing control channel parameters in an L2TP class. All of the timing control channel parameter configurations are optional and may be configured in any order. If these parameters are not configured, default values are applied.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **l2tp-class** [*l2tp-class-name*]
4. **retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*}
5. **timeout setup** *seconds*
6. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2tp-class** [*l2tp-class-name*] **Example:** `Device(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, to configure multiple L2TP classes, you must specify a unique *l2tp-class-name* for each one.
Step 4	**retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*} **Example:** `Device(config-l2tp-class)# retransmit retries 10`	(Optional) Configures parameters that affect the retransmission of control packets. • **initial retries**—Specifies how many SCCRQs are resent before giving up on the session. Valid values for the *initial-retries* argument range from 1 to 1000. The default value is 2. • **retries**—Specifies how many retransmission cycles occur before determining that the peer PE device is not responding. Valid values for the *retries* argument range from 1 to 1000. The default value is 15. • **timeout** {**max** | **min**}—Specifies maximum and minimum retransmission intervals (in seconds) for resending control packets. Valid values for the *timeout* argument range from 1 to 8. The default maximum interval is 8. The default minimum interval is 1.
Step 5	**timeout setup** *seconds* **Example:** `Device(config-l2tp-class)# timeout setup 400`	(Optional) Configures the amount of time, in seconds, allowed to set up a control channel. • Valid values for the *seconds* argument range from 60 to 6000. The default value is 300.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Device(config-l2tp-class)# exit`	Exits L2TP class configuration mode.

Configuring L2TPv3 Control Channel Authentication Parameters

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 are optional and may be configured in any order. If these parameters are not configured, default values are applied.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2tp-class** [*l2tp-class-name*]
4. **authentication**
5. **password** [**0** | **7**] *password*
6. **hostname** *name*
7. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**l2tp-class** [*l2tp-class-name*] **Example:** `Device(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, to configure multiple L2TP classes, you must specify a unique *l2tp-class-name* for each one.
Step 4	**authentication** **Example:** `Device(config-l2tp-class)# authentication`	(Optional) Enables authentication for the control channel between PE devices.

Step 5	**password** [**0** | **7**] *password*
	Example:
	`Device(config-l2tp-class)# password cisco`
		(Optional) Configures the password used for control channel authentication.
		• [**0** | **7**]—(Optional) 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 password between peer devices.

Step 6	**hostname** *name* **Example:** `Device(config-l2tp-class)# hostname yb2`	(Optional) Specifies a hostname used to identify the device during L2TP control channel authentication. • If you do not use this command, the default hostname of the device is used.
Step 7	**exit** **Example:** `Device(config-l2tp-class)# exit`	Exits L2TP class configuration mode.

Configuring L2TPv3 Control Message Hashing

This task configures L2TPv3 Control Message Hashing feature for an L2TP class.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2tp-class** [*l2tp-class-name*]
4. **digest** [**secret** [**0** | **7**] *password*] [**hash** {**md5** | **sha**}]
5. **digest** check
6. **hidden**
7. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2tp-class** [*l2tp-class-name*] **Example:** `Device(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, to configure multiple L2TP classes, you must specify a unique *l2tp-class-name* for each one.

Step 4	**digest** [**secret** [**0** | **7**] *password*] [**hash** {**md5** | **sha**}]
	Example:
	`Device(config-l2tp-class)# digest secret cisco hash sha`
		(Optional) Enables L2TPv3 control channel authentication or integrity checking.
		• **secret**—(Optional) 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 devices. 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** | **sha**}—(Optional) Specifies the hash function to be used in per-message digest calculations.
		  • **md5**—Specifies HMAC-MD5 hashing.
		  • **sha**—Specifies HMAC-SHA-1 hashing.
		The default hash function is **md5**.

Step 5	**digest** check **Example:**	(Optional) Enables the validation of the message digest in received control messages.

	Command or Action	Purpose
	`Device(config-l2tp-class)# digest check`	• Validation of the message digest is enabled by default. **Note** Validation of the message digest cannot be disabled if authentication has been enabled using the **digest secret** command. If authentication has not been configured with the **digest secret** command, the digest check can be disabled to increase performance.
Step 6	**hidden** **Example:** `Device(config-l2tp-class)# hidden`	(Optional) Enables AV pair hiding when sending control messages to an L2TPv3 peer. • AV pair hiding is disabled by default. • Only the hiding of the cookie AV pair is supported. • If a cookie is configured in L2TP class configuration mode (see the section *"Manually Configuring L2TPv3 Session Parameters*"), enabling AV pair hiding causes that cookie to be sent to the peer as a hidden AV pair using the password configured with the **digest secret** command. **Note** AV pair hiding is enabled only if authentication has been enabled using the **digest secret** command, and no other authentication method is configured.
Step 7	**exit** **Example:** `Device(config-l2tp-class)# exit`	Exits L2TP class configuration mode.

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.

Before you begin

Before performing this task, you must enable control channel authentication as documented in the L2TPv3 Control Message Hashing Restrictions task.

Note This task is not compatible with authentication passwords configured with the older, CHAP-like control channel authentication system.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **l2tp-class** *l2tp-class-name*
4. **digest** [**secret** [**0** | **7**] *password*] [**hash** {**md5** | **sha**}]
5. **end**
6. **show l2tun tunnel all**
7. **configure terminal**
8. **l2tp-class** [*l2tp-class-name*]
9. **no digest** [**secret** [**0** | **7**] *password* [**hash** {**md5** | **sha**}]
10. **end**
11. **show l2tun tunnel all**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2tp-class** *l2tp-class-name* **Example:** `Device(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 4	**digest** [**secret** [**0**	**7**] *password*] [**hash** {**md5**	**sha**}] **Example:** `Device(config-l2tp-class)# digest secret cisco2 hash sha`	Configures a new password to be used in L2TPv3 control channel authentication. • A maximum of two passwords may be configured at any time. **Note** Authentication will now occur using both the old and new passwords.
Step 5	**end** **Example:** `Device(config-l2tp-class)# end`	Ends your configuration session by exiting to privileged EXEC mode.
Step 6	**show l2tun tunnel all** **Example:** `Device# show l2tun tunnel all`	(Optional) Displays the current state of Layer 2 tunnels and information about configured tunnels, including local and remote L2TP hostnames, aggregate packet counts, and control channel information.

	Command or Action	Purpose
		• Tunnels should be updated with the new control channel authentication password within a matter of seconds. If a tunnel does not update to show that two secrets are configured after several minutes have passed, the tunnel can be cleared manually and a defect report should be filed with the Cisco Technical Assistance Center (TAC). To clear an L2TPv3 tunnel manually, perform the task described in the section "Manually Clearing L2TPv3 Tunnels. " **Note** Issue this command to determine whether any tunnel is 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 7	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 8	**l2tp-class** [*l2tp-class-name*] **Example:** `Device(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, to configure multiple L2TP classes, you must specify a unique *l2tp-class-name* for each one.
Step 9	**no digest** [**secret** [**0** | **7**] *password* [**hash** {**md5** | **sha**}] **Example:** `Device(config-l2tp-class)# no digest secret cisco hash sha`	Removes the old password used in L2TPv3 control channel authentication. **Note** Do not remove the old password until all peer PE devices have been updated with the new password.
Step 10	**end** **Example:** `Device(config-l2tp-class)# end`	Ends your configuration session by exiting to privileged EXEC mode.
Step 11	**show l2tun tunnel all** **Example:** `Device# show l2tun tunnel all`	(Optional) Displays the current state of Layer 2 tunnels and information about configured tunnels, including local and remote 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 cleared manually and a defect report should be filed with TAC. To clear an L2TPv3 tunnel manually,

	Command or Action	Purpose
		perform the task described in the section "Manually Clearing L2TPv3 Tunnels." **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

The L2TP hello packet keepalive interval control channel maintenance parameter can be configured in L2TP class configuration mode.

This task configures the interval used for hello messages in an L2TP class. This control channel parameter configuration is optional. If this parameter is not configured, the default value is applied.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2tp-class** [*l2tp-class-name*]
4. **hello** *interval*
5. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2tp-class** [*l2tp-class-name*] **Example:** `Device(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, to configure multiple L2TP classes, you must specify a unique *l2tp-class-name* for each one.

	Command or Action	Purpose
Step 4	**hello** *interval* **Example:** `Device(config-l2tp-class)# hello 100`	(Optional) 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.
Step 5	**exit** **Example:** `Device(config-l2tp-class)# exit`	Exits L2TP class configuration mode.

Configuring the L2TPv3 Pseudowire

Perform this task to configure the L2TPv3 pseudowire.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **pseudowire-class** [*pw-class-name*]
4. **encapsulation l2tpv3**
5. **protocol** {**l2tpv3** | **none**} [*l2tp-class-name*]
6. **ip local interface** *interface-name*
7. **ip pmtu**
8. **ip tos** {**value** *value* | **reflect**}
9. **ip dfbit set**
10. **ip ttl** *value*
11. **ip protocol** {**l2tp** | *protocol-number*}
12. **sequencing** {**transmit** | **receive** | **both**}
13. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**pseudowire-class** [*pw-class-name*] **Example:** `Device(config)# pseudowire-class etherpw`	Enters pseudowire class configuration mode and optionally specifies the name of the L2TP pseudowire class.
Step 4	**encapsulation l2tpv3** **Example:** `Device(config-pw)# encapsulation l2tpv3`	Specifies that L2TPv3 is used as the data encapsulation method to tunnel IP traffic.
Step 5	**protocol** {**l2tpv3** | **none**} [*l2tp-class-name*] **Example:** `Device(config-pw)# protocol l2tpv3 class1`	(Optional) Specifies the L2TPv3 signaling protocol to be used to manage the pseudowires created with the control channel parameters in the specified L2TP class (see the section "Configuring L2TP Control Channel Parameters"). • If the *l2tp-class-name* argument is not specified, the default values for L2TP control channel parameters are used. The default **protocol** option is **l2tpv3**. • If you do not want to use signaling in the L2TPv3 sessions created with this pseudowire class, enter **protocol none**.
Step 6	**ip local interface** *interface-name* **Example:** `Device(config-pw)# ip local interface e0/0`	Specifies the PE device interface whose IP address is to be used as the source IP address for sending tunneled packets. • The same or a different local interface name can be used for each of the pseudowire classes configured between a pair of PE devices. **Note** This command must be configured for pseudowire-class configurations using L2TPv3 as the data encapsulation method.
Step 7	**ip pmtu** **Example:** `Device(config-pw)# ip pmtu`	(Optional) Enables the discovery of the PMTU for tunneled traffic and helps fragmentation. • This command enables the processing of ICMP unreachable messages that indicate fragmentation errors in the backbone network that carries L2TPv3 session traffic. Also, this command enables MTU checking for IP packets sent into the session and that have the DF bit set. Any IP packet larger than the MTU is dropped and an ICMP unreachable message is sent. MTU discovery is disabled by default. **Note** The **ip pmtu** command is not supported if you disabled signaling with the **protocol none** command in Step 5.

	Command or Action	Purpose
		• This command must be enabled in the pseudowire class configuration to enable fragmentation of IP packets before the data enters the pseudowire. **Note** To enable fragmentation of IP packets before the data enters the pseudowire, Cisco recommends that you also enter the **ip dfbit set** command in pseudowire class configuration mode. This allows the PMTU to be obtained more rapidly. **Note** When the **ip pmtu** command is enabled, the DF bit is copied from the inner IP header to the outer IP header. If no IP header is found inside the Layer 2 frame, the DF bit in the outer IP is set to 0.
Step 8	**ip tos** {**value** *value* | **reflect**} **Example:** `Device(config-pw)# ip tos reflect`	(Optional) Configures the value of the ToS byte in IP headers of tunneled packets, or reflects the ToS byte value from the inner IP header. • Valid values for the *value* argument range from 0 to 255. The default ToS byte value is 0.
Step 9	**ip dfbit set** **Example:** `Device(config-pw)# ip dfbit set`	(Optional) Configures the value of the DF bit in the outer headers of tunneled packets. • Use this command if (for performance reasons) you do not want reassembly of tunneled packets on the peer PE device. • This command is disabled by default.
Step 10	**ip ttl** *value* **Example:** `Device(config-pw)# ip ttl 100`	(Optional) Configures the value of the time to live (TTL) byte in the IP headers of tunneled packets. • Valid values for the *value* argument range from 1 to 255. The default TTL byte value is 255.
Step 11	**ip protocol** {**l2tp** | *protocol-number*} **Example:** `Device(config-pw)# ip protocol l2tp`	(Optional) Configures the IP protocol to be used for tunneling packets.
Step 12	**sequencing** {**transmit** | **receive** | **both**} **Example:** `Device(config-pw)# sequencing both`	(Optional) Specifies the direction in which sequencing of data packets in a pseudowire is enabled: • **transmit**—Updates the Sequence Number field in the headers of data packets sent over the pseudowire according to the data encapsulation method that is used.

	Command or Action	Purpose
		• **receive**—Keeps the Sequence Number field in the headers of data packets received over the pseudowire. Out-of-order packets are dropped. • **both**—Enables both the **transmit** and **receive** options.
Step 13	**exit** **Example:** `Device(config-pw)# exit`	Exits pseudowire class configuration mode.

Configuring the Xconnect Attachment Circuit

The virtual circuit identifier that you configure creates the binding between a pseudowire configured on a PE device and an attachment circuit in a CE device. The virtual circuit identifier configured on the PE device at one end of the L2TPv3 control channel must also be configured on the peer PE device at the other end.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot / port*
4. **xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}]
5. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot / port* **Example:** `Device(config)# interface ethernet 0/0`	Specifies the interface by type (for example, Ethernet), slot, and port number, and enters interface configuration mode.
Step 4	**xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit**	**receive**	**both**}] **Example:**	Specifies the IP address of the peer PE device and the 32-bit virtual circuit identifier shared between the PE at each end of the control channel.

	Command or Action	Purpose

	`Device(config-if)# xconnect 10.0.3.201 123 pw-class vlan-xconnect`	• The peer device ID (IP address) and virtual circuit ID must be a unique combination on the device.
		• At least one of the following pseudowire class parameters must be configured for the *pseudowire-parameters* argument:
		  • **encapsulation** {**l2tpv3** [**manual**] | **mpls**}—Specifies the tunneling method used to encapsulate data in the pseudowire:
		  • **l2tpv3**—L2TPv3 is the tunneling method to be used.
		  • **manual**—(Optional) No signaling is to be used in the L2TPv3 control channel. This command places the device in xconnect configuration mode for the manual configuration of L2TPv3 parameters for the attachment circuit.
		  • **mpls**—MPLS is the tunneling method to be used.
		• **pw-class** {*pw-class-name*}—The pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken.
		• The optional **encapsulation** parameter specifies the method of pseudowire tunneling used: L2TPv3 or MPLS. Enter **manual** if you do not want signaling to be used in the L2TPv3 control channel. The **encapsulation l2tpv3 manual** keyword combination enters xconnect configuration submode. See the section "*Manually Configuring L2TPv3 Session Parameters*" for the other L2TPv3 commands that you must enter to complete the configuration of the L2TPv3 control channel. If you do not enter an **encapsulation** value, the encapsulation method entered with the **password** command in the Configuring the Xconnect Attachment Circuit task is used.
		• The optional **pw-class** parameter binds the xconnect statement to a specific pseudowire class. The pseudowire class then serves as the template configuration for all attachment circuits bound to it. Specify the pseudowire-class option if you need to configure more advanced options.
		Note
		You must configure either the **encapsulation** or the **pw-class** option or both.

	Command or Action	Purpose
		Note If you select L2TPv3 as your data encapsulation method, you must specify the **pw-class** keyword. • The optional **sequencing** parameter specifies whether sequencing is required for packets that are received, sent, or both received and sent.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Configure L2TPv3 on a Switched Virtual Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **pseudowire-class** *pw-class-name*
4. **encapsulation l2tpv3**
5. **protocol** {**l2tpv3** | **none**} *l2tp-class-name*
6. **xconnect** *ip address vc-id***encapsulation l2tpv3 pw-class** *pw-class-name*
7. **ip local interface** *interface-name* **loopback**
8. **ip address** *ip address*
9. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pseudowire-class** *pw-class-name* **Example:** `Device(config)# pseudowire-class pc`	Enters pseudowire class configuration mode and specifies the name of the L2TP pseudowire class.

	Command or Action	Purpose
Step 4	**encapsulation l2tpv3** **Example:** `Device(config-pw)# encapsulation l2tpv3`	Specifies L2TPv3, which is used as the data encapsulation method to tunnel IP traffic.
Step 5	**protocol** {**l2tpv3** | **none**} *l2tp-class-name* **Example:** `Device(config-pw)# protocol l2tpv3 class1 pc`	(Optional) Specifies the L2TPv3 signaling protocol to be used to manage the pseudowires created with the control channel parameters in the specified L2TP class (see the section "Configuring L2TP Control Channel Parameters"). If the l2tp-class-name argument is not specified, the default values for L2TP control channel parameters are used. The default **protocol** option is **l2tpv3**. If you do not want to use signaling in the L2TPv3 sessions createdwith this pseudowire class, enter **protocol none**.
Step 6	**xconnect** *ip address vc-id* **encapsulation l2tpv3 pw-class** *pw-class-name* **Example:** `Device(config-if)# xconnect 10.0.3.201 123 encapsulation l2tpv3 pw-class pc`	Specifies the IP address of the peer PE device and the 32-bit virtual circuit identifier shared between the PE at each end of the control channel, and enters xconnect configuration mode. • The peer device ID (IP address) and virtual circuit ID must be a unique combination on the device. • The **encapsulation l2tpv3** parameter specifies that L2TPv3 is to be used as the pseudowire tunneling method. • The mandatory **pw-class** and *pw-class-name* keyword and argument combination specifies the pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken.
Step 7	**ip local interface** *interface-name* **loopback** **Example:** `Device(config-pw)# ip local interface ge0/0/0 loopback0`	Creates a loopback interface and enters interface configuration mode. Specifies the PE device interface whose IP address is to be used as the source IP address for sending tunneled packets. The same or a different local interface name can be used for each ofthe pseudowire classes configured between a pair of PE devices. **Note** This command must be configured for pseudowire-class configurations using L2TPv3 as the data encapsulation method.
Step 8	**ip address** *ip address* **Example:** `Device(config-pw)# ip address 10.1.0.1 255.255.255.255`	Assigns an IP address to the interface.

	Command or Action	Purpose
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Manually Configuring L2TPv3 Session Parameters

When you bind an attachment circuit to an L2TPv3 pseudowire for the xconnect service by using the **xconnect l2tpv3 manual** command (see the section "Configuring the Xconnect Attachment Circuit") because you do not want signaling, you must configure L2TP-specific parameters to complete the L2TPv3 control channel configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot / port*
4. **xconnect** *peer-ip-address vc-id* **encapsulation l2tpv3 manual pw-class** *pw-class-name*
5. **l2tp id** *local-session-id remote-session-id*
6. **l2tp cookie local** *size low-value* [*high-value*]
7. **l2tp cookie remote** *size low-value* [*high-value*]
8. **l2tp hello** *l2tp-class-name*
9. **exit**
10. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot / port* **Example:** `Device(config)# interface ethernet 0/0`	Specifies the interface by type (for example, Ethernet), slot, and port number, and enters interface configuration mode.
Step 4	**xconnect** *peer-ip-address vc-id* **encapsulation l2tpv3 manual pw-class** *pw-class-name*	Specifies the IP address of the peer PE device and the 32-bit virtual circuit identifier shared between the PE at

	Command or Action	Purpose
	Example: `Device(config-if)# xconnect 10.0.3.201 123 encapsulation l2tpv3 manual pw-class vlan-xconnect`	each end of the control channel, and enters xconnect configuration mode. • The peer device ID (IP address) and virtual circuit ID must be a unique combination on the device. • The **encapsulation l2tpv3 manual** parameter specifies that L2TPv3 is to be used as the pseudowire tunneling method. • The mandatory **pw-class** *pw-class-name* keyword and argument combination specifies the pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken.
Step 5	**l2tp id** *local-session-id remote-session-id* **Example:** `Device(config-if-xconn)# l2tp id 222 111`	Configures the identifiers for the local L2TPv3 session and for the remote L2TPv3 session on the peer PE device. • This command is required to complete the attachment circuit configuration and a static L2TPv3 session configuration.
Step 6	**l2tp cookie local** *size low-value* [*high-value*] **Example:** `Device(config-if-xconn)# l2tp cookie local 4 54321`	(Optional) Specifies the value that the peer PE must include in the cookie field of incoming (received) L2TP packets. • The size of the cookie field can be 4 or 8 bytes. If you do not enter this command, no cookie value is included in the header of L2TP packets. • If you configure the cookie length in incoming packets as 8 bytes, you must specify a 4-byte high value and a 4-byte low value.
Step 7	**l2tp cookie remote** *size low-value* [*high-value*] **Example:** `Device(config-if-xconn)# l2tp cookie remote 4 12345`	(Optional) Specifies the value that the device includes in the cookie field of outgoing (sent) L2TP packets. • The size of the cookie field can be 4 or 8 bytes. If you do not enter this command, no cookie value is included in the header of L2TP packets. • If you configure the cookie length in outgoing packets as 8 bytes, you must specify a 4-byte high value and a 4-byte low value.
Step 8	**l2tp hello** *l2tp-class-name* **Example:** `Device(config-if-xconn)# l2tp hello l2tp-defaults`	(Optional) Specifies the L2TP class name to be used (see the section "Configuring L2TP Control Channel Parameters") for control channel configuration parameters, including the interval to use between hello keepalive messages. **Note** This command assumes that there is no control plane to negotiate control channel parameters and that a control

	Command or Action	Purpose
		channel is to be used to provide keepalive support through an exchange of L2TP hello messages. By default, no hello messages are sent.
Step 9	**exit** **Example:** `Device(config-if-xconn)# exit`	Exits xconnect configuration mode.
Step 10	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Configuring Protocol Demultiplexing for L2TPv3

Configuring Protocol Demultiplexing for Ethernet Interfaces

Perform this task to configure the Protocol Demultiplexing feature on an Ethernet interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot / port*
4. **ip address** *ip-address mask* [**secondary**]
5. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*
6. **match protocol ipv6**
7. **exit**
8. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type slot / port* **Example:** `Device(config)# interface ethernet 0/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Device(config-if)# ip address 172.16.128.4`	Sets a primary or secondary IP address for an interface.
Step 5	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Device(config-if)# xconnect 10.0.3.201 888 pw-class demux`	Specifies the IP address of the peer PE device and the 32-bit VCI shared between the PE at each end of the control channel, and enters xconnect configuration mode. • The peer device ID (IP address) and virtual circuit ID must be a unique combination on the device. • **pw-class** *pw-class-name*—The pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken. The **pw-class** parameter binds the xconnect statement to a specific pseudowire class. The pseudowire class then serves as the template configuration for all attachment circuits bound to it. **Note** The L2TPv3 session can also be provisioned manually. See the section "*Manually Configuring L2TPv3 Session Parameters*" for information about manually configuring the L2TPv3 session parameters.
Step 6	**match protocol ipv6** **Example:** `Device(config-if-xconn)# match protocol ipv6`	Enables protocol demultiplexing of IPv6 traffic.
Step 7	**exit** **Example:** `Device(config-if-xconn)# exit`	Exits xconnect configuration mode.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Configuring an L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations

The L2TPv3 Custom Ethertype for dot1q and QinQ Encapsulations feature lets you configure an Ethertype other than 0x8100 on Gigabit Ethernet interfaces with QinQ or dot1Q encapsulations. You can set the custom Ethertype to 0x9100, 0x9200, or 0x88A8. To define the Ethertype field type, you use the **dot1q tunneling ethertype** command.

Perform this task to set a custom Ethertype.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **dot1q tunneling ethertype** {**0x88A8** | **0x9100** | **0x9200**}
5. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/0/0`	Specifies an interface and enters interface configuration mode.
Step 4	**dot1q tunneling ethertype** {**0x88A8**	**0x9100**	**0x9200**} **Example:** `Device(config-if)# dot1q tunneling ethertype 0x9100`	Defines the Ethertype field type used by peer devices when implementing Q-in-Q VLAN tagging.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Configuring GEC over L2TPv3

Perform this task to configure Gigabit EtherChannel (GEC) over Layer 2 Tunneling Protocol Version 3 (L2TPv3).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface Loopback0**
4. **ip address** *ip-address*

5. **exit**
6. **pseudowire-class** [*pw-class-name*]
7. **encapsulation l2tpv3**
8. **ip local interface** *interface-name*
9. **exit**
10. **interface port-channel** *channel-number*
11. **xconnect** *peer-ip-address* **encapsulation l2tpv3 pw-class** *pw-class-name*
12. **exit**
13. **interface gigabitethernet** *interface-type-number*
14. **channel-group** *channel-group-number*
15. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface Loopback0** **Example:** `Device(config)# interface Loopback0`	Creates a loopback interface and enters interface configuration mode.
Step 4	**ip address** *ip-address* **Example:** `Device(config-if)# ip address 10.1.0.1 255.255.255.255`	Assigns an IP address to the interface.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 6	**pseudowire-class** [*pw-class-name*] **Example:** `Device(config)# pseudowire-class l2tpv3`	Enters pseudowire class configuration mode and optionally specifies the name of the Layer 2 Tunneling Protocol (L2TP) pseudowire class.
Step 7	**encapsulation l2tpv3** **Example:** `Device(config-pw)# encapsulation l2tpv3`	Specifies that L2TPv3 is used as the data encapsulation method to tunnel IP traffic.

	Command or Action	Purpose
Step 8	**ip local interface** *interface-name* **Example:** `Device(config-pw)# ip local interface loopback0`	Specifies the provider edge (PE) interface whose IP address is to be used as the source IP address for sending tunneled packets. • Use the same local interface name for all pseudowire classes that are configured between a pair of PE devices. **Note** This command must be configured for pseudowire-class configurations using L2TPv3 as the data encapsulation method.
Step 9	**exit** **Example:** `Device(config-pw)# exit`	Exits pseudowire class configuration mode and enters global configuration mode.
Step 10	**interface** **port-channel** *channel-number* **Example:** `Device# interface port-channel 1`	Defines a port channel and enters interface configuration mode.
Step 11	**xconnect** *peer-ip-address* **encapsulation l2tpv3 pw-class** *pw-class-name* **Example:** `Device(config-subif)# xconnect 10.0.3.201 1234 encapsulation l2tpv3 pw-class l2tpv3`	Specifies the IP address of the peer PE device and the 32-bit virtual circuit identifier (VCI) shared between the PE at each end of the control channel. • The combination of the peer device ID and the VCI must be unique. • **pw-class** *pw-class-name*—The pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken. The **pw-class** parameter binds **xconnect** to a specific pseudowire class. The pseudowire class then serves as a template for all attachment circuits bound to it.
Step 12	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and enters global configuration mode.
Step 13	**interface** **gigabitethernet** *interface-type-number* **Example:** `Device(config)# interface gigabitEthernet 0/0/0`	Enters interface configuration mode.
Step 14	**channel-group** *channel-group-number* **Example:** `Device(config-if)# channel-group 1`	Add the interface to an EtherChannel group.
Step 15	**end** **Example:**	Exits interface configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-if)# end`	

Configuring GEC with Dot1Q

Perform this task to configure Gigabit EtherChannel (GEC) with VLAN over Layer 2 Tunneling Protocol Version 3 (L2TPv3).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *interface-number*
4. **encapsulation dot1q** *vlan-id*
5. **xconnect** *peer-ip-address* **encapsulation l2tpv3 pw-class** *pw-class-name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *interface-number* **Example:** `Device(config)# interface port-channel 1.1`	Defines a port channel and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 100`	Specifies that dot1q is used as the data encapsulation method to tunnel IP traffic.
Step 5	**xconnect** *peer-ip-address* **encapsulation l2tpv3 pw-class** *pw-class-name* **Example:** `Device(config-subif)# xconnect 10.0.3.201 1234 encapsulation l2tpv3 pw-class l2tpv3`	Specifies the IP address of the peer provider edge (PE) device and the 32-bit virtual circuit identifier (VCI) that is shared between the PE device at each end of the control channel. • The combination of the peer device ID and the VCI must be unique.

	Command or Action	Purpose
		• **pw-class** *pw-class-name*—The pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken. The **pw-class** parameter binds **xconnect** to a specific pseudowire class. The pseudowire class then serves as a template for all attachment circuits bound to it.
Step 6	**end** **Example:** `Device# end`	Exits subinterface configuration mode and returns to privileged EXEC mode .

Configuring GEC with QinQ

Perform this task to configure Gigabit EtherChannel (GEC) with queue-in-queue (QinQ) over Layer 2 Tunneling Protocol Version 3 (L2TPv3).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface port-channel** *interface-number*
4. **encapsulation dot1q** *vlan-id* **second-dot1q** *second-vlan-id*
5. **xconnect** *peer-ip-address* **encapsulation l2tpv3 pw-class** *pw-class-name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *interface-number* **Example:** `Device(config)# interface port-channel 1.1`	Defines the subinterface as a port channel and enters subinterface configuration mode.

	Command or Action	Purpose
Step 4	**encapsulation dot1q** *vlan-id* **second-dot1q** *second-vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 100 second-dot1q 200`	Specifies that QinQ is used as the data encapsulation method to tunnel IP traffic.
Step 5	**xconnect** *peer-ip-address* **encapsulation l2tpv3 pw-class** *pw-class-name* **Example:** `Device(config-subif)# xconnect 10.0.3.202 1234 encapsulation l2tpv3 pw-class l2tpv3`	Specifies the IP address of the peer provider edge (PE) device and the 32-bit virtual circuit identifier (VCI) that is shared between the PE device at each end of the control channel. • The combination of the peer device ID and the VCI must be unique. • **pw-class** *pw-class-name*—The pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken. The **pw-class** parameter binds **xconnect** to a specific pseudowire class. The pseudowire class then serves as a template for all attachment circuits bound to it.
Step 6	**end** **Example:** `Device# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Manually Clearing L2TPv3 Tunnels

Perform this task to manually clear a specific L2TPv3 tunnel and all the sessions in that tunnel.

SUMMARY STEPS

1. **enable**
2. **clear l2tun** {**l2tp-class** *l2tp-class-name* | **tunnel id** *tunnel-id* | **local ip** *ip-address* | **remote ip** *ip-address* | **all**}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear l2tun** {**l2tp-class** *l2tp-class-name*	**tunnel id** *tunnel-id*	**local ip** *ip-address*	**remote ip** *ip-address*	**all**}	Clears the specified L2TPv3 tunnel. (This command is not available if there are no L2TPv3 tunnel sessions configured.)

	Command or Action	Purpose
	Example: `Device# clear l2tun tunnel id 56789`	• **l2tp-class** *l2tp-class-name*—All L2TPv3 tunnels with the specified L2TP class name are torn down. • **tunnel id** *tunnel-id*—The L2TPv3 tunnel with the specified tunnel ID are torn down. • **local ip** *ip-address*—All L2TPv3 tunnels with the specified local IP address are torn down. • **remote ip** *ip-address*—All L2TPv3 tunnels with the specified remote IP address are torn down. • **all**—All L2TPv3 tunnels are torn down.

Configuration Examples for Layer 2 Tunneling Protocol Version 3

Note The 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.

Example: Configuring an L2TPv3 Session for an Xconnect Ethernet Interface

L2TPv3 is the only encapsulation method that supports a manually provisioned session setup. This example shows how to configure a static session configuration in which all control channel parameters are set up in advance. There is no control plane used and no negotiation phase to set up the control channel. The PE device starts sending tunneled traffic as soon as the Ethernet interface (int e0/0) comes up. The virtual circuit identifier, 123, is not used. The PE sends L2TP data packets with session ID 111 and cookie 12345. In turn, the PE expects to receive L2TP data packets with session ID 222 and cookie 54321.

```
l2tp-class l2tp-defaults
 retransmit initial retries 30
 cookie-size 8
pseudowire-class ether-pw
 encapsulation l2tpv3
 protocol none
 ip local interface Loopback0
interface Ethernet 0/0
 xconnect 10.0.3.201 123 encapsulation l2tpv3 manual pw-class ether-pw
 l2tp id 222 111
 l2tp cookie local 4 54321
 l2tp cookie remote 4 12345
 l2tp hello l2tp-defaults
```

Example: Configuring a Negotiated L2TPv3 Session for an Xconnect VLAN Subinterface

The following is a sample configuration of a dynamic L2TPv3 session for a VLAN xconnect interface. In this example, only VLAN traffic with a VLAN ID of 5 is tunneled. In the other direction, the L2TPv3 session identified by a virtual circuit identifier of 123 receives forwarded frames whose VLAN ID fields are rewritten to contain the value 5. L2TPv3 is used as both the control plane protocol and the data encapsulation.

```
l2tp-class class1
 authentication
 password secret
pseudowire-class vlan-xconnect
 encapsulation l2tpv3
 protocol l2tpv3 class1
 ip local interface Loopback0
interface Ethernet0/0.1
 encapsulation dot1q 5
 xconnect 10.0.3.201 123 pw-class vlan-xconnect
```

Example: Configure a Static L2TPv3 Session for a SVI

Configure an SVI for various components of an L2TPv3 session:

```
pseudowire-class pc
 encapsulation l2tpv3
 ip local interface Loopback0
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
interface GigabitEthernet0/0/0
 ip address 12.0.0.1 255.255.255.252
interface GigabitEthernet0/1/0
 switchport access vlan 30
 switchport mode access
interface Vlan30
 xconnect 2.2.2.2 4294967295 encapsulation l2tpv3 pw-class pc
```

Example: Configuring a Negotiated L2TPv3 Session for Local HDLC Switching

The following is a sample configuration of a dynamic L2TPv3 session for local HDLC switching. In this example, note that it is necessary to configure two different IP addresses at the endpoints of the L2TPv3 pseudowire because the virtual circuit identifier must be unique for a given IP address.

```
interface loopback 1
 ip address 10.0.0.1 255.255.255.255
interface loopback 2
 ip address 10.0.0.2 255.255.255.255
pseudowire-class loopback1
 encapsulation l2tpv3
 ip local interface loopback1
pseudowire-class loopback2
 encapsulation l2tpv3
 ip local interface loopback2
interface s0/0
 encapsulation hdlc
 xconnect 10.0.0.1 100 pw-class loopback2
interface s0/1
```

```
 encapsulation hdlc
 xconnect 10.0.0.2 100 pw-class loopback1
```

Example: Verifying an L2TPv3 Session

To display information about current L2TPv3 sessions on a device, use the **show l2tun session brief** command.

```
Device# show l2tun session brief
L2TP Session Information Total tunnels 1 sessions 1
LocID      TunID      Peer-address    State     Username, Intf/
                                      sess/cir  Vcid, Circuit
2391726297 2382731778 6.6.6.6         est,UP    100, Gi0/2/0
```

To display detailed information about current L2TPv3 sessions on a device, use the **show l2tun session all** command.

```
Device# show l2tun session all
L2TP Session Information Total tunnels 1 sessions 1
Session id 2391726297 is up, logical session id 36272, tunnel id 2382731778
  Remote session id is 193836624, remote tunnel id 2280318174
  Locally initiated session
  Unique ID is 12
Session Layer 2 circuit, type is Ethernet, name is GigabitEthernet0/2/0
  Session vcid is 100
  Circuit state is UP
    Local circuit state is UP
    Remote circuit state is UP
Call serial number is 98300002
Remote tunnel name is l2tp-asr-2
  Internet address is 6.6.6.6
Local tunnel name is l2tp-asr-1
  Internet address is 3.3.3.3
IP protocol 115
  Session is L2TP signaled
  Session state is established, time since change 00:05:25
    94 Packets sent, 58 received
    9690 Bytes sent, 5642 received
  Last clearing of counters never
  Counters, ignoring last clear:
    94 Packets sent, 58 received
    9690 Bytes sent, 5642 received
    Receive packets dropped:
      out-of-order:             0
      other:                    0
      total:                    0
Send packets dropped:
      exceeded session MTU:     0
      other:                    0
      total:                    0
  DF bit off, ToS reflect disabled, ToS value 0, TTL value 255
  Sending UDP checksums are disabled
  Received UDP checksums are verified
  No session cookie information available
  FS cached header information:
    encap size = 24 bytes
    45000014 00000000 ff73a965 03030303
    06060606 0b8db650
  Sequencing is off
  Conditional debugging is disabled
  SSM switch id is 4101, SSM segment id is 12294
```

Example: Verify a Static L2TPv3 Session for a Switched Virtual Interface

```
show xconnect interface Vlan30 detail
Legend:       XC ST=Xconnect State   S1=Segment1 State    S2=Segment2 State
UP=Up         DN=Down                AD=Admin Down        IA=Inactive
SB=Standby    HS=Hot Standby         RV=Recovering        NH=No Hardware

XC ST     Segment 1                      S1 Segment 2                          S2
------+-----------------------------------+--+-----------------------------------+--
UP pri   ac Vl30:30(Eth VLAN)             UP l2tp 2.2.2.2:4294967295            UP
         Interworking: vlan                    Session ID: 2947605650
                                               Tunnel ID: 3954331565
                                               Peer name: Clarinet-4451
                                               Protocol State: UP
                                               Remote Circuit State: UP
                                               pw-class: pc
```

Example: Verifying an L2TP Control Channel

The L2TP control channel is used to negotiate capabilities, monitor the health of the peer PE device, and set up various components of an L2TPv3 session.

To display information about L2TP control channels to other L2TP-enabled devices for all L2TP sessions on the device, use the **show l2tun tunnel** command.

```
Device# show l2tun tunnel
L2TP Tunnel Information Total tunnels 1 sessions 1
LocTunID   RemTunID   Remote Name   State  Remote Address  Sessn L2TP Class/
                                                          Count VPDN Group
2382731778 2280318174 l2tp-asr-2    est    6.6.6.6         1     l2tp_default_cl
```

To display detailed information about L2TP control channels to other L2TP-enabled devices for all L2TP sessions on the device, use the **show l2tun tunnel all** command.

```
Device# show l2tun tunnel all
L2TP Tunnel Information Total tunnels 1 sessions 1
Tunnel id 2382731778 is up, remote id is 2280318174, 1 active sessions
  Locally initiated tunnel
  Tunnel state is established, time since change 00:02:59
  Tunnel transport is IP  (115)
  Remote tunnel name is l2tp-asr-2
    Internet Address 6.6.6.6, port 0
  Local tunnel name is l2tp-asr-1
    Internet Address 3.3.3.3, port 0
  L2TP class for tunnel is l2tp_default_class
  Counters, taking last clear into account:
    54 packets sent, 35 received
    5676 bytes sent, 3442 received
  Last clearing of counters never
  Counters, ignoring last clear:
    54 packets sent, 35 received
    5676 bytes sent, 3442 received
Control Ns 5, Nr 4
  Local RWS 1024 (default), Remote RWS 1024
  Control channel Congestion Control is disabled
  Tunnel PMTU checking disabled
  Retransmission time 1, max 1 seconds
  Unsent queuesize 0, max 0
  Resend queuesize 0, max 2
  Total resends 0, ZLB ACKs sent 2
  Total out-of-order dropped pkts 0
  Total out-of-order reorder pkts 0
```

```
  Total peer authentication failures 0
  Current no session pak queue check 0 of 5
  Retransmit time distribution: 0 0 0 0 0 0 0 0 0 0
  Control message authentication is disabled
```

Example: Configuring L2TPv3 Control Channel Authentication

The following example shows how to configure CHAP-style authentication of the L2TPv3 control channel:

```
l2tp-class class0
 authentication
 password cisco
```

The following example shows how to configure control channel authentication using the L2TPv3 Control Message Hashing feature:

```
l2tp-class class1
 digest secret cisco hash sha
 hidden
```

The following example shows how to configure control channel integrity checking and how to disable validation of the message digest using the L2TPv3 Control Message Hashing feature:

```
l2tp-class class2
 digest hash sha
 no digest check
```

The following example shows how to disable the validation of the message digest using the L2TPv3 Control Message Hashing feature:

```
l2tp-class class3
 no digest check
```

Example: Configuring L2TPv3 Digest Secret Graceful Switchover

The following example shows how to use the L2TPv3 Digest Secret Graceful Switchover feature to change the L2TP control channel authentication password for the L2TP class named class1. This example assumes that you already have an old password configured for the L2TP class named class1.

```
Device(config)# l2tp-class class1
Device(config-l2tp-class)# digest secret cisco2 hash sha
!
! Verify that all peer PE devices have been updated to use the new password before
! removing the old password.
!
Device(config-l2tp-class)# no digest secret cisco hash sha
```

Example: Verifying L2TPv3 Digest Secret Graceful Switchover

The following **show l2tun tunnel all** command output shows information about the L2TPv3 Digest Secret Graceful Switchover feature:

```
Device# show l2tun tunnel all
! The output below displays control channel password information for a tunnel which has
! been updated with the new control channel authentication password.
!
Tunnel id 12345 is up, remote id is 54321, 1 active sessions
Control message authentication is on, 2 secrets configured
Last message authenticated with first digest secret
!
```

```
! The output below displays control channel password information for a tunnel which has
! only a single control channel authentication password configured.
!
Tunnel id 23456 is up, remote id is 65432, 1 active sessions
!
Control message authentication is on, 1 secrets configured
Last message authenticated with first digest secret
!
! The output below displays control channel password information for a tunnel which is
! communicating with a peer that has only the new control channel authentication password
! configured.
!
Tunnel id 56789 is up, remote id is 98765, 1 active sessions
!
Control message authentication is on, 2 secrets configured
Last message authenticated with second digest secret
```

Example: Configuring a Pseudowire Class for Fragmentation of IP Packets

The following is a sample configuration of a pseudowire class that will allow IP traffic generated from the CE device to be fragmented before entering the pseudowire:

```
pseudowire class class1
 encapsulation l2tpv3
 ip local interface Loopback0
 ip pmtu
 ip dfbit set
```

Example: Configuring Protocol Demultiplexing for L2TPv3

The following example shows how to configure the L2TPv3 Protocol Demultiplexing feature on IPv4 PE devices. The PE devices facing the IPv6 network do not require the IPv6 configuration.

```
interface ethernet 0/1
 ip address 172.16.128.4
 xconnect 10.0.3.201 888 pw-class demux
  match protocol ipv6
```

Example: Manually Clearing an L2TPv3 Tunnel

The following example demonstrates how to manually clear a specific L2TPv3 tunnel using the tunnel ID:

```
clear l2tun tunnel 65432
```

Example: Configuring an L2TPv3 Custom Ethertype for Dot1q and QinQ Encapsulations

The following example shows how to configure an Ethertype other than 0x8100 on Gigabit Ethernet interfaces with QinQ or dot1Q encapsulations. In this example, the Ethertype field is set to 0x9100 on Gigabit Ethernet interface 1/0/0.

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 1/0/0
Device(config-if)# dot1q tunneling ethertype 0x9100
```

Example: Configuring an L2TPv3 HDLC Like-to-Like Layer 2 Transport

Example: Configuring an L2TPv3 HDLC Like-to-Like Layer 2 Transport on Dynamic Mode

The following example shows how to configure xconnect on a serial interface with HDLC encapsulation on a dynamic mode. The dynamic mode uses L2TPv3 signaling in control channel to set up the L2TPv3 tunnel.

```
pseudowire-class 774
 encapsulation l2tpv3
 protocol l2tpv3
 ip local interface GigabitEthernet0/0/1.774
!
interface Serial0/2/0:0
 no ip address
 xconnect 4.4.4.4 200 pw-class 774
```

Example: Configuring an L2TPv3 HDLC Like-to-Like Layer 2 Transport on Static Mode

The following example shows how to configure xconnect on a serial interface with HDLC encapsulation on a static mode. The static mode is used to disable signaling in the L2TPv3 control channel. Since signaling is disabled, you must specify the manual option in xconnect and configure L2TP-specific parameters to complete the L2TPv3 control channel configuration.

```
pseudowire-class pe1-ether-pw
 encapsulation l2tpv3
 protocol none
 ip local interface Loopback1
!
interface Serial0/2/0:0
 no ip address
 xconnect 2.2.2.2 50 encapsulation l2tpv3 manual pw-class pe1-ether-pw
  l2tp id 111 111
  l2tp cookie local 4 54321
  l2tp cookie remote 4 12345
```

Example: Configuring GEC over L2TPv3

The following is a sample configuration of Gigabit EtherChannel (GEC) over Layer 2 Tunneling Protocol Version 3 (L2TPv3):

```
Device> enable
Device# configure terminal
Device(config)# interface Loopback0
Device(config-if)# ip address 10.1.0.1 255.255.255.255
Device(config-if)# exit
Device(config)# pseudowire-class l2tpv3
Device(config-pw)# encapsulation l2tpv3
Device(config-pw)# ip local interface loopback0
Device(config-if)# exit
Device(config)# interface port-channel 1
Device(config-if)# xconnect 1.1.1.1 1234 encapsulation l2tpv3 pw-class l2tpv3
Device(config-if)# exit
Device(config)# interface g0/0/0
Device(config-if)# channel-group 1
Device(config-if)# end
```

Example: Configuring GEC with Dot1q over L2TPv3

The following is a sample configuration of a Gigabit EtherChannel (GEC) with dot1q over Layer 2 Tunneling Protocol Version 3 (L2TPv3):

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 1
Device(config-if)# interface port-channel 1.1
Device(config-subif)# encapsulation dot1q 100
Device(config-subif)# xconnect 10.0.0.2 1234 encapsulation l2tpv3 pw-class l2tpv3
Device(config-subif)# end
```

Example: Configuring GEC with QinQ over L2TPv3

The following is a sample configuration of a Gigabit EtherChannel (GEC) with queue-in-queue (QinQ) over Layer 2 Tunneling Protocol Version 3 (L2TPv3):

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 1
Device(config-if)# interface port-channel 1.1
Device(config-subif)# encapsulation dot1q 100 second-dot1q 200
Device(config-subif)# xconnect 10.0.0.3 1234 encapsulation l2tpv3 pw-class l2tpv3
Device(config-subif)# end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Master Commands List, All Releases
WAN commands: complete command syntax, command mode, defaults, usage guidelines and examples	Wide-Area Networking Command Reference
Layer 2 Tunnel Protocol Version 3	*Layer 2 Tunneling Protocol Version 3*
Any Transport over MPLS	*Any Transport over MPLS*
Cisco 12000 series routers hardware support	*Cross-Platform Release Notes for Cisco IOS Release 12.0S*
Cisco 7600 series routers hardware support	*Cross-Platform Release Notes for Cisco IOS Release 12.2SR*
Cisco 3270 series routers hardware support	*Release Notes for Cisco IOS Software Release 12.2SE*

Standards and RFCs

Standard/RFC	Title
draft-ietf-l2tpext-l2tp-base-03.txt	*Layer Two Tunneling Protocol (Version 3) 'L2TPv3'*
draft-martini-l2circuit-trans-mpls-09.txt	*Transport of Layer 2 Frames Over MPLS*
draft-ietf-pwe3-frame-relay-03.txt.	*Encapsulation Methods for Transport of Frame Relay over MPLS Networks*
draft-martini-l2circuit-encap-mpls-04.txt.	*Encapsulation Methods for Transport of Layer 2 Frames Over IP and MPLS Networks*
draft-ietf-pwe3-ethernet-encap-08.txt.	*Encapsulation Methods for Transport of Ethernet over MPLS Networks*
draft-ietf-pwe3-hdlc-ppp-encap-mpls-03.txt.	*Encapsulation Methods for Transport of PPP/HDLC over MPLS Networks*
draft-ietf-ppvpn-l2vpn-00.txt.	*An Architecture for L2VPNs*

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Glossary

AV pairs—Attribute-value pairs.

CEF—Cisco Express Forwarding. The Layer 3 IP switching technology that optimizes network performance and scalability for networks with large and dynamic traffic patterns.

data-link control layer—Layer 2 in the SNA architectural model. Responsible for the transmission of data over a particular physical link. Corresponds approximately to the data link layer of the OSI model.

DCE—Data circuit-terminating equipment (ITU-T expansion). Devices and connections of a communications network that comprise the network end of the user-to-network interface.

DF bit—Don't Fragment bit. The bit in the IP header that can be set to indicate that the packet should not be fragmented.

DTE—Data terminal equipment. The device at the user end of a user-network interface that serves as a data source, destination, or both.

HDLC—High-Level Data Link Control. A generic link-level communications protocol developed by the ISO. HDLC manages synchronous, code-transparent, serial information transfer over a link connection.

ICMP—Internet Control Message Protocol. A network protocol that handles network errors and error messages.

IDB— Interface descriptor block.

IS-IS—Intermediate System-to-Intermediate System. The OSI link-state hierarchical routing protocol based on DECnet Phase V routing, whereby ISs (devices) exchange routing information based on a single metric to determine network topology.

L2TP—An extension to PPP that merges features of two tunneling protocols: Layer 2 Forwarding (L2F) from Cisco Systems and Point-to-Point Tunneling Protocol (PPTP) from Microsoft. L2TP is an IETF standard endorsed by Cisco Systems and other networking industry leaders.

L2TPv3—The draft version of L2TP that enhances functionality in RFC 2661 (L2TP).

LMI—Local Management Interface.

MPLS—Multiprotocol Label Switching. A switching method that forwards IP traffic using a label. This label instructs the devices in the network where to forward packets based on preestablished IP routing information.

MQC—Modular quality of service CLI.

MTU—Maximum Transmission Unit. The maximum packet size, in bytes, that a particular interface can handle.

PMTU—Path MTU.

PVC—Permanent virtual circuit. A virtual circuit that is permanently established. A Frame Relay logical link, whose endpoints and class of service are defined by network management. Analogous to an X.25 permanent virtual circuit, a PVC consists of the originating Frame Relay network element address, originating data-link control identifier, terminating Frame Relay network element address, and termination data-link control identifier. Originating refers to the access interface from which the PVC is initiated. Terminating refers to the access interface at which the PVC stops. Many data network customers require a PVC between two points. PVCs save the bandwidth associated with circuit establishment and tear down in situations where certain virtual circuits must exist all the time. Data terminating equipment with a need for continuous communication uses PVCs.

PW—Pseudowire.

SNMP—Simple Network Management Protocol. The network management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices and manage configurations, statistics collection, performance, and security.

tunneling—Architecture that is designed to provide the services necessary to implement any standard point-to-point encapsulation scheme.

UNI—User-Network Interface.

VPDN—Virtual private dialup network. A network that allows separate and autonomous protocol domains to share common access infrastructure, including modems, access servers, and ISDN devices. A VPDN enables users to configure secure networks that take advantage of ISPs that tunnel remote access traffic through the ISP cloud.



CHAPTER 12

L2VPN Pseudowire Redundancy

The L2VPN Pseudowire Redundancy feature lets you configure your network to detect a failure in the network and reroute the Layer 2 (L2) service to another endpoint that can continue to provide service. This feature provides the ability to recover from a failure either of the remote provider edge (PE) router or of the link between the PE and customer edge (CE) routers.

- Prerequisites for L2VPN Pseudowire Redundancy, on page 231
- Restrictions for L2VPN Pseudowire Redundancy, on page 232
- Information About L2VPN Pseudowire Redundancy, on page 232
- How to Configure L2VPN Pseudowire Redundancy, on page 234
- Configuration Examples for L2VPN Pseudowire Redundancy, on page 245
- Configuration Examples for L2VPN Pseudowire Redundancy using the commands associated with the L2VPN Protocol-Based CLIs feature, on page 247
- Additional References, on page 251
- Feature Information for L2VPN Pseudowire Redundancy, on page 252

Prerequisites for L2VPN Pseudowire Redundancy

- This feature module requires that you understand how to configure basic L2 virtual private networks (VPNs).
  - Any Transport over MPLS
  - L2 VPN Interworking
  - Layer 2 Tunneling Protocol Version 3 (L2TPv3)
- The L2VPN Pseudowire Redundancy feature requires that the following mechanisms be in place to enable you to detect a failure in the network:
  - Label-switched paths (LSP) Ping/Traceroute and Any Transport over MPLS Virtual Circuit Connection Verification (AToM VCCV)
  - Local Management Interface (LMI)
  - Operation, Administration, and Maintenance (OAM)

Restrictions for L2VPN Pseudowire Redundancy

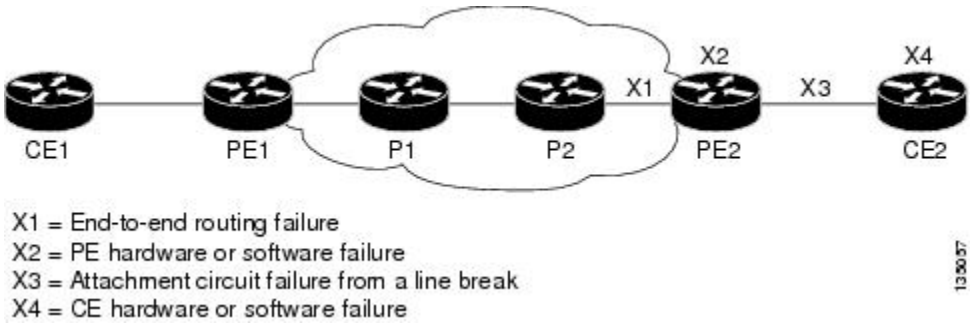
- The default Label Distribution Protocol (LDP) session hold-down timer will enable the software to detect failures in about 180 seconds. That time can be configured so that the software can detect failures more quickly. See the **mpls ldp holdtime** command for more information.
- L2VPN Pseudowire Redundancy does not support pseudowire interworking mode with L2TPv3. The connectivity between CEs may be impacted if you have interworking IP configured in the pseudowire class.
- The primary and backup pseudowires must run the same type of transport service. The primary and backup pseudowires must be configured with AToM or L2TPv3.
- The backup peer can only be configured for nonstatic L2TPv3 sessions. The backup L2TPv3 session cannot be static L2TPv3 session. The encapsulation type of primary and backup pseudowire must be the same.
- If you use L2VPN Pseudowire Redundancy with L2VPN Interworking, the interworking method must be the same for the primary and backup pseudowires.
- L2VPN Pseudowire Redundancy does support setting the experimental (EXP) bit on the Multiprotocol Label Switching (MPLS) pseudowire.
- L2VPN Pseudowire Redundancy does not support different pseudowire encapsulation types on the MPLS pseudowire.
- The **mpls l2transport route** command is not supported. Use the **xconnect** command instead.
- The ability to have the backup pseudowire fully operational at the same time that the primary pseudowire is operational is not supported. The backup pseudowire becomes active only after the primary pseudowire fails.
- The AToM VCCV feature is supported only on the active pseudowire.
- More than one backup pseudowire is not supported.

Information About L2VPN Pseudowire Redundancy

Introduction to L2VPN Pseudowire Redundancy

L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data can take over. However, there are some parts of the network where this rerouting mechanism does not protect against interruptions in service. The figure below shows those parts of the network that are vulnerable to an interruption in service.

Figure 33: Points of Potential Failure in an L2VPN Network

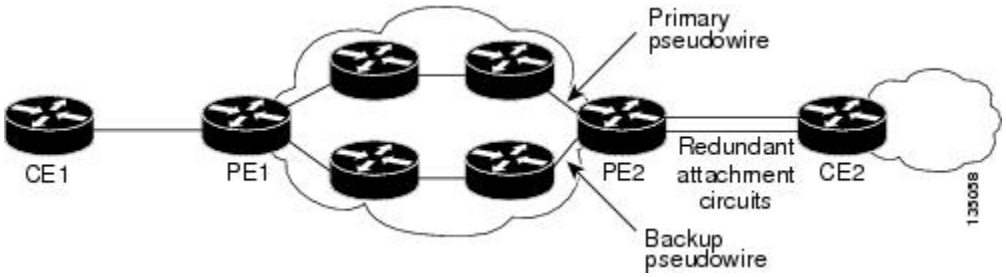


The L2VPN Pseudowire Redundancy feature provides the ability to ensure that the CE2 router in the figure above can always maintain network connectivity, even if one or all the failures in the figure occur.

The L2VPN Pseudowire Redundancy feature enables you to set up backup pseudowires. You can configure the network with redundant pseudowires and redundant network elements, which are shown in the three figures below.

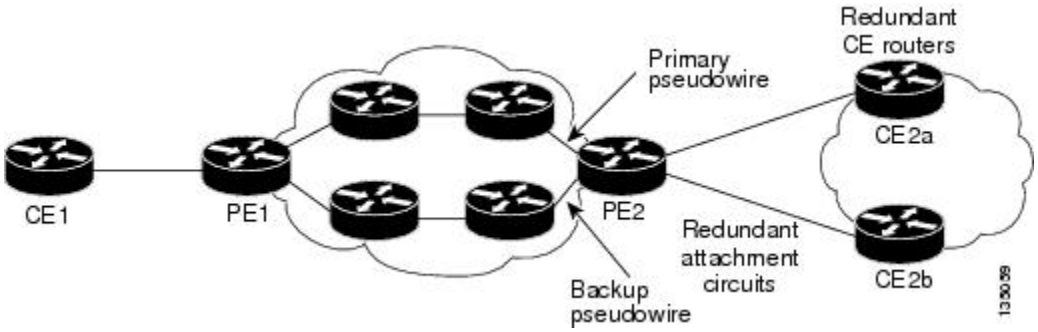
The figure below shows a network with redundant pseudowires and redundant attachment circuits.

Figure 34: L2VPN Network with Redundant PWs and Attachment Circuits



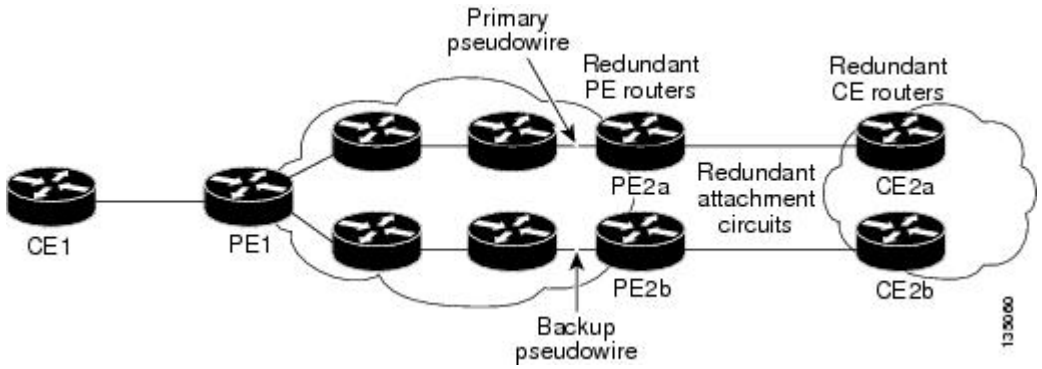
The figure below shows a network with redundant pseudowires, attachment circuits, and CE routers.

Figure 35: L2VPN Network with Redundant PWs, Attachment Circuits, and CE Routers



The figure below shows a network with redundant pseudowires, attachment circuits, CE routers, and PE routers.

Figure 36: L2VPN Network with Redundant PWs, Attachment Circuits, CE Routers, and PE Routers



How to Configure L2VPN Pseudowire Redundancy

The L2VPN Pseudowire Redundancy feature enables you to configure a backup pseudowire in case the primary pseudowire fails. When the primary pseudowire fails, the PE router can switch to the backup pseudowire. You can have the primary pseudowire resume operation after it comes back up.

Configuring the Pseudowire

The successful transmission of the Layer 2 frames between PE routers is due to the configuration of the PE routers. You set up the connection, called a pseudowire, between the routers.

The pseudowire-class configuration group specifies the characteristics of the tunneling mechanism, which are:

- Encapsulation type
- Control protocol
- Payload-specific options

You must specify the **encapsulation mpls**command as part of the pseudowire class for the AToM VCs to work properly. If you omit the **encapsulation mpls**command as part of the **xconnect**command, you receive the following error:

```
% Incomplete command.
```

Perform this task to configure a pseudowire class.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **pseudowire-class** name
4. **encapsulation mpls**
5. **interworking** {**ethernet** | **ip**}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**pseudowire-class** name **Example:** `Router(config)# pseudowire-class atom`	Establishes a pseudowire class with a name that you specify. Enters pseudowire class configuration mode.
Step 4	**encapsulation mpls** **Example:** `Router(config-pw-class)# encapsulation mpls`	Specifies the tunneling encapsulation. For AToM, the encapsulation type is **mpls**.
Step 5	**interworking** {**ethernet** | **ip**} **Example:** `Router(config-pw-class)# interworking ip`	(Optional) Enables the translation between the different Layer 2 encapsulations.

Configuring the Pseudowire using the commands associated with the L2VPN Protocol-Based CLIs feature

The successful transmission of the Layer 2 frames between PE routers is due to the configuration of the PE routers. You set up the connection, called a pseudowire, between the routers.

The pseudowire-class configuration group specifies the characteristics of the tunneling mechanism, which are:

- Encapsulation type
- Control protocol
- Payload-specific options

You must specify the **encapsulation mpls**command as part of the pseudowire class for the AToM VCs to work properly. If you omit the **encapsulation mpls**command as part of the **l2vpn xconnectcontext** command, you receive the following error:

```
% Incomplete command.
```

Perform this task to configure a pseudowire class.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface pseudowire** *number*
4. **encapsulation mpls**
5. **neighbor** *peer-address vcid-value*
6. **interworking** {**ethernet** | **ip**}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface pseudowire** *number* **Example:** `Router(config)# interface pseudowire 1`	Establishes an interface pseudowire with a value that you specify. Enters pseudowire configuration mode.
Step 4	**encapsulation mpls** **Example:** `Router(config-pw)# encapsulation mpls`	Specifies the tunneling encapsulation. For AToM, the encapsulation type is **mpls**.
Step 5	**neighbor** *peer-address vcid-value* **Example:** `Router(config-pw)#  neighbor 10.0.0.1 123`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 6	**interworking** {**ethernet**	**ip**} **Example:** `Router(config-pw)# interworking ip`	(Optional) Enables the translation between the different Layer 2 encapsulations.

Configuring L2VPN Pseudowire Redundancy

Perform this task to configure the L2VPN Pseudowire Redundancy feature.

Before you begin

For each transport type, the **xconnect**command is configured slightly differently. The following configuration steps use Ethernet VLAN over MPLS, which is configured in subinterface configuration mode. See *Any Transport over MPLS* to determine how to configure the **xconnect** command for other transport types.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *slot* / *subslot* / *interface* **.** *subinterface*
4. **encapsulation dot1q** *vlan-id*
5. **xconnect** *peer-router-id vcid* {**encapsulation mpls**| **pw-class** *pw-class-name}*
6. **backup peer** *peer-router-ip-addr vcid* [**pw-class** *pw-class-name*]
7. **backup delay** *e nable-delay* {*disable-delay* | **never**}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot* / *subslot* / *interface* **.** *subinterface* **Example:** `Router(config)# interface gigabitethernet0/0/0.1`	Specifies the Gigabit Ethernet subinterface and enters subinterface configuration mode. **Note** Make sure that the subinterface on the adjoining CE router is on the same VLAN as this PE router.
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `Router(config-subif)# encapsulation dot1q 100`	Enables the subinterface to accept 802.1Q VLAN packets. **Note** The subinterfaces between the CE and PE routers that are running Ethernet over MPLS must be in the same subnet.

	Command or Action	Purpose
Step 5	**xconnect** *peer-router-id vcid* {**encapsulation mpls**| **pw-class** *pw-class-name}* **Example:** `Router(config-subif)# xconnect 10.0.0.1 123 pw-class atom`	Binds the attachment circuit to a pseudowire VC and enters xconnect configuration mode. • The syntax for this command is the same as for all other Layer 2 transports.
Step 6	**backup peer** *peer-router-ip-addr vcid* [**pw-class** *pw-class-name*] **Example:** `Router(config-if-xconn)# backup peer 10.0.0.3 125 pw-class atom`	Specifies a redundant peer for the pseudowire VC. The pseudowire class name must match the name that you specified when you created the pseudowire class, but you can use a different pw-class in the **backup peer** command than the name that you used in the primary **xconnect** command.
Step 7	**backup delay** *e nable-delay* {*disable-delay* | **never**} **Example:** `Router(config-if-xconn)# backup delay 5 never`	Specifies how long (in seconds) the backup pseudowire VC should wait to take over after the primary pseudowire VC goes down. The range is from 0 to 180. Specifies how long the primary pseudowire should wait after it becomes active to take over for the backup pseudowire VC. The range is from 0 to 180 seconds. If you specify the **never** keyword, the primary pseudowire VC never takes over for the backup.

Configuring L2VPN Pseudowire Redundancy using the commands associated with the L2VPN Protocol-Based CLIs feature

Perform this task to configure the L2VPN Pseudowire Redundancy feature.

Before you begin

For each transport type, the **l2vpn xconnect context** command is configured slightly differently. The following configuration steps use Ethernet VLAN over MPLS, which is configured in subinterface configuration mode. See *Any Transport over MPLS* to determine how to configure the **l2vpn xconnect context** command for other transport types.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet** *slot* / *subslot* / *interface* **.** *subinterface*
4. **encapsulation dot1q** *vlan-id*
5. **end**
6. **interface pseudowire** *number*
7. **source template type pseudowire** *template-name*
8. **neighbor** *peer-address vcid-value*
9. **exit**

10. **l2vpn xconnect context** *context-name*
11. **member pseudowire** *interface-number*
12. **member pseudowire** *interface-number*
13. **member gigabitethernet** *interface-number*
14. **redundancy delay** *enable-delay*{*disable-delay* | **never**}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot* / *subslot* / *interface* **.** *subinterface* **Example:** `Device(config)# interface gigabitethernet0/0/0.1`	Specifies the Gigabit Ethernet subinterface and enters subinterface configuration mode. Make sure that the subinterface on the adjoining CE router is on the same VLAN as this PE router.
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 100`	Enables the subinterface to accept 802.1Q VLAN packets. The subinterfaces between the CE and PE routers that are running Ethernet over MPLS must be in the same subnet. All other subinterfaces and backbone routers do not.
Step 5	**end** **Example:** `Router(config-subif)# end`	Exits to privileged EXEC mode.
Step 6	**interface pseudowire** *number* **Example:** `Router(config)# interface pseudowire 100`	Specifies the pseudowire interface and enters interface configuration mode.
Step 7	**source template type pseudowire** *template-name* **Example:** `Router(config-if)# source template type pseudowire atom`	Configures the source template of type pseudowire named atom

	Command or Action	Purpose
Step 8	**neighbor** *peer-address vcid-value* **Example:** `Router(config-if)# neighbor 10.0.0.1 123`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 9	**exit** **Example:** `Router(config-if)# exit`	Exits to privileged EXEC mode.
Step 10	**l2vpn xconnect context** *context-name* **Example:** `Router(config)# l2vpn xconnect context con1`	Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.
Step 11	**member pseudowire** *interface-number* **Example:** `Device(config-xconnect)# member pseudowire 100 group GR_1 priority 2`	Specifies a member pseudowire to form a Layer 2 VPN (L2VPN) cross connect.
Step 12	**member pseudowire** *interface-number* **Example:** `Device(config-xconnect)# member pseudowire 1001 group GR_1 priority 2`	Specifies a second member pseudowire for redundancy.
Step 13	**member gigabitethernet** *interface-number* **Example:** `Device(config-xconnect)# member GigabitEthernet0/0/0.1 service instance 1`	Specifies the location of the Gigabit Ethernet member interface.
Step 14	**redundancy delay** *enable-delay*{*disable-delay* | **never**} **Example:** `Device(config-xconnect)# redundancy delay 0 0 group GR_1`	Specifies how long (in seconds) the backup pseudowire VC should wait to take over after the primary pseudowire VC goes down. The range is 0 to 180. Specifies how long the primary pseudowire should wait after it becomes active to take over for the backup pseudowire VC. The range is 0 to 180 seconds. If you specify the **never keyword**, the primary pseudowire VC never takes over for the backup.

Forcing a Manual Switchover to the Backup Pseudowire VC

To force the router switch over to the backup or primary pseudowire, you can enter the **xconnect backup force switchover** command in privileged EXEC mode. You can specify either the interface of the primary attachment circuit (AC) to switch to or the IP address and VC ID of the peer router.

A manual switchover can be made only if the interface or peer specified in the command is actually available and the xconnect moves to the fully active state when executing the command.

SUMMARY STEPS

1. **enable**
2. **xconnect backup force-switchover {** **interface** *interface-info* | **peer** *ip-address vcid}*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**xconnect backup force-switchover {** **interface** *interface-info*	**peer** *ip-address vcid}* **Example:** `Router# xconnect backup force-switchover peer 10.10.10.1 123`	Specifies that the router should switch to the backup or to the primary pseudowire.

Verifying the L2VPN Pseudowire Redundancy Configuration

Perform this task to verify that the L2VPN Pseudowire Redundancy feature is correctly configured.

SUMMARY STEPS

1. **show mpls l2transport vc**
2. **show xconnect all**
3. **xconnect logging redundancy**

DETAILED STEPS

Procedure

Step 1 **show mpls l2transport vc**

The following is sample output from the **show mpls l2transport vc** command. In this example, the primary attachment circuit is up. The backup attachment circuit is available, but not currently selected.

Example:

```
Router# show mpls l2transport vc
```

```
Local intf     Local circuit           Dest address    VC ID      Status
-------------  ----------------------- --------------- ---------- ----------
Et0/0.1        Eth VLAN 101            10.0.0.2        101        UP
Et0/0.1        Eth VLAN 101            10.0.0.3        201        DOWN
Router# show mpls l2transport vc detail
Local interface: Et0/0.1 up, line protocol up, Eth VLAN 101 up
   Destination address 10.0.0.2 VC ID: 101, VC status UP
   .
   .
   .
Local interface: Et0/0.1 down, line protocol down, Eth VLAN 101 down
   Destination address 10.0.0.3 VC ID: 201, VC status down
   .
   .
   .
```

Step 2 **show xconnect all**

In this example, the topology is Attachment Circuit 1 to Pseudowire 1 with a Pseudowire 2 as a backup:

Example:

```
Router# show xconnect all
Legend: XC ST=Xconnect State, S1=Segment1 State, S2=Segment2 State
UP=Up, DN=Down, AD=Admin Down, IA=Inactive, NH=No Hardware
XC ST  Segment 1                         S1 Segment 2                         S2
------+---------------------------------+--+---------------------------------+--
UP pri ac   Et0/0(Ethernet)              UP mpls 10.55.55.2:1000              UP
IA sec ac   Et0/0(Ethernet)              UP mpls 10.55.55.3:1001              DN
```

In this example, the topology is Attachment Circuit 1 to Attachment Circuit 2 with a pseudowire backup for Attachment Circuit 2:

Example:

```
Router# show xconnect all
Legend: XC ST=Xconnect State, S1=Segment1 State, S2=Segment2 State
UP=Up, DN=Down, AD=Admin Down, IA=Inactive, NH=No Hardware
XC ST  Segment 1                         S1 Segment 2                         S2
------+---------------------------------+--+---------------------------------+--
UP pri ac   Se6/0:150(FR DLCI)           UP ac   Se8/0:150(FR DLCI)           UP
IA sec ac   Se6/0:150(FR DLCI)           UP mpls 10.55.55.3:7151              DN
```

Step 3 **xconnect logging redundancy**

In addition to the **show mpls l2transport vc** command and the **show xconnect** command, you can use the **xconnect logging redundancy** command to track the status of the xconnect redundancy group:

Example:

```
Router(config)# xconnect logging redundancy
```

When this command is configured, the following messages are displayed during switchover events:

Activating the primary member:

Example:

```
00:01:07: %XCONNECT-5-REDUNDANCY: Activating primary member 10.55.55.2:1000
```

Activating the backup member:

Example:

```
00:01:05: %XCONNECT-5-REDUNDANCY: Activating secondary member 10.55.55.3:1001
```

Verifying the L2VPN Pseudowire Redundancy Configuration using the commands associated with the L2VPN Protocol-Based CLIs feature

Use the following commands to verify that the L2VPN Pseudowire Redundancy feature is correctly configured.

SUMMARY STEPS

1. **show l2vpn atom vc**
2. **show l2vpn service all**
3. **logging redundancy**
4. **logging pseudowire status**

DETAILED STEPS

Procedure

Step 1 **show l2vpn atom vc**

In this example, the primary attachment circuit is up. The backup attachment circuit is available, but not currently selected. The **show** output displays as follows:

Example:

```
Device# show l2vpn atom vc
Local intf     Local circuit           Dest address    VC ID      Status
-------------  ----------------------- --------------- ---------- ----------
Et0/0.1        Eth VLAN 101            10.0.0.2        101        UP
Et0/0.1        Eth VLAN 101            10.0.0.3        201        DOWN
Router# show l2vpn atom vc detail
Local interface: Et0/0.1 up, line protocol up, Eth VLAN 101 up
   Destination address 10.0.0.2 VC ID: 101, VC status UP
   .
   .
   .
Local interface: Et0/0.1 down, line protocol down, Eth VLAN 101 down
   Destination address 10.0.0.3 VC ID: 201, VC status down
   .
   .
   .
```

Step 2 **show l2vpn service all**

In this example, the topology is attachment circuit 1 to pseudowire 1 with apPseudowire 2 as a backup:

Example:

```
Device# show l2vpn service all
Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down             AD=Admin Down     IA=Inactive
```

```
        SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                  Prio  St  XC St
  ---------          -----       -------------                  ----  --  -----
VPWS name: foo, State: UP
  Eth1/1.1                       Eth1/1.1:100(Eth VLAN)         0     UP  UP
  pw101              blue        102.1.1.1:100(MPLS)            2     UP  UP
  pw102              blue        103.1.1.1:100(MPLS)            5     SB  IA
  pw103              blue        104.1.1.1:100(MPLS)            8     SB  IA
  pw104              blue        105.1.1.1:100(MPLS)            11    SB  IA
```

In this example, the topology is attachment circuit 1 to attachment circuit 2 with a pseudowire backup for attachment circuit 2:

Example:

```
Device# show l2vpn service all
Legend: XC ST=Xconnect State, S1=Segment1 State, S2=Segment2 State
UP=Up, DN=Down, AD=Admin Down, IA=Inactive, NH=No Hardware
XC ST  Segment 1                       S1 Segment 2                          S2
------+--------------------------------+--+--------------------------------+--
UP pri ac   Se6/0:150(FR DLCI)         UP ac   Se8/0:150(FR DLCI)           UP
IA sec ac   Se6/0:150(FR DLCI)         UP mpls 10.55.55.3:7151              DN
```

Step 3 **logging redundancy**

In addition to the **show l2vpn atom vc** command and the **show l2vpn service** command, you can use the **logging redundancy** command to enable system message log (syslog) reporting of xconnect redundancy status events:

Example:

```
Device(config)# l2vpn
Device(config-l2vpn)# logging redundancy
```

When this command is configured, the messages below will be generated during switchover events:

Activating the primary member:

Example:

```
Device(config)# l2vpn
Device(config-l2vpn)# logging pseudowire status
```

When this command is configured, this is configured the status of the pseudowire can be monitored:

Activating the primary member:

Example:

```
00:01:07: %XCONNECT-5-REDUNDANCY: Activating primary member 10.55.55.2:1000
```

Activating the backup member:

Example:

```
00:01:05: %XCONNECT-5-REDUNDANCY: Activating secondary member 10.55.55.3:1001
```

Step 4 **logging pseudowire status**

you can use the **logging pseudowire status** command to monitor the status of the pseudowire.

Example:

```
Device(config)# l2vpn
Device(config-l2vpn)# logging pseudowire status
```

Configuration Examples for L2VPN Pseudowire Redundancy

Each of the configuration examples refers to one of the following pseudowire classes:

- AToM (like-to-like) pseudowire class:

```
pseudowire-class mpls
 encapsulation mpls
```

- L2VPN IP interworking:

```
pseudowire-class mpls-ip
 encapsulation mpls
 interworking ip
```

Example L2VPN Pseudowire Redundancy and AToM (Like to Like)

The following example shows a High-Level Data Link Control (HDLC) attachment circuit xconnect with a backup pseudowire:

```
interface Serial4/0
 xconnect 10.55.55.2 4000 pw-class mpls
 backup peer 10.55.55.3 4001 pw-class mpls
```

The following example shows a Frame Relay attachment circuit xconnect with a backup pseudowire:

```
connect fr-fr-pw Serial6/0 225 l2transport
 xconnect 10.55.55.2 5225 pw-class mpls
 backup peer 10.55.55.3 5226 pw-class mpls
```

Example L2VPN Pseudowire Redundancy and L2VPN Interworking

The following example shows an Ethernet attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
interface Ethernet0/0
 xconnect 10.55.55.2 1000 pw-class mpls-ip
 backup peer 10.55.55.3 1001 pw-class mpls-ip
```

The following example shows an Ethernet VLAN attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
interface Ethernet1/0.1
 encapsulation dot1Q 200
 no ip directed-broadcast
```

```
 xconnect 10.55.55.2 5200 pw-class mpls-ip
 backup peer 10.55.55.3 5201 pw-class mpls-ip
```

The following example shows a Frame Relay attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
connect fr-ppp-pw Serial6/0 250 l2transport
 xconnect 10.55.55.2 8250 pw-class mpls-ip
 backup peer 10.55.55.3 8251 pw-class mpls-ip
```

The following example shows a PPP attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
interface Serial7/0
 encapsulation ppp
 xconnect 10.55.55.2 2175 pw-class mpls-ip
 backup peer 10.55.55.3 2176 pw-class mpls-ip
```

Example L2VPN Pseudowire Redundancy with Layer 2 Local Switching

The following example shows an Ethernet VLAN-VLAN local switching xconnect with a pseudowire backup for Ethernet segment E2/0.2. If the subinterface associated with E2/0.2 goes down, the backup pseudowire is activated:

```
connect vlan-vlan Ethernet1/0.2 Ethernet2/0.2
 backup peer 10.55.55.3 1101 pw-class mpls
```

The following example shows a Frame Relay-to-Frame Relay local switching connect with a pseudowire backup for Frame Relay segment S8/0 150. If data-link connection identifier (DLCI) 150 on S8/0 goes down, the backup pseudowire is activated:

```
connect fr-fr-ls Serial6/0 150 Serial8/0 150
 backup peer 10.55.55.3 7151 pw-class mpls
```

Example L2VPN Pseudowire Redundancy and Layer 2 Tunneling Protocol Version 3

The following example shows how to configure a backup peer for an xconnect session:

```
pseudowire-class 773
 encapsulation l2tpv3
 ip local interface GigabitEthernet0/0/0.773
!
pseudowire-class 774
 encapsulation l2tpv3
 ip local interface GigabitEthernet0/0/1.774
!
interface GigabitEthernet0/0/0.780
 encapsulation dot1Q 780
 xconnect 10.22.73.14 100 pw-class 773
  backup peer 10.22.74.14 101 pw-class 774
  backup delay 0 0
```

The following example shows how to configure a Gigabit Ethernet port with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/2
 xconnect 10.22.70.83 50 pw-class pe1-pw-primary
 backup peer 20.22.70.85 51 pw-class pe1-pw-secondary
```

The following example shows how to configure a Gigabit Ethernet VLAN with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/0.100
 encapsulation dot1q 100
 xconnect 10.22.70.83 60 pw-class pe1-pw-primary
 backup peer 10.22.70.85 61 pw-class pe1-pw-secondary
```

The following example shows how to configure a Gigabit Ethernet Q-in-Q with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/0.200
 encapsulation dot1q 200 second-dot1q 400
 xconnect 10.22.70.83 70 pw-class pe1-pw-primary
 backup peer 10.22.70.85 71 pw-class pe1-pw-secondary
```

The following example shows how to configure a Gigabit Ethernet Q-in-any with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/0.300
 encapsulation dot1q 300 second-dot1q any
 xconnect 10.22.70.83 80 pw-class pe1-pw-primary
 backup peer 10.22.70.85 81 pw-class pe1-pw-secondary
```

The following example shows how to configure an HDLC with L2VPN pseudowire redundancy and L2TPv3

```
interface Serial0/2/0:0
 no ip address
 xconnect 10.22.71.83 40 pw-class pe1-pw-hdlc
 backup peer 10.22.70.85 41 pw-class pe1-pw-hdlc-2
```

Configuration Examples for L2VPN Pseudowire Redundancy using the commands associated with the L2VPN Protocol-Based CLIs feature

Each of the configuration examples refers to one of the following interface pseudowires:

- AToM (like-to-like) interface pseudowire:

```
interface pseudowire 1
 encapsulation mpls
 neighbor 33.33.33.33 1
```

- L2VPN IP interworking:

```
interface pseudowire 1
 encapsulation mpls
```

```
 neighbor 33.33.33.33 1
 interworking ip
```

Example L2VPN Pseudowire Redundancy and AToM (Like to Like) using the commands associated with the L2VPN Protocol-Based CLIs feature

The following example shows a High-Level Data Link Control (HDLC) attachment circuit xconnect with a backup pseudowire:

```
interface Serial4/0
 interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.55.55.3 4001
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
```

The following example shows a Frame Relay attachment circuit xconnect with a backup pseudowire:

```
connect fr-fr-pw Serial6/0 225 l2transport
interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.55.55.3 5226
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
```

Example L2VPN Pseudowire Redundancy and L2VPN Interworking using the commands associated with the L2VPN Protocol-Based CLIs feature

The following example shows an Ethernet attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
interface Ethernet0/0
interface pseudowire 100
 source template type pseudowire ether-pw
!
l2vpn xconnect context con1
 member pseudowrire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows an Ethernet VLAN attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
interface Ethernet1/0.1
```

```
 encapsulation dot1Q 200
 no ip directed-broadcast
 interface pseudowire 100
 source template type pseudowire ether-pw
!
l2vpn xconnect context con1
 member pseudowrire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows a Frame Relay attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
connect fr-ppp-pw Serial6/0 250 l2transport
 interface pseudowire 100
 source template type pseudowire ether-pw
!
l2vpn xconnect context con1
 member pseudowrire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows a PPP attachment circuit xconnect with L2VPN IP interworking and a backup pseudowire:

```
interface Serial7/0
 encapsulation ppp
 interface pseudowire 100
 source template type pseudowire ether-pw
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

Example L2VPN Pseudowire Redundancy and Layer 2 Tunneling Protocol Version 3 using the commands associated with the L2VPN Protocol-Based CLIs feature

The following example shows how to configure a backup peer for an xconnect session:

```
interface pseudowire 773
 encapsulation l2tpv3
 ip local interface GigabitEthernet0/0/0.773
!
interface pseudowire 774
 encapsulation l2tpv3
 ip local interface GigabitEthernet0/0/1.774
!
interface GigabitEthernet0/0/0.780
 encapsulation dot1Q 780
 interface pseudowire 100
```

```
 source template type pseudowire ether-pw
 neighbor 10.22.73.14 100
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows how to configure a Gigabit Ethernet port with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/2
 interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.22.70.83 50
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows how to configure a Gigabit Ethernet VLAN with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/0.100
 encapsulation dot1q 100
 interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.22.70.83 60
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows how to configure a Gigabit Ethernet Q-in-Q with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/0.200
 encapsulation dot1q 200 second-dot1q 400
 interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.22.70.83 70
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows how to configure a Gigabit Ethernet Q-in-any with L2VPN pseudowire redundancy and L2TPv3:

```
interface GigabitEthernet0/0/0.300
```

```
 encapsulation dot1q 300 second-dot1q any
 interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.22.70.83 80
!
l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

The following example shows how to configure an HDLC with L2VPN pseudowire redundancy and L2TPv3

```
interface Serial0/2/0:0
 no ip address
 interface pseudowire 100
 source template type pseudowire ether-pw
 neighbor 10.22.71.83 40
!
l2vpn xconnect context con1
 l2vpn xconnect context con1
 member pseudowire 100 group GR_1 priority 1
 member pseudowire 1001 group GR_1 priority 2
 member GigabitEthernet0/0/2 service-instance 1
 redundancy delay 0 0 group GR_1
 interworking ip
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Wide-area networking commands	*Cisco IOS Wide-Area Networking Command Reference*
Cisco IOS XE Multiprotocol Label Switching configuration tasks	*Cisco IOS XE Multiprotocol Label Switching Configuration Guide*
Cisco IOS XE Wide-area networking configuration tasks	*Cisco IOS XE Wide-Area Networking Configuration Guide*

Standards

Standards	Title
None	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

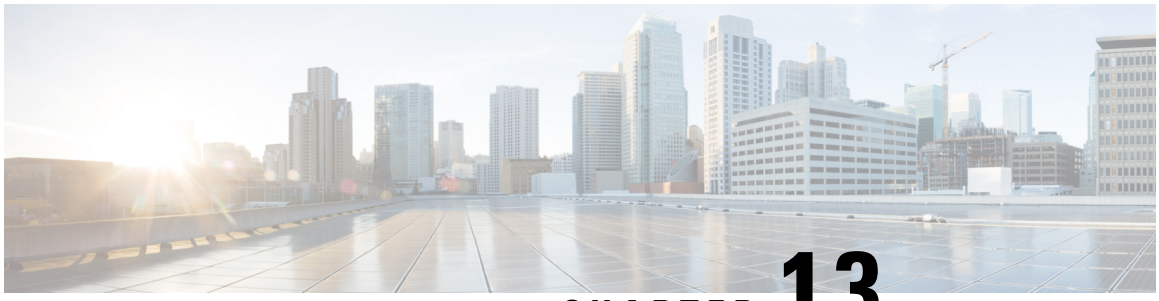
Feature Information for L2VPN Pseudowire Redundancy

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 26: Feature Information for L2VPN Pseudowire Redundancy

Feature Name	Releases	Feature Information
L2VPN Pseudowire Redundancy	XE 2.3 XE 3.3S	This feature enables you to set up your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. In Cisco IOS XE Release 2.3, this feature was integrated into the Cisco ASR 1000 Series Aggregation Service Routers. In Cisco IOS XE Release 3.3S, this feature supports Layer 2 Tunneling Protocol Version 3 (L2TPv3). The following commands were introduced or modified: **backup delay (L2VPN local switching), backup peer, show xconnect, xconnect backup force-switchover, xconnect logging redundancy.**
L2VPN Pseudowire Redundancies	Cisco IOS XE Fuji 16.9.1	In Cisco IOS XE Fuji 16.9.1, this feature is supported on Cisco 1000 Series ISRs.



CHAPTER 13

Layer 2 Local Switching

The Layer 2 Local Switching feature allows you to switch Layer 2 data in two ways:

- Between two interfaces on the same router
- Between two circuits on the same interface port, which is called same-port switching

The following interface-to-interface switching combinations are supported by this feature:

- ATM to ATM
- ATM to Ethernet
- Ethernet/Ethernet VLAN to Ethernet/Ethernet VLAN
- Frame Relay to Frame Relay

The following same-port switching features are supported:

- ATM Permanent Virtual Circuit (PVC) and Permanent Virtual Path (PVP)
- Ethernet VLAN
- Frame Relay

- Prerequisites for Layer 2 Local Switching, on page 255
- Restrictions for Layer 2 Local Switching, on page 256
- Information About Layer 2 Local Switching, on page 256
- How to Configure Layer 2 Local Switching, on page 257
- Configuration Examples for Layer 2 Local Switching, on page 267
- Additional References, on page 270
- Feature Information for Layer 2 Local Switching, on page 271

Prerequisites for Layer 2 Local Switching

You must enable Cisco Express Forwarding for the Cisco ASR 1000 Series Aggregation Services Router.

Restrictions for Layer 2 Local Switching

- For Ethernet/Ethernet VLAN circuits, the Cisco ASR 1000 Series Aggregation Services Router must have Ethernet Adapters.
- For Frame Relay local switching, you must globally issue the **frame-relay switching** command.

Information About Layer 2 Local Switching

Layer 2 Local Switching Overview

Local switching allows you to switch Layer 2 data between two interfaces of the same type (for example, Ethernet to Ethernet or Frame Relay to Frame Relay) or between interfaces of different types (for example, Ethernet VLAN to Ethernet VLAN or Ethernet to Ethernet VLAN) on the same router. The interfaces can be on the same line card or on two different cards. During these kinds of switching, the Layer 2 address is used, not the Layer 3 address.

Additionally, same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

NSF SSO—Local Switching Overview

Nonstop forwarding (NSF) and stateful switchover (SSO) improve the availability of the network by providing redundant Route Processors and checkpointing of data to ensure minimal packet loss when the primary Route Processor goes down. NSF/SSO support is available for the following locally switched attachment circuits:

- Ethernet/Ethernet VLAN to Ethernet/Ethernet VLAN
- Frame Relay to Frame Relay

Layer 2 Local Switching Applications

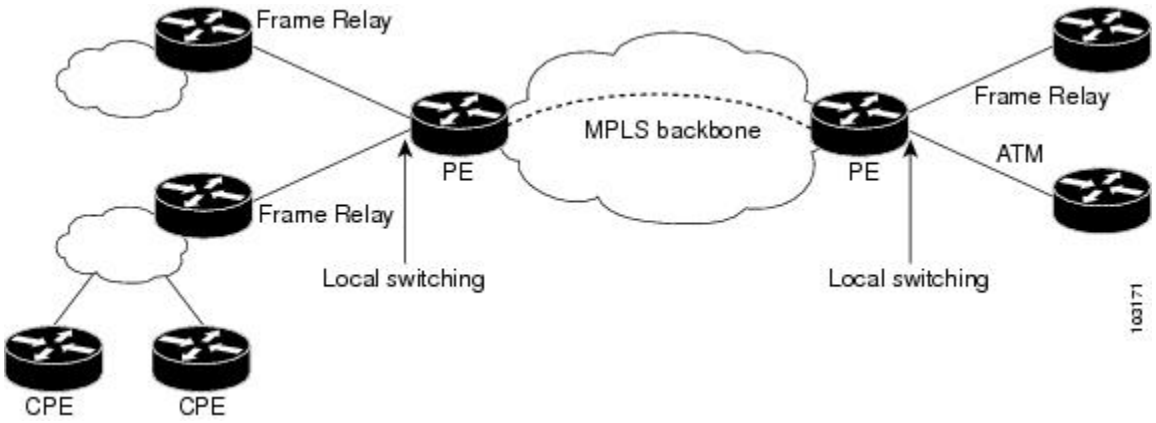
Incumbent local exchange carriers (ILECs) that use an interexchange carrier (IXC) to carry traffic between two local exchange carriers can use the Layer 2 Local Switching feature. Telecom regulations require the ILECs to pay the IXCs to carry that traffic. At times, the ILECs cannot terminate customer connections that are in different local access and transport areas (LATAs). In other cases, customer connections terminate in the same LATA, which may also be on the same router.

For example, company A has more than 50 LATAs across the country and uses three routers for each LATA. Company A uses companies B and C to carry traffic between local exchange carriers. Local switching of Layer 2 frames on the same router might be required.

Similarly, if a router is using, for example, a channelized interface, it might need to switch incoming and outgoing traffic across two logical interfaces that reside on a single physical port. The same-port local switching feature addresses that implementation.

The figure below shows a network that uses local switching for both Frame Relay to Frame Relay and ATM to Frame Relay local switching.

Figure 37: Local Switching Example



How to Configure Layer 2 Local Switching

Configuring Ethernet VLAN Same-Port Switching

Perform this task to configure Ethernet VLAN same-port switching.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface fastethernet** *slot* / *port* . *subinterface-number*
4. **encapsulation dot1q** *vlan-id*
5. **exit**
6. **interface fastethernet** *slot* / *port* . *subinterface-number*
7. **encapsulation dot1q** *vlan-id*
8. **exit**
9. **connect** *connection-name type number type number*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Router# configure terminal`	
Step 3	**interface fastethernet** *slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface fastethernet6/0.1`	Specifies the first Fast Ethernet line card, subslot (if available), port, and subinterface, and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `Router(config-subif)# encapsulation dot1q 10`	Enables the subinterface to accept 802.1Q VLAN packets and specifies the first VLAN.
Step 5	**exit** **Example:** `Router(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 6	**interface fastethernet** *slot* / *port* **.** *subinterface-number* **Example:** `Router(config)# interface fastethernet6/0.2`	Specifies the second Fast Ethernet line card, subslot (if available), port, and subinterface, and enters subinterface configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **Example:** `Router(config-subif)# encapsulation dot1q 20`	Enables the subinterface to accept 802.1Q VLAN packets and specifies the second VLAN.
Step 8	**exit** **Example:** `Router(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 9	**connect** *connection-name type number type number* **Example:** `Router(config)# connect conn fastethernet 6/0.1 fastethernet 6/0.2`	Creates a local connection between the two subinterfaces (and hence their previously specified VLANs) on the same Fast Ethernet port.

Configuring Ethernet Port Mode to Ethernet VLAN Local Switching

Perform this task to configure local switching for Ethernet (port mode) to Ethernet VLAN.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **interface fastethernet** *slot* / *subslot* / *port*
4. **interface fastethernet** *slot* / *port* / *subinterface-number*
5. **encapsulation dot1q** *vlan-id*
6. **exit**
7. **connect** *connection-name type number type number*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface fastethernet** *slot* / *subslot* / *port* **Example:** `Router(config)# interface fastethernet3/0/0`	Specifies a Fast Ethernet line card, subslot (if available), and port, and enters interface configuration mode. • This is the interface on one side of the PE router that passes Ethernet packets to and from the customer edge (CE) router.
Step 4	**interface fastethernet** *slot* / *port* / *subinterface-number* **Example:** `Router(config-if)# interface fastethernet6/0/0.1`	Specifies a Fast Ethernet line card, subslot (if available), port, and subinterface, and enters subinterface configuration mode. • This is the interface on the other side of the PE router than passes Ethernet VLAN packets to and from the CE router.
Step 5	**encapsulation dot1q** *vlan-id* **Example:** `Router(config-subif)# encapsulation dot1q 100`	Enables the interface to accept 802.1Q VLAN packets.
Step 6	**exit** **Example:** `Router(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 7	**connect** *connection-name type number type number* **Example:** `Router(config)# connect eth-ethvlan-con fastethernet 3/0/0 fastethernet 6/0/0.1`	Creates a local connection between the two interfaces.

Configuring ATM-to-ATM PVC Local Switching and Same-Port Switching

You can configure local switching for both ATM AAL5 and ATM AAL0 encapsulation types.

Creating the ATM PVC is not required. If you do not create a PVC, one is created for you. For ATM-to-ATM local switching, the autoprovisioned PVC is given the default encapsulation type AAL0 cell relay.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *slot* / *port*
4. **pvc** *vpi* / *vci* **l2transport**
5. **encapsulation** *layer-type*
6. **exit**
7. **exit**
8. **connect** *connection-name interface pvc interface pvc*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *slot* / *port* **Example:** `Router(config)# interface atm1/0/0`	Specifies an ATM line card, subslot (if available), and port, and enters interface configuration mode.
Step 4	**pvc** *vpi* / *vci* **l2transport** **Example:** `Router(config-if)# pvc 1/100 l2transport`	Assigns a VPI and VCI and enters ATM PVC l2transport configuration mode. • The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.

	Command or Action	Purpose
Step 5	**encapsulation** *layer-type* **Example:** `Router(cfg-if-atm-l2trans-pvc)# encapsulation aal5`	Specifies the encapsulation type for the ATM PVC. Both AAL0 and AAL5 are supported. • Repeat Steps 3 through 5 for another ATM PVC on the same router.
Step 6	**exit** **Example:** `Router(cfg-if-atm-l2trans-pvc)# exit`	Exits PVC l2transport configuration mode and returns to interface configuration mode.
Step 7	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**connect** *connection-name interface pvc interface pvc* **Example:** `Router(config)# connect atm-con atm1/0/0 1/100 atm2/0/0 1/100`	Creates a local connection between the two specified permanent virtual circuits.

Configuring ATM-to-ATM PVP Local Switching

Perform this task to configure ATM-to-ATM PVP local switching.

Starting with Cisco IOS Release 12.0(30)S, you can configure same-port switching, as detailed in the Configuring ATM PVP Same-Port Switching, on page 262.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *slot/port*
4. **atm pvp** *vpi* **l2transport**
5. **exit**
6. **exit**
7. **connect** *connection-name interface pvp interface pvp*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *slot/port* **Example:** `Router(config)# interface atm1/0`	Specifies an ATM line card, subslot (if available), and port and enters interface configuration mode.
Step 4	**atm pvp** *vpi* **l2transport** **Example:** `Router(config-if)# atm pvp 100 l2transport`	Identifies the virtual path and enters PVP l2transport configuration mode. The **l2transport**keyword indicates that the PVP is a switched PVP instead of a terminated PVP. • Repeat Steps 3 and 4 for another ATM permanent virtual path on the same router.
Step 5	**exit** **Example:** `Router(config-if-atm-l2trans-pvp)# exit`	Exits PVP l2transport configuration mode and returns to interface configuration mode.
Step 6	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**connect** *connection-name interface pvp interface pvp* **Example:** `Router(config)# connect atm-con atm1/0 100 atm2/0 200`	Creates a local connection between the two specified permanent virtual paths.

Configuring ATM PVP Same-Port Switching

Perform this task to configure ATM PVP switching on an ATM interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *slot/subslot/port*
4. **atm pvp** *vpi* **l2transport**
5. **exit**
6. **exit**
7. **connect** *connection-name interface pvp interface pvp*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *slot/subslot/port* **Example:** `Router(config)# interface atm1/0/0`	Specifies an ATM line card, subslot (if available), and port and enters interface configuration mode.
Step 4	**atm pvp** *vpi* **l2transport** **Example:** `Router(config-if)# atm pvp 100 l2transport`	Specifies one VPI and enters PVP l2transport configuration mode. Repeat this step for the other ATM permanent virtual path on this same port. • The **l2transport** keyword indicates that the indicated PVP is a switched PVP instead of a terminated PVP.
Step 5	**exit** **Example:** `Router(config-if-atm-l2trans-pvp)# exit`	Exits PVP l2transport configuration mode and returns to interface configuration mode.
Step 6	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**connect** *connection-name interface pvp interface pvp* **Example:** `Router(config)# connect atm-con atm1/0/0 100 atm1/0/0 200`	Creates the local connection between the two specified permanent virtual paths.

Configuring Frame Relay-to-Frame Relay Local Switching

For information about Frame Relay-to-Frame Relay local switching, see the Distributed Frame Relay Switching feature module.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip cef distributed**

4. **frame-relay switching**
5. **interface** *type number*
6. **encapsulation frame-relay** [**cisco** | **ietf**]
7. **frame-relay interface-dlci** *dlci* **switched**
8. **exit**
9. **exit**
10. **connect** *connection-name interface dlci interface dlci*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip cef distributed** **Example:** `Router(config)# ip cef distributed`	Enables Cisco Express Forwarding operation.
Step 4	**frame-relay switching** **Example:** `Router(config)# frame-relay switching`	Enables PVC switching on a Frame Relay DCE device or a Network-to-Network Interface (NNI).
Step 5	**interface** *type number* **Example:** `Router(config)# interface serial 0`	Specifies a Frame Relay interface and enters interface configuration mode.
Step 6	**encapsulation frame-relay** [**cisco**	**ietf**] **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation. • The default is **cisco** encapsulation. • You do not need to specify an encapsulation type.
Step 7	**frame-relay interface-dlci** *dlci* **switched** **Example:** `Router(config-if)# frame-relay interface-dlci 100 switched`	(Optional) Creates a switched PVC and enters Frame Relay DLCI configuration mode. • Repeat Steps 5 through 7 for each switched PVC. • If you do not create a Frame Relay PVC in this step, it will automatically be created by the **connect** command.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Router(config-fr-dlci)# exit`	Exits Frame Relay DLCI configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**connect** *connection-name interface dlci interface dlci* **Example:** `Router(config)# connect connection1 serial0 100 serial1 101`	Defines a connection between Frame Relay PVCs.

Verifying Layer 2 Local Switching

Verifying Layer 2 Local Switching Configuration

To verify configuration of the Layer 2 local switching feature, use the **show connection** command on the provider edge (PE) router.

SUMMARY STEPS

1. **show connection** [**all** | *element* | **id** *id* | **name** *name* | **port** *port*]

DETAILED STEPS

Procedure

show connection [**all** | *element* | **id** *id* | **name** *name* | **port** *port*]

The **show connection**command displays the local connection between a Gigabit Ethernet interface and another local Gigabit Ethernet interface:

Example:

```
Router# show connection name ethconn1
Connection: 1 - ethconn1
 Current State: UP
 Segment 1: GigabitEthernet0/0/0.1 up
 Segment 2: GigabitEthernet0/0/0.2 up
```

Verifying the NSF SSO Local Switching Configuration

Layer 2 local switching provides NSF/SSO support for Local Switching of the following attachment circuits on the same router:

- Ethernet/Ethernet VLAN to Ethernet/Ethernet VLAN

For information about configuring NSF/SSO on the Route Processors, see the " Stateful Switchover " module in the *Cisco IOS XE High Availability Configuration Guide* . Perform this task to verify that the NSF/SSO: Layer 2 Local Switching feature is working correctly.

SUMMARY STEPS

1. **ping**
2. **redundancy force-switchover**
3. **show connection all**
4. **ping**

DETAILED STEPS

Procedure

Step 1 **ping**

Issue the **ping**command or initiate traffic between the two CE routers.

Step 2 **redundancy force-switchover**

Force the switchover from the active RP to the standby RP by using the **redundancy force-switchover** command. This manual procedure allows for a "graceful" or controlled shutdown of the active RP and switchover to the standby RP. This graceful shutdown allows critical cleanup to occur.

Step 3 **show connection all**

Issue the **show connection all**command to ensure that the Layer 2 local switching connection on the dual RP is operating:

Example:

```
Router# show connection all
D   Name            Segment 1              Segment 2                  State
================================================================================
1    conn            Gi0/0/0.1              Gi0/0/0.2                  UP
```

Step 4 **ping**

Issue the **ping** command from the CE router to verify that the contiguous packet outage was minimal during the switchover.

Troubleshooting Tips

You can troubleshoot Layer 2 local switching using the following commands on the PE router:

- **debug conn**
- **show connection**

Configuration Examples for Layer 2 Local Switching

Example: Configuring Ethernet VLAN Same-Port Switching

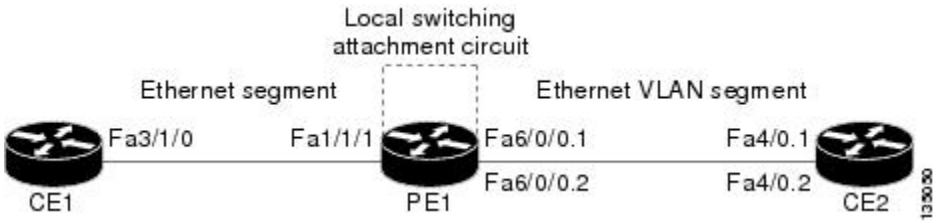
The following example shows same-port switching between two VLANs on one Ethernet interface:

```
interface fastethernet 0/0.1
 encapsulation dot1q 1
interface fastethernet 0/0.2
 encapsulation dot1q 2
connect conn FastEthernet 0/0.1 FastEthernet 0/0.2
```

Example: Configuring NSF SSO Ethernet Port Mode to Ethernet VLAN Local Switching

The following configuration uses the network topology shown in the figure below.

Figure 38: NSF/SSO: Layer 2 Local Switching: Ethernet to Ethernet VLAN



The following example shows the configuration of the CE interfaces to connect to the PE1 router:

CE1	CE2
ip routing ! interface fa3/1/0 description: connection to PE fa1/1/1 no shutdown ip address 10.1.1.1 255.255.255.0	ip routing ! interface fa4/0 no shutdown ! interface fa4/0.1 description: connection to PE1 fa6/0/0.1 encapsulation dot1Q 10 ip address 10.1.1.2 255.255.255.0 ! interface fa4/0.2 description - connection to PE1 fa6/0/0.2 encapsulation dot1Q 20 ip address 172.16.1.2 255.255.255.0

The following example shows the configuration of the PE1 router with NSF/SSO and the PE interfaces to the CE routers:

PE1

```
redundancy
 no keepalive-enable
 mode sso
!
!
ip routing
ip cef distributed
```

```
!
interface fa1/1/1
 description - connection to CE1 fa3/1/0
 no shutdown
 no ip address
!
!
interface fa6/0/0
 no shutdown
 no ip address
!
interface fa6/0/0.1
 description - connection to CE2 fa4/0.1
 encapsulation dot1Q 10
 no ip address
!
interface fa6/0/0.2
 description - connection to CE2 fa4/0.2
 encapsulation dot1Q 20
 no ip address
```

Example: ATM-to-ATM Local Switching

The following example shows local switching on ATM interfaces configured for AAL5:

```
interface atm1/0/0
 pvc 0/100 l2transport
 encapsulation aal5
interface atm2/0/0
 pvc 0/100 l2transport
```

```
 encapsulation aal5
connect aal5-conn atm1/0/0 0/100 atm2/0/0 0/100
```

Example: ATM PVC Same-Port Switching

The following example shows same-port switching between two PVCs on one ATM interface:

```
interface atm1/0/0
 pvc 0/100 l2transport
 encapsulation aal5
 pvc 0/200 l2transport
 encapsulation aal5
connect conn atm1/0/0 0/100 atm1/0/0 0/200
```

Example: ATM PVP Same-Port Switching

The following example shows same-port switching between two PVPs on one ATM interface:

```
interface atm1/0/0
 atm pvp 100 l2transport
 atm pvp 200 l2transport
connect conn atm1/0/0 100 atm1/0/0 200
```

Example: Configuring Frame Relay-to-Frame Relay Local Switching

The following example shows serial interfaces configured for Frame Relay. The **connect** command allows local switching between these two interfaces.

```
frame-relay switching
ip cef distributed
interface serial3/0/0
 encapsulation frame-relay
 frame-relay interface-dlci 100 switched
 frame-relay intf-type dce
interface serial3/1/0
 encapsulation frame-relay ietf
 frame-relay interface-dlci 200 switched
 frame-relay intf-type dce
connect fr-con serial3/0/0 100 serial3/1/0 200
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
WAN Commands	*Cisco IOS Wide-Area Networking Command Reference*
Stateful switchover configuration information	"Stateful Switchover " module in the *Cisco IOS XE High Availability Configuration Guide*

Standards

Standard	Title
draft-ietf-l2tpext-l2tp-base-03.txt	*Layer Two Tunneling Protocol (Version 3) 'L2TPv3'*
draft-martini-l2circuit-trans-mpls-09.txt	*Transport of Layer 2 Frames Over MPLS*
draft-martini-l2circuit-encap-mpls-04.txt	*Encapsulation Methods for Transport of Layer 2 Frames Over IP and MPLS Networks*
draft-ietf-ppvpn-l2vpn-00.txt	*An Architecture for L2VPNs*

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Layer 2 Local Switching

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 27: Feature Information for Layer 2 Local Switching

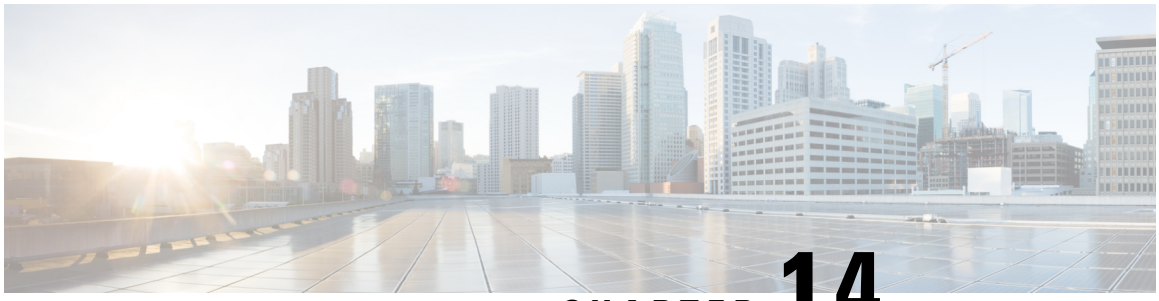
Feature Name	Releases	Feature Information
Layer 2 Local Switching	Cisco IOS XE Release 2.5	The Layer 2 Local Switching feature allows you to switch Layer 2 data between two interfaces on the same router, and in some cases to switch Layer 2 data between two circuits on the same interface port. In Cisco IOS XE Release 2.5, this feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. Support was added for the following local switching types: • Ethernet to Ethernet VLAN • Same-port switching for Ethernet VLAN The following commands were introduced or modified: **connect** (L2VPN local switching), **show connection**.
Layer 2 Local Switching - ATM to ATM	Cisco IOS XE Release 3.3S	In Cisco IOS XE Release 3.3S, this feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. The following commands were introduced or modified: **connect** (L2VPN local switching), **show connection**.
Layer 2 Local Switching - Frame Relay to Frame Relay	Cisco IOS XE Release 3.9S	In Cisco IOS XE Release 3.9S, this feature was introduced on the Cisco ISR 4400 Series Routers.



PART III

Frame Relay

- Configuring Frame Relay, on page 275
- Frame Relay Queueing and Fragmentation at the Interface, on page 305
- Frame Relay MIB Enhancements, on page 319
- Frame Relay PVC Interface Priority Queueing, on page 325
- ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 333
- Frame Relay show Command and debug Command Enhancements, on page 351
- L2VPN Local Switching—Frame Relay-Ethernet/VLAN , on page 355



CHAPTER 14

Configuring Frame Relay

Frame Relay is a high-performance Wide Area Network (WAN) protocol that operates at the physical and data link layers. The Cisco IOS XE Frame Relay implementation currently supports routing for IPv4, IPv6, and Multiprotocol Label Switching (MPLS).

- Restrictions for Configuring Frame Relay, on page 275
- Information About Frame Relay, on page 276
- How to Configure Frame Relay, on page 285
- Configuration Examples for Frame Relay, on page 296
- Additional References, on page 302
- Feature Information for Configuring Frame Relay, on page 303

Restrictions for Configuring Frame Relay

Cisco IOS XE software does not support the following:

- Multipoint permanent virtual circuits (PVCs)
- Switched virtual circuits (SVCs)
- Frame relay switching
- 4-byte extended addresses
- End-to-end keepalives
- FRF.9 payload compression
- Data stream compression
- Packet by packet encapsulation payload compression
- Multi-point frame-relay
- Legacy frame-relay traffic shaping (Cisco IOS XE software supports only policy map-based MQC.)
- MQC based frame relay traffic shaping is not supported on frame relay main interface.
- Function "set fr-de" for HQos configuration

Information About Frame Relay

Frame Relay Hardware Configurations

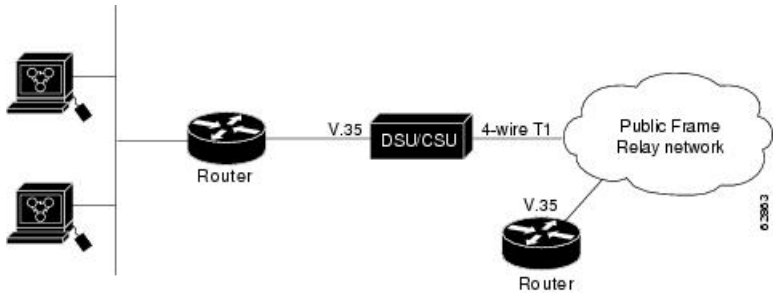
You can create Frame Relay connections using one of the following hardware configurations:

- Devices and access servers connected directly to the Frame Relay switch
- Devices and access servers connected directly to a channel service unit/digital service unit (CSU/DSU), which then connects to a remote Frame Relay switch

Note Devices can connect to Frame Relay networks either by direct connection to a Frame Relay switch, through a direct connection to a Point of sale (POS) interface or a T1/T3 interface, or through CSU/DSUs. However, a single device interface configured for Frame Relay can be configured for only one of these methods.

The CSU/DSU converts V.35 or RS-449 signals to the properly coded T1 transmission signal for successful reception by the Frame Relay network. The figure below illustrates the connections among the components.

Figure 39: Typical Frame Relay Configuration



The Frame Relay interface actually consists of one physical connection between the network server and the switch that provides the service. This single physical connection provides direct connectivity to each device on a network.

Frame Relay Encapsulation

Frame Relay supports encapsulation of all supported protocols in conformance with RFC 1490, *Multiprotocol Interconnect over Frame Relay*, allowing interoperability among multiple vendors. Use the IETF form of Frame Relay encapsulation if your device or access server is connected to another vendor's equipment across a Frame Relay network. IETF encapsulation is supported either at the interface level or on a per-VC basis.

Shut down the interface prior to changing encapsulation types. Although shutting down the interface is not required, it ensures that the interface is reset for the new encapsulation.

Dynamic or Static Address Mapping

Dynamic Address Mapping

Dynamic address mapping uses Frame Relay Inverse Address Resolution Protocol (ARP) to request the next-hop protocol address for a specific connection, given its known Data link connection identifier (DLCI). Responses to Inverse ARP requests are entered in an address-to-DLCI mapping table on the device or access server. The DLCI mapping table is then used to supply the next-hop protocol address or the DLCI for outgoing traffic.

Inverse ARP is enabled by default for all protocols it supports. However, it can be disabled for specific protocol-DLCI pairs. As a result, you can use dynamic mapping for some protocols and static mapping for other protocols on the same DLCI. You can explicitly disable Inverse ARP for a protocol-DLCI pair if you know that the protocol is not supported on the other end of the connection. For more information, see the Disabling or Reenabling Frame Relay Inverse ARP section.



Note Because Inverse ARP is enabled by default, no additional command is required to configure dynamic mapping on an interface and packets are not sent out for protocols that are not enabled on the interface.

Static Address Mapping

A static map links a specified next-hop protocol address to a specified Data link connection identifier (DLCI). Static mapping removes the need for Inverse Address Resolution Protocol (ARP) requests; when you supply a static map, Inverse ARP is automatically disabled for the specified protocol on the specified DLCI. You must use static mapping in the any of the following scenarios:

- If the device at the other end does not support Inverse ARP at all
- If the device does not support Inverse ARP for a specific protocol that you want to use over Frame Relay.

You can simplify the configuration for the Open Shortest Path First (OSPF) protocol by adding the optional **broadcast** keyword when doing this task. Refer to the **frame-relay map** command description in the *Cisco IOS Wide-Area Networking Command Reference* and the examples at the end of this chapter for more information about using the **broadcast** keyword.

LMI

The Cisco IOS XE software supports Local Management Interface (LMI) autosense, which enables the interface to determine the LMI type supported by the switch. Support for LMI autosense means that you need not configure the LMI explicitly.

LMI autosense is active in the following situations:

- The device is powered up or the interface changes state to up.
- The line protocol is down but the line is up.
- The interface is a Frame Relay Data Terminal Equipment (DTE).
- The LMI type is not explicitly configured.

Activating LMI Autosense

Status Request

When Local Management Interface (LMI) autosense is active, it sends out a full status request in all three LMI types to the switch. The order which is implemented in rapid succession is as follows:

- ANSI
- ITU
- Cisco

Cisco IOS XE software provides the ability to listen in on both DLCI 1023 (cisco LMI) and DLCI 0 (ANSI and ITU) simultaneously.

Status Messages

One or more of the status requests will prompts a reply (status message) from the switch. The device decodes the format of the reply and configures itself automatically. If more than one reply is received, the device configures itself with the type of the last received reply. This is to accommodate intelligent switches that can handle multiple formats simultaneously.

LMI Autosense

If Local Management Interface (LMI) autosense is unsuccessful, an intelligent retry scheme is built in. Every N391 interval (default is 60 seconds, which is 6 keep exchanges at 10 seconds each), LMI autosense attempts to ascertain the LMI type. For more information about N391, see the **frame-relay lmi-n391dte** command in the chapter "Frame Relay Commands " in the *Cisco IOS Wide-Area Networking Command Reference* .

The only visible indication to the user that LMI autosense is in progress is that **debug frame lmi** is enabled. At every N391 interval, the user sees 3 rapid status inquiries from the serial interface one in each of the following LMI-type:

- ANSI
- ITU
- Cisco

Configuration Options

No configuration options are provided; LMI autosense is transparent to the user. You can turn off LMI autosense by explicitly configuring an Local Management Interface (LMI) type. The LMI type must be written into NVRAM so that next time the device powers up, LMI autosense will be inactive. At the end of autoinstall, a **frame-relay lmi-type** *xxx* statement is included within the interface configuration. This configuration is not automatically written to NVRAM; you must explicitly write the configuration to NVRAM by using the **copy system:running-config** or **copy nvram:startup-config** command.

MQC-Based Frame Relay Traffic Shaping

Legacy frame-relay traffic shaping is not supported. Cisco IOS XE software only supports policy map based MQC.

Traffic-Shaping Map Class for the Interface

If you specify a Frame Relay map class for a main interface, all the virtual circuits (VCs) on its subinterfaces inherit all the traffic-shaping parameters defined for the class. You can override the default for a specific data link connection identifier (DLCI) on a specific subinterface by using the **class** VC configuration command to assign the DLCI explicitly to a different class. For information about setting up subinterfaces, refer the section Configuring Frame Relay Subinterfaces, on page 291 .

Specifying Map Class with Queueing and Traffic-Shaping Parameters

When defining a map class for Frame Relay, you can specify the average and peak rates (in bits per second) allowed on virtual circuits (VCs) associated with the map class. You can also specify *either* a custom queue list *or* a priority queue group to use on VCs associated with the map class.

Defining Access Lists

You can specify access lists and associate them with the custom queue list defined for any map class. The list number specified in the access list and the custom queue list tie them together. See the appropriate protocol chapters for information about defining access lists for the protocols you want to transmit on the Frame Relay network.

Understanding Frame Relay Subinterfaces

Frame Relay subinterfaces provide a mechanism for supporting partially meshed Frame Relay networks. Most protocols assume transitivity on a logical network; that is, if station A can communicate with station B, and station B can communicate to station C, then station A should be able to communicate to station C directly. Transitivity is true on LANs, but not on Frame Relay networks unless A is directly connected to C.

Additionally, certain protocols such as AppleTalk and transparent bridging are not supported on partially meshed networks because they require *split horizon* . Split horizon is a routing technique in which a packet received on an interface cannot be sent from the same interface even if received and transmitted on different virtual circuits (VCs) .

Configuring Frame Relay subinterfaces ensures that a single physical interface is considered as multiple virtual interfaces. Hence, packets received on one virtual interface can be forwarded to another virtual interface even if they are configured on the same physical interface.

Subinterfaces address the limitations of Frame Relay networks by providing an option to subdivide a partially meshed Frame Relay network into a number of smaller, fully meshed (or point-to-point) subnetworks. Each subnetwork is assigned its own network number and appears to the protocols as if it were reachable through a separate interface. (Note that point-to-point subinterfaces can be unnumbered for use with IP, thus reducing the addressing burden that might otherwise result.)

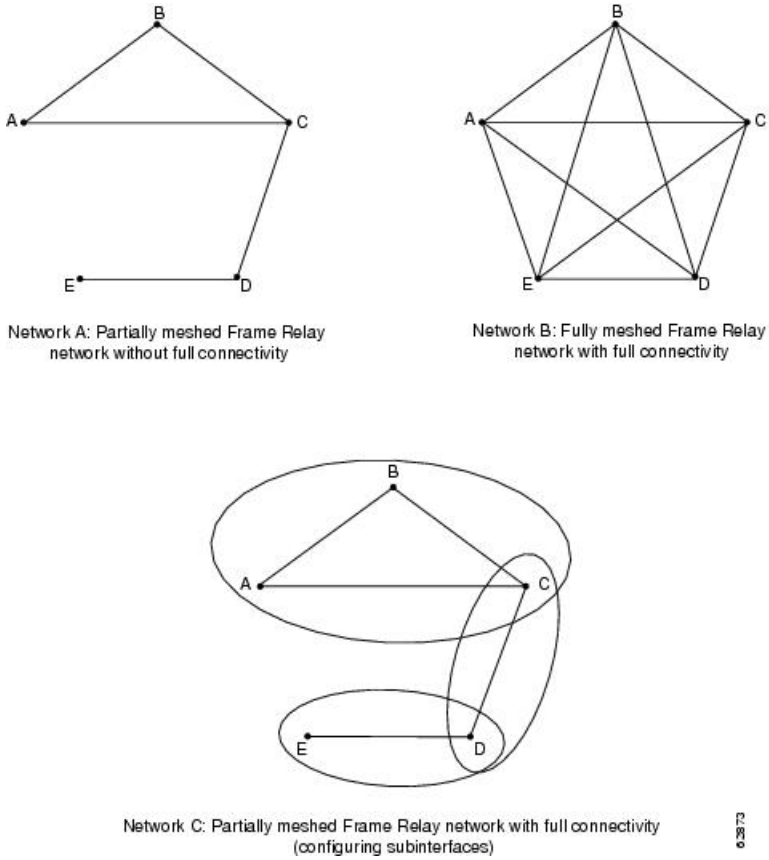


Note Cisco IOS XE software supports configuration of point-to-point subinterfaces.

The figure below shows a five-node Frame Relay network that is partially meshed (network A). If the entire network is viewed as a single subnetwork (with a single network number assigned), most protocols assume that node A can transmit a packet directly to node E, when, in fact it must be relayed through nodes C and D. This network can work with certain protocols (for example, IP). However, this network does not work with other protocols (for example, AppleTalk), because nodes C and D do not relay the packet out at the same interface on which it was received. To make this network fully functional, we need to created a fully meshed

network (network B). However, a fully meshed network requires a large number of permanent virtual circuits (PVCs), which may not be economically feasible.

Figure 40: Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network



By using subinterfaces, you can divide the Frame Relay network into 3 smaller subnetworks (network C) with separate network numbers. Nodes A, B, and C are connected to a fully meshed network, and nodes C and D, as well as nodes D and E, are connected via point-to-point networks. In this configuration, nodes C and D can access 2 subinterfaces and can therefore forward packets without violating split horizon rules. If transparent bridging is being used, each subinterface is viewed as a separate bridge port.

Subinterface Addressing

For point-to-point subinterfaces, the destination is presumed to be known and is identified or implied in the **frame-relay interface-dlci** command.

Note The **frame-relay interface-dlci** command is typically used on subinterfaces; however, it can also be applied to main interfaces. The command is used to enable routing protocols on main interfaces that are configured to use Inverse ARP. This command is also helpful for assigning a specific class to a single permanent virtual circuit (PVC) on a multipoint subinterface.

If you define a subinterface for point-to-point communication, you cannot reassign the same subinterface number to be used for multipoint communication without first rebooting the device or access server. Instead, you can simply avoid using that subinterface number and use a different subinterface number.

Backup Interface for a Subinterface

Both point-to-point and multipoint Frame Relay subinterfaces can be configured with a backup interface. This approach allows individual permanent virtual circuit (PVCs) to be backed up in case of failure rather than depending on the entire Frame Relay connection to fail before the backup takes over. You can configure a subinterface for backup on failure only, not for backup based on loading of the line.

If the main interface has a backup interface, it has a precedence over the backup interface of the subinterface in the case of complete loss of connectivity with the Frame Relay network. As a result, a subinterface backup is activated only in the following cases:

- If the main interface is up
- If the interface is down and does not have a backup interface defined

If a subinterface fails while its backup interface is in use, and the main interface goes down, the backup subinterface remains connected.

Disabling or Reenabling Frame Relay Inverse ARP

Frame Relay Inverse Address Resolution Protocol (ARP) is a method of building dynamic address mappings in Frame Relay networks that run DECnet, IP, and Novell IPX. Inverse ARP allows the device or access server to discover the protocol address of a device associated with the virtual circuit (VC).

Inverse ARP creates dynamic address mappings, as contrasted with the **frame-relay map** command, which defines static mappings between a specific protocol address and a specific data link connection identifier (DLCI).

Inverse ARP is enabled by default but can be disabled explicitly for a given protocol and DLCI pair. Disable or reenable Inverse ARP under the following conditions:

- Disable Inverse ARP for a selected protocol and DLCI pair when you know that the protocol is not supported at the other end of the connection.
- Reenable Inverse ARP for a protocol and DLCI pair if conditions or equipment change and the protocol is then supported at the other end of the connection.



Note If you change from a point-to-point subinterface to a multipoint subinterface, change the subinterface number. Frame Relay Inverse ARP will be on by default, and no further action is required.

You do not need to enable or disable Inverse ARP if you have a point-to-point interface.

Frame Relay Fragmentation

End-to-End FRF.12 Fragmentation

The purpose of end-to-end Frame Relay Fragmentation 12 (FRF.12) is to support real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data transmission. FRF.12

fragmentation is defined by the FRF.12 Implementation Agreement. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. In this way, real-time and non-real-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic.

End-to-end FRF.12 fragmentation is recommended for use on permanent virtual circuits (PVCs) that share links with other PVCs that are transporting voice and on PVCs transporting Voice over IP (VoIP). Although VoIP packets should not be fragmented, they can be interleaved with fragmented packets.

FRF.12 is configured on a per-PVC basis using a Frame Relay map class. The map class can be applied to one or many PVCs. Frame Relay traffic shaping must be enabled on the interface for fragmentation.

Note When Frame Relay fragmentation is configured, Weighted Fair Queuing (WFQ) or Low Latency Queuing (LLQ) is mandatory. If a map class is configured for Frame Relay fragmentation and the queuing type on that map class is not WFQ or LLQ, the configured queueing type is automatically overridden by WFQ with the default values. To configure LLQ for Frame Relay, refer to the *Cisco IOS XE Quality of Service Solutions Configuration Guide* .

Setting the Fragment Size

Set the fragment size so that voice packets are not fragmented and do not experience a serialization delay greater than 20 ms.

To set the fragment size, the link speed must be taken into account. The fragment size should be larger than the voice packets, but small enough to minimize latency on the voice packets. Turn on fragmentation for low speed links (less than 768 kbps).

Set the fragment size based on the lowest port speed between the routers. For example, if there is a hub and spoke Frame Relay topology where the hub has a T1 speed and the remote routers have 64 kbps port speeds, the fragment size needs to be set for the 64 kbps speed on both routers. Any other PVCs that share the same physical interface need to configure the fragmentation to the size used by the voice PVC.

If the lowest link speed in the path is 64 kbps, the recommended fragment size (for 10 ms serialization delay) is 80 bytes. If the lowest link speed is 128 kbps, the recommended fragment size is 160 bytes.

For more information, refer to the " Fragmentation (FRF.12)" section in the VoIP over Frame Relay with Quality of Service (Fragmentation, Traffic Shaping, LLQ / IP RTP Priority) document.

TCP IP Header Compression

TCP/IP header compression, as described by RFC 1144, *Compressing TCP/IP Headers for Low-Speed Serial Links* is designed to improve the efficiency of bandwidth utilization over low-speed serial links. A typical TCP/IP packet includes a 40-byte datagram header. Once a connection is established, the header information is redundant and need not be repeated in every packet that is sent. Reconstructing a smaller header that identifies the connection, indicates the fields that have changed and the amount of change reduces the number of bytes transmitted. The average compressed header is 10 bytes long.

For this algorithm to function, packets must arrive in order. If packets arrive out of order, the reconstruction will appear to create regular TCP/IP packets but the packets will not match the original. Because priority queueing changes the order in which packets are transmitted, enabling priority queueing on the interface is not recommended.

Note If you configure an interface with Cisco-proprietary encapsulation and TCP/IP header compression, Frame Relay IP maps inherit the compression characteristics of the interface. However, if you configure the interface with IETF encapsulation, the interface cannot be configured for compression. Frame Relay maps will have to be configured individually to support TCP/IP header compression.

Specifying an Individual IP Map for TCP IP Header Compression

Note An interface configured to support TCP/IP header compression does not also support priority queuing or custom queuing.

TCP/IP header compression requires Cisco-proprietary encapsulation. If you need to have IETF encapsulation on an interface as a whole, you can still configure a specific IP map to use Cisco-proprietary encapsulation and TCP header compression. In addition, if you configure the interface to perform TCP/IP header compression, you can still configure a specific IP map not to compress TCP/IP headers.

You can specify whether TCP/IP header compression is active or passive. Active compression subjects every outgoing packet to TCP/IP header compression. Passive compression subjects an outgoing TCP/IP packet to header compression only if a packet had a compressed TCP/IP header when it was received.

Specifying an Interface for TCP IP Header Compression

You can configure the interface with an active or passive TCP/IP header compression. Active compression, the default, subjects all outgoing TCP/IP packets to header compression. Passive compression subjects an outgoing packet to header compression only if the packet had a compressed TCP/IP header when it was received on that interface.

Note If an interface configured with Cisco-proprietary encapsulation is later configured with IETF encapsulation, all TCP/IP header compression characteristics are lost. To apply TCP/IP header compression over an interface configured with IETF encapsulation, you must configure individual IP maps, as described in the *Configuring an Individual IP Map for TCP IP Header Compression* section.

Real-Time Header Compression with Frame Relay Encapsulation

Real-time Transport Protocol (RTP) is a protocol used for carrying packetized audio and video traffic over an IP network. It provides end-to-end network transport functions intended for these real-time traffic applications and multicast or unicast network services. RTP is described in RFC 1889,*A Transport Protocol for Real-Time Applications* . RTP is not intended for data traffic, which uses TCP or UDP.

For configuration tasks and examples of RTP header compression using Frame Relay encapsulation, see the *Cisco IOS XE IP Multicast Configuration Guide* .

The commands for configuring this feature are available in the *Cisco IOS IP Multicast Command Reference.*

Discard Eligibility

Frame Relay packets can be set with low priority or low time sensitivity. These packets will be the first to be dropped when a Frame Relay switch is congested. The mechanism that allows a Frame Relay switch to identify such packets is the discard eligibility (DE) bit.

Discard eligibility requires the Frame Relay network to be able to interpret the DE bit. Some networks take no action when the DE bit is set, and others use the DE bit to determine which packets to discard. The best interpretation is to use the DE bit to determine which packets should be dropped first and also which packets have lower time sensitivity.

You can create DE lists that identify the characteristics of packets to be eligible for discarding, and you can also specify DE groups to identify the data link connection identifier (DLCI) that is affected.

You can create DE lists based on the protocol or the interface, and on characteristics such as fragmentation of the packet, a specific TCP or UDP port, an access list number, or a packet size.

DLCI Priority Levels

Data Link Connection Identifier (DLCI) priority levels allow you to separate different types of traffic and provides a traffic management tool for congestion problems caused by the following:

- Mixing batch and interactive traffic over the same DLCI
- Queuing traffic from sites with high-speed access to destination sites with lower-speed access

Before you configure the DLCI priority levels, you must:

- Enable Frame Relay encapsulation.
- Define dynamic or static address mapping.
- Ensure that you define each of the DLCIs to which you intend to apply levels. You can associate priority-level DLCIs with subinterfaces.
- Configure the LMI.



Note DLCI priority levels provide a way to define multiple parallel DLCIs for different types of traffic. DLCI priority levels do not assign priority queues within the device or access server. In fact, they are independent of the priority queues of the device. However, if you enable queuing and use the same DLCIs for queuing, then high-priority DLCIs can be put into high-priority queues.

How to Configure Frame Relay

Enabling Frame Relay Encapsulation on an Interface



Note Frame Relay encapsulation is a prerequisite for any Frame Relay commands on an interface.

To enable Frame Relay encapsulation on the interface level, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **encapsulation frame-relay**[**ietf**]
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *typenumber* **Example:** `Device(config)# int ethernet 0/1`	Specifies the interface, and enters interface configuration mode.
Step 4	**encapsulation frame-relay**[**ietf**] **Example:** `Device(config-if)# encapsulation frame-relay ietf`	Enables and specifies the Frame Relay encapsulation method.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring Static Address Mapping

To establish static mapping according to your network requirements, use the following command in interface configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **frame-relay map** *protocol protocol-address dlci* [**broadcast**] [**ietf**] [**cisco**]
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *typenumber* **Example:** `Device(config)# int ethernet 0/1`	Specifies the interface, and enters interface configuration mode.
Step 4	**frame-relay map** *protocol protocol-address dlci* [**broadcast**] [**ietf**] [**cisco**] **Example:** `Device(config-if)#`	Enables and specifies the Frame Relay encapsulation method.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Explicitly Configuring the LMI

Setting the LMI Type

If the device or access server is attached to a public data network (PDN), the LMI type must match the type used on the public network. Otherwise, the LMI type can be set to suit the requirements of your private Frame Relay network. You can set one of the following three types of LMIs on Cisco devices:

- ANSI T1.617 Annex D
- Cisco
- ITU-T Q.933 Annex A

To do so, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **frame-relay lmi-type** {**ansi** | **cisco** | **q933a**}
5. **end**
6. **copy nvram:startup-config** *destination*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *typenumber* **Example:**	Specifies the interface, and enters interface configuration mode.

	Command or Action	Purpose
	`Device(config)# int ethernet 0/1`	
Step 4	**frame-relay lmi-type** {**ansi** | **cisco** | **q933a**} **Example:** `Device(config-if)#`	Sets the LMI type.
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 6	**copy nvram:startup-config** *destination* **Example:** `Device#`	Writes the LMI type to NVRAM.

Setting the LMI Keepalive Interval

A keepalive interval must be set to configure the Local Management Interface (LMI). By default, this interval is 10 seconds. According to the LMI protocol, the keepalive interval must be less than the corresponding interval on the switch. To set the keepalive interval, use the following command in interface configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *typenumber*
4. **keepalive** *keepalive period*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *typenumber* **Example:** `Device(config)# int ethernet 0/1`	Specifies the interface, and enters interface configuration mode.
Step 4	**keepalive** *keepalive period* **Example:** `Device(config-if)# keepalive 300`	Sets the keepalive interval. • *keepalive period*- Valid range is from 0 to 32767. **Note** To disable keepalives on networks that do not utilize LMI, use the **no keepalive** command.
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Setting the LMI Polling and Timer Intervals

You can set various optional counters, intervals, and thresholds to fine-tune the operation of your Local Management Interface data terminal equipment (LMI DTE) and data communications equipment (DCE) devices. Set these attributes by using one or more of the following commands in interface configuration mode:

Command	Purpose
frame-relay lmi-n392dce *threshold*	Sets the DCE and Network-to-Network Interface (NNI) error threshold.
frame-relay lmi-n393dce *events*	Sets the DCE and NNI monitored events count.
frame-relay lmi-t392dce *seconds*	Sets the polling verification timer on a DCE or NNI interface.
frame-relay lmi-n391dte *keep-exchanges*	Sets a full status polling interval on a DTE or NNI interface.
frame-relay lmi-n392dte *threshold*	Sets the DTE or NNI error threshold.
frame-relay lmi-n393dte *events*	Sets the DTE and NNI monitored events count.

Configuring MQC-Based Frame Relay Traffic Shaping

Specifying a Traffic-Shaping Map Class for the Interface

To specify a map class for the specified interface, use the following command beginning in interface configuration mode:

SUMMARY STEPS

1. Router(config-if)# **frame-relay class** *map-class-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay class** *map-class-name*	Specifies a Frame Relay map class for the interface.

Defining a Map Class with Queueing and Traffic-Shaping Parameters

To define a map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class class-default**
5. **bandwidth** {*bandwidth-in-kbps* | **remaining** | **percent** }
6. **priority** [*bandwidth-in-kbps* | **level** | **percent**]
7. **shape average** {*rate-in-bps* |**percent**}
8. **shape adaptive** *rate-in-bps*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map testmap`	Specifies a policy map to define and enters policy map configuration mode.
Step 4	**class class-default** **Example:** `Device(config-pmap)# class class-default`	Specifies a system default class and enters policy-map class configuration .

	Command or Action	Purpose
Step 5	**bandwidth** {*bandwidth-in-kbps* | **remaining** | **percent** } **Example:** `Device(config-pmap-c)#  bandwidth 50`	Configures a minimum bandwidth guarantee for a class.
Step 6	**priority** [*bandwidth-in-kbps* | **level** | **percent**] **Example:** `Device(config-pmap-c)# priority 150`	Assigns priority to a class of traffic belonging to a policy map.
Step 7	**shape average** {*rate-in-bps* |**percent**} **Example:** `Device(config-pmap-c)# shape average 8000`	Shapes traffic to the indicated bit rate according to the algorithm specified.
Step 8	**shape adaptive** *rate-in-bps* **Example:** `Device(config-pmap-c)# shape adaptive 9000`	Shapes traffic to the indicated bit rate according to the algorithm specified.

Customizing Frame Relay for Your Network

Configuring Frame Relay Subinterfaces

Configuring Subinterfaces

Note Multipoint DLCI configurations are currently not supported. Cisco IOS XE software supports point-to-point connections.

To configure subinterfaces on a Frame Relay network, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface type** *number* **.** *subinterface-number* {**multipoint** | **point-to-point**}
2. Router(config-subif)# **encapsulation frame-relay**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **interface type** *number* **.** *subinterface-number* {**multipoint** | **point-to-point**}	Creates a point-to-point or multipoint subinterface. • Cisco IOS XE software only supports point-to-point subinterfaces.

	Command or Action	Purpose
Step 2	Router(config-subif)# **encapsulation frame-relay**	Configures Frame Relay encapsulation on the serial interface.

Defining Subinterface Addressing on Point-to-Point Subinterfaces

If you specified a point-to-point subinterface in the preceding procedure, use the following command in subinterface configuration mode:

SUMMARY STEPS

1. Router(config-subif)# **frame-relay interface-dlci** *dlci*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config-subif)# **frame-relay interface-dlci** *dlci*	Associates the selected point-to-point subinterface with a DLCI.

Configuring a Backup Interface for a Subinterface

To configure a backup interface for a Frame Relay subinterface, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface type** *number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config)# **interface type** *number* **.** *subinterface-number* **point-to-point**
4. Router(config-subif)# **frame-relay interface-dlci** *dlci*
5. Router(config-subif)# **backup interface type** *number*
6. Router(config-subif)# **backup delay** *enable-delay disable-delay*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **interface type** *number*	Specifies the interface.
Step 2	Router(config-if)# **encapsulation frame-relay**	Configures Frame Relay encapsulation.
Step 3	Router(config)# **interface type** *number* **.** *subinterface-number* **point-to-point**	Configures the subinterface.
Step 4	Router(config-subif)# **frame-relay interface-dlci** *dlci*	Specifies DLCI for the subinterface.

	Command or Action	Purpose
Step 5	Router(config-subif)# **backup interface type** *number*	Configures backup interface for the subinterface.
Step 6	Router(config-subif)# **backup delay** *enable-delay disable-delay*	Specifies backup enable and disable delay.

Disabling or Reenabling Frame Relay Inverse ARP

To select or disable Inverse ARP, use one of the following commands in interface configuration mode:

Command	Purpose
frame-relay inverse-arp *protocol dlci*	Enables Frame Relay Inverse ARP for a specific protocol and DLCI pair, only if it was previously disabled.
no frame relay inverse-arp *protocol dlci*	Disables Frame Relay Inverse ARP for a specific protocol and DLCI pair.

Configuring Frame Relay Fragmentation

Configuring End-to-End FRF.12 Fragmentation

To configure FRF.12 fragmentation in a Frame Relay map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay fragment** *fragment_size*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies a map class to define QoS values for a Frame Relay SVC or PVC. The map class can be applied to one or many PVCs.
Step 2	Router(config-map-class)# **frame-relay fragment** *fragment_size*	Configures Frame Relay fragmentation for the map class. The *fragment_size* argument defines the payload size of a fragment; it excludes the Frame Relay headers and any Frame Relay fragmentation header. The valid range is from 16 to 1600 bytes, and the default is 53.

Verifying the Configuration of End-to-End FRF.12 Fragmentation

To verify FRF.12 fragmentation, use one or more of the following EXEC commands:

Command	Purpose
show frame-relay fragment [**interface** *interface*] [*dlci*]	Displays Frame Relay fragmentation information.
show frame-relay pvc [**interface** *interface*] [*dlci*]	Displays statistics about PVCs for Frame Relay interfaces.

Configuring TCP IP Header Compression

Configuring an Individual IP Map for TCP IP Header Compression

To configure an IP map to use Cisco-proprietary encapsulation and TCP/IP header compression, use the following command in interface configuration mode:

Command	Purpose
frame-relay map ip *ip-address dlci* [**broadcast**] **tcp header-compression** [**active** | **passive**] [**connections** *number*]	Configures an IP map to use TCP/IP header compression. Cisco-proprietary encapsulation is enabled by default.

Configuring an Interface for TCP IP Header Compression

To apply TCP/IP header compression to an interface, you must use the following commands in interface configuration mode:

SUMMARY STEPS

1. Router(config-if)# **encapsulation frame-relay**
2. Router(config-if)# **frame-relay ip tcp header-compression** [**passive**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config-if)# **encapsulation frame-relay**	Configures Cisco-proprietary encapsulation on the interface.
Step 2	Router(config-if)# **frame-relay ip tcp header-compression** [**passive**]	Enables TCP/IP header compression.

Disabling TCP IP Header Compression

You can disable TCP/IP header compression by using either of two commands that have different effects, depending on whether Frame Relay IP maps have been explicitly configured for TCP/IP header compression or have inherited their compression characteristics from the interface.

Frame Relay IP maps that have explicitly configured TCP/IP header compression must also have TCP/IP header compression explicitly disabled.

To disable TCP/IP header compression, use one of the following commands in interface configuration mode:

Command	Purpose
no frame-relay ip tcp header-compression	Disables TCP/IP header compression on all Frame Relay IP maps that are not explicitly configured for TCP header compression.
frame-relay map ip *ip-address dlci* **nocompress**	Disables RTP and TCP/IP header compression on a specified Frame Relay IP map.

Configuring Discard Eligibility

Defining a DE List

To define a DE list specifying the packets that can be dropped when the Frame Relay switch is congested, use the following command in global configuration mode:

SUMMARY STEPS

1. Router(config)# **frame-relay de-list** *list-number* {**protocol** *protocol* | **interface** *type number*} *characteristic*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **frame-relay de-list** *list-number* {**protocol** *protocol* | **interface** *type number*} *characteristic*	Defines a DE list.

Defining a DE Group

To define a DE group specifying the DE list and DLCI affected, use the following command in interface configuration mode:

Command	Purpose
frame-relay de-group *group-number dlci*	Defines a DE group.

Configuring DLCI Priority Levels

To configure DLCI priority levels, use the following command in interface configuration mode:

Command	Purpose
frame-relay priority-dlci-group *group-number high-dlci medium-dlci normal-dlci low-dlci*	Enables multiple parallel DLCIs for different Frame Relay traffic types; associates and sets level of specified DLCIs with same group. **Note** If you do not explicitly specify a DLCI for each of the priority levels, the last DLCI specified in the command line is used as the value of the remaining arguments. At a minimum, you must configure the high-priority and the medium-priority DLCIs.

Monitoring and Maintaining the Frame Relay Connections

To monitor Frame Relay connections, use any of the following commands in EXEC mode:

Command	Purpose
clear frame-relay-inarp	Clears dynamically created Frame Relay maps, which are created by the use of Inverse ARP.
show interfaces serial *type number*	Displays information about Frame Relay DLCIs and the LMI.
show frame-relay lmi [*type number*]	Displays LMI statistics.
show frame-relay map	Displays the current Frame Relay map entries.
show frame-relay pvc [*type number* [*dlci*]]	Displays PVC statistics.
show frame-relay route	Displays configured static routes.
show frame-relay traffic	Displays Frame Relay traffic statistics.
show frame-relay lapf	Displays information about the status of LAPF.
show frame-relay svc maplist	Displays all the SVCs under a specified map list.

Configuration Examples for Frame Relay

Example IETF Encapsulation

Example IETF Encapsulation on the Interface

The following example sets IETF encapsulation at the interface level. The keyword **ietf** sets the default encapsulation method for all maps to IETF.

```
encapsulation frame-relay ietf
frame-relay map ip 131.108.123.2 48 broadcast
frame-relay map ip 131.108.123.3 49 broadcast
```

Example IETF Encapsulation on a Per-DLCI Basis

The following example configures IETF encapsulation on a per-DLCI basis. This configuration has the same result as the configuration in the first example.

```
encapsulation frame-relay
frame-relay map ip 131.108.123.2 48 broadcast ietf
frame-relay map ip 131.108.123.3 49 broadcast ietf
```

Example Static Address Mapping

Example Two Routers in Static Mode

The following example shows how to configure two routers for static mode:

Configuration for Router 1

```
interface serial0
 ip address 131.108.64.2 255.255.255.0
 encapsulation frame-relay
 keepalive 10
 frame-relay map ip 131.108.64.1 43
```

Configuration for Router 2

```
interface serial1
 ip address 131.108.64.1 255.255.255.0
 encapsulation frame-relay
 keepalive 10
 frame-relay map ip 131.108.64.2 43
```

Example Subinterface

Example Basic Subinterface

In the following example, subinterface 1 is configured as a point-to-point subnet and subinterface 2 is configured as a multipoint subnet.

```
interface serial 0
 encapsulation frame-relay
interface serial 0.1 point-to-point
 ip address 10.0.1.1 255.255.255.0
 frame-relay interface-dlci 42
!
interface serial 0.2 multipoint
 ip address 10.0.2.1 255.255.255.0
 frame-relay map ip 10.0.2.2 18
```

Example Frame Relay Traffic Shaping

Example Configuring Class-Based Weighted Fair Queueing

The following example provides a sample configuration for Class-Based Weighted Fair Queueing (CBWFQ) with FRTS:

```
class-map voice
 match ip dscp ef
policy-map llq
 class voice
  priority 32
policy-map shape-policy-map
 class class-default
  shape average 64000
  shape adaptive 32000
  service-policy llq
map-class frame-relay shape-map-class

service-policy output shape-policy-map
interface serial 0/0
encapsulation frame-relay
interface serial 0/0.1 point-to-point
ip address 192.168.1.1 255.255.255.0
frame-relay interface-dlci 100
class shape-map-class
```

Example Configuring Class-Based Weighted Fair Queueing with Fragmentation

The following example provides a sample configuration for CBWFQ and fragmentation with FRTS. This configuration example is exactly the same as the example shown in the Example Configuring Class-Based Weighted Fair Queueing section, with the addition of the **frame-relay fragment** command to configure fragmentation.

```
class-map voice
 match ip dscp ef
policy-map llq
 class voice
  priority 32
policy-map shape-policy-map
 class class-default
  shape average 64000
  shape adaptive 32000
  service-policy llq
map-class frame-relay shape-map-class
 frame-relay fragment 80
 service-policy output shape-policy-map
interface serial 0/0
 encapsulation frame-relay
interface serial 0/0.1 point-to-point
 ip address 192.168.1.1 255.255.255.0
 frame-relay interface-dlci 100
  class shape-map-class
```

Example Backward Compatibility

The following configuration provides backward compatibility and interoperability with versions not compliant with RFC 1490. The **ietf** keyword is used to generate RFC 1490 traffic. This configuration is possible because of the flexibility provided by separately defining each map entry.

```
encapsulation frame-relay
frame-relay map ip 131.108.123.2 48 broadcast ietf
! interoperability is provided by IETF encapsulation
frame-relay map ip 131.108.123.3 49 broadcast ietf
frame-relay map ip 131.108.123.7 58 broadcast
! this line allows the router to connect with a
! device running an older version of software
frame-relay map decnet 21.7 49 broadcast
```

Example Booting from a Network Server over Frame Relay

When booting from a TFTP server over Frame Relay, you cannot boot from a network server via a broadcast. You must boot from a specific TFTP host. Also, a **frame-relay map** command must exist for the host from which you will boot.

For example, if file "gs3-bfx" is to be booted from a host with IP address 131.108.126.2, the following commands would need to be in the configuration:

```
boot system gs3-bfx 131.108.126.2
!
interface Serial 0
 encapsulation frame-relay
 frame-relay map IP 131.108.126.2 100 broadcast
```

The **frame-relay map** command is used to map an IP address into a DLCI address. To boot over Frame Relay, you must explicitly give the address of the network server to boot from, and a **frame-relay map** entry must exist for that site. For example, if file "gs3-bfx.83-2.0" is to be booted from a host with IP address 131.108.126.111, the following commands must be in the configuration:

```
boot system gs3-bfx.83-2.0 131.108.13.111
!
interface Serial 1
 ip address 131.108.126.200 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 131.108.126.111 100 broadcast
```

In this case, 100 is the DLCI that can get to host 131.108.126.111.

The remote router must be configured with the following command:

```
frame-relay map ip 131.108.126.200 101 broadcast
```

This entry allows the remote router to return a boot image (from the network server) to the router booting over Frame Relay. Here, 101 is a DLCI of the router being booted.

Example Frame Relay Fragmentation Configuration

Example FRF.12 Fragmentation

The following example shows the configuration of pure end-to-end FRF.12 fragmentation and weighted fair queueing in the map class called "frag". The fragment payload size is set to 40 bytes. The "frag" map class is associated with DLCI 100 on serial interface 1.

```
router(config)#
interface serial 1

router(config-if)# frame-relay interface-dlci 100
router(config-fr-dlci)# class frag
router(config-fr-dlci)# exit
router(config)# map-class frame-relay frag
router(config-map-class)# frame-relay fragment 40
```

Example TCP IP Header Compression

Example IP Map with Inherited TCP IP Header Compression

Note Shut down the interface or subinterface prior to adding or changing compression techniques. Although shutdown is not required, shutting down the interface ensures that it is reset for the new data structures.

The following example shows an interface configured for TCP/IP header compression and an IP map that inherits the compression characteristics. Note that the Frame Relay IP map is not explicitly configured for header compression.

```
interface serial 1
 encapsulation frame-relay
 ip address 131.108.177.178 255.255.255.0
 frame-relay map ip 131.108.177.177 177 broadcast
 frame-relay ip tcp header-compression passive
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics; the IP map has inherited passive TCP/IP header compression:

```
Router> show frame-relay map
Serial 1   (administratively down): ip 131.108.177.177
           dlci 177 (0xB1,0x2C10), static,
           broadcast,
           CISCO
           TCP/IP Header Compression (inherited), passive (inherited)
```

This example also applies to dynamic mappings achieved with the use of Inverse ARP on point-to-point subinterfaces where no Frame Relay maps are configured.

Example Using an IP Map to Override TCP IP Header Compression

The following example shows the use of a Frame Relay IP map to override the compression set on the interface:

```
interface serial 1
 encapsulation frame-relay
 ip address 131.108.177.178 255.255.255.0
 frame-relay map ip 131.108.177.177 177 broadcast nocompress
 frame-relay ip tcp header-compression passive
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics; the IP map has not inherited TCP header compression:

```
Router> show frame-relay map
Serial 1   (administratively down): ip 131.108.177.177
           dlci 177 (0xB1,0x2C10), static,
           broadcast,
           CISCO
```

Example Disabling Inherited TCP IP Header Compression

In this example, following is the initial configuration:

```
interface serial 1
 encapsulation frame-relay
 ip address 131.108.177.179 255.255.255.0
 frame-relay ip tcp header-compression passive
```

```
 frame-relay map ip 131.108.177.177 177 broadcast
 frame-relay map ip 131.108.177.178 178 broadcast tcp header-compression
```

Enter the following commands to enable inherited TCP/IP header compression:

```
serial interface 1
 no frame-relay ip tcp header-compression
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics:

```
Router> show frame-relay map
Serial 1   (administratively down): ip 131.108.177.177 177
           dlci 177(0xB1, 0x2C10), static,
           broadcast
           CISCO
Serial 1   (administratively down): ip 131.108.177.178 178
           dlci 178(0xB2,0x2C20), static
           broadcast
           CISCO
           TCP/IP Header Compression (enabled)
```

As a result, header compression is disabled for the first map (with DLCI 177), which inherited its header compression characteristics from the interface. However, header compression is not disabled for the second map (DLCI 178), which is explicitly configured for header compression.

Example Disabling Explicit TCP IP Header Compression

In this example, the initial configuration is the same as in the preceding example, but you must enter the following set of commands to enable explicit TCP/IP header compression:

```
serial interface 1
 no frame-relay ip tcp header-compression
 frame-relay map ip 131.108.177.178 178 nocompress
```

Use of the **show frame-relay map** command will display the resulting compression and encapsulation characteristics:

```
Router> show frame-relay map
Serial 1   (administratively down): ip 131.108.177.177 177
           dlci 177(0xB1,0x2C10), static,
           broadcast
           CISCO
Serial 1   (administratively down): ip 131.108.177.178 178
           dlci 178(0xB2,0x2C20), static
           broadcast
           CISCO
```

The result of the commands is to disable header compression for the first map (with DLCI 177), which inherited its header compression characteristics from the interface, and also explicitly to disable header compression for the second map (with DLCI 178), which was explicitly configured for header compression.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS XE Wide-Area Networking configuration tasks	*Cisco IOS XE Wide-Area Networking Configuration Guide*, Release 2
Wide-Area networking commands	*Cisco IOS Wide-Area Networking Command Reference*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

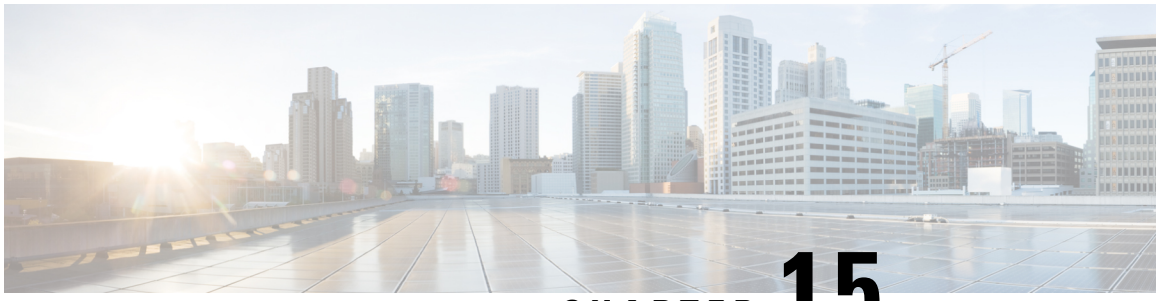
Feature Information for Configuring Frame Relay

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 28: Feature Information for Configuring Frame Relay

Feature Name	Releases	Feature Information
Frame Relay	Cisco IOS XE Release 2.1	Frame Relay is a high-performance WAN protocol that operates at the physical and data link layers.
Frame Relay Encapsulation	Cisco IOS XE Release 2.1	Frame Relay supports encapsulation of all supported protocols in conformance with RFC 1490, allowing interoperability between multiple vendors.
Frame Relay Fragmentation (FRF.12)	Cisco IOS XE Release 2.1	End-to-end FRF.12 fragmentation supports real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data. FRF.12 fragmentation is defined by the FRF.12 Implementation Agreement.
Local Management Interface	Cisco IOS XE Release 2.1	Local Management Interface (LMI) autosense enables an interface to determine the LMI type supported by a switch. With the support for LMI autosense, you do not need to configure the LMI explicitly.



CHAPTER 15

Frame Relay Queueing and Fragmentation at the Interface

The Frame Relay Queueing and Fragmentation at the Interface feature introduces support for low-latency queueing (LLQ) and FRF.12 end-to-end fragmentation on a Frame Relay interface.

- Restrictions for Frame Relay Queueing and Fragmentation at the Interface, on page 305
- Information About Frame Relay Queueing and Fragmentation at the Interface, on page 305
- How to Configure Frame Relay Queueing and Fragmentation at the Interface, on page 307
- Configuration Examples for Frame Relay Queueing and Fragmentation at the Interface, on page 315
- Additional References, on page 316
- Feature Information for Frame Relay Queueing and Fragmentation at the Interface, on page 317

Restrictions for Frame Relay Queueing and Fragmentation at the Interface

- Interface fragmentation and Frame Relay traffic shaping cannot be configured at the same time.
- Interface fragmentation and class-based fragmentation cannot be configured at the same time.
- Frame Relay switched virtual circuits (SVCs) are not supported.
- Hierarchical shaping and multiple shapers are not supported.

Information About Frame Relay Queueing and Fragmentation at the Interface

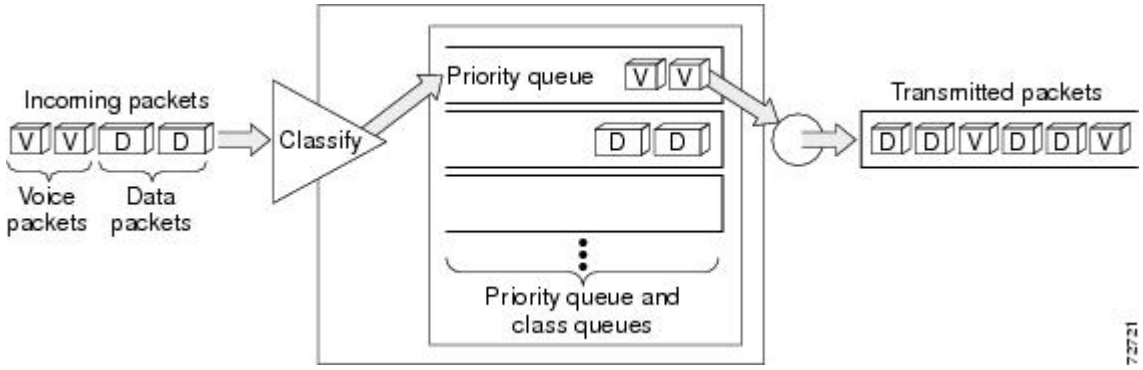
The Frame Relay Queueing and Fragmentation at the Interface feature simplifies the configuration of low-latency, low-jitter quality of service (QoS) by enabling the queueing policy and fragmentation configured on the main interface to apply to all permanent virtual circuits (PVCs) and subinterfaces under that interface. Before the introduction of this feature, queueing and fragmentation had to be configured on each individual PVC. Subrate shaping can also be configured on the interface.

How Frame Relay Queueing and Fragmentation at the Interface Works

When FRF.12 end-to-end fragmentation is enabled on an interface, all PVCs on the main interface and its subinterfaces will have fragmentation enabled with the same configured fragment size. To maintain low latency and low jitter for high-priority traffic, the configured fragment size must be greater than the largest high-priority frames. This configuration will prevent high-priority traffic from being fragmented and queued behind lower-priority fragmented frames. If the size of a high-priority frame is larger than the configured fragment size, the high-priority frame will be fragmented. Local Management Interface (LMI) traffic will not be fragmented and is guaranteed its required bandwidth.

When a low-latency queueing policy map is applied to the interface, traffic through the interface is identified using class maps and is directed to the appropriate queue. Time-sensitive traffic such as voice should be classified as high priority and will be queued on the priority queue. Traffic that does not fall into one of the defined classes will be queued on the class-default queue. Frames from the priority queue and class queues are subject to fragmentation and interleaving. As long as the configured fragment size is larger than the high-priority frames, the priority queue traffic will not be fragmented and will be interleaved with fragmented frames from other class queues. This approach provides the highest QoS transmission for priority queue traffic. The figure below illustrates the interface queueing and fragmentation process.

Figure 41: Frame Relay Queueing and Fragmentation at the Interface



Subrate shaping can also be applied to the interface, but interleaving of high-priority frames will not work when shaping is configured. If shaping is not configured, each PVC will be allowed to send bursts of traffic up to the physical line rate.

When shaping is configured and traffic exceeds the rate at which the shaper can send frames, the traffic is queued at the shaping layer using fair queueing. After a frame passes through the shaper, the frame is queued at the interface using whatever queueing method is configured. If shaping is not configured, then queueing occurs only at the interface.

Note For interleaving to work, both fragmentation and the low-latency queueing policy must be configured with shaping disabled.

The Frame Relay Queueing and Fragmentation at the Interface feature supports the following functionality:

- Voice over Frame Relay
- Weighted Random Early Detection
- Frame Relay payload compression

Note When payload compression and Frame Relay fragmentation are used at the same time, payload compression is always performed before fragmentation.

- IP header compression

Benefits of Frame Relay Queueing and Fragmentation at the Interface

Simple Configuration

The Frame Relay Queueing and Fragmentation at the Interface feature allows fragmentation, low-latency queueing, and subrate shaping to be configured on a Frame Relay interface queue. The fragmentation and queueing and shaping policy will apply to all PVCs and subinterfaces under the main interface, eliminating the need to configure QoS on each PVC individually.

Flexible Bandwidth

This feature allows PVCs to preserve the logical separation of traffic from different services while reducing bandwidth partitioning between PVCs. Each PVC can send bursts of traffic up to the interface shaping rate or, if shaping is not configured, the physical interface line rate.

How to Configure Frame Relay Queueing and Fragmentation at the Interface

Configuring Class Policy for the Priority Queue

To configure a policy map for the priority class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map*
4. **class** *class-name*
5. Router(config-pmap-c)# **priority** *bandwidth-kbps*
6. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map* **Example:** `Router(config) policy-map policy1`	Specifies the name of the policy map to be created or modified. • Use this command to define the queueing policy for the priority queue.
Step 4	**class** *class-name* **Example:** `Router(config-pmap)# class c1`	Specifies the name of a class to be created and included in the service policy. • The class name that you specify in the policy map defines the characteristics for that class and its match criteria as configured using the **class-map** command.
Step 5	Router(config-pmap-c)# **priority** *bandwidth-kbps* **Example:** `Router(config-pmap-c)# priority 30`	Creates a strict priority class and specifies the amount of bandwidth, in kbps, to be assigned to the class.
Step 6	**exit** **Example:** `Router(config-pmap-c)# exit`	Exits the current configuration mode.

Configuring Class Policy for the Bandwidth Queues

To configure a policy map and create class policies that make up the service policy, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map*
4. **class** *class-name*
5. Router(config-pmap-c)# **bandwidth** *bandwidth-kbps*
6. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map* **Example:** `Router(config)# policy-map policy1`	Specifies the name of the policy map to be created or modified. • Use this command to define the queueing policy for the priority queue. • The bandwidth queues and the priority queue use the same policy map.
Step 4	**class** *class-name* **Example:** `Router(config-pmap)# class c1`	Specifies the name of a class to be created and included in the service policy. • The class name that you specify in the policy map defines the characteristics for that class and its match criteria as configured using the **class-map**command.
Step 5	Router(config-pmap-c)# **bandwidth** *bandwidth-kbps* **Example:** `Router(config-pmap-c)# bandwidth 10`	Specifies the amount of bandwidth to be assigned to the class, in kbps, or as a percentage of the available bandwidth. Bandwidth must be specified in kbps or as a percentage consistently across classes. (Bandwidth of the priority queue must be specified in kbps.) • The sum of all bandwidth allocation on an interface cannot exceed 75 percent of the total available interface bandwidth. However, if you need to configure more than 75 percent of the interface bandwidth to classes, you can override the 75 percent maximum by using the **max-reserved-bandwidth** command.
Step 6	**exit** **Example:** `Router(config-pmap-c)# exit`	Exits the current configuration mode.

Configuring the Shaping Policy Using the Class-Default Class

In general, the class-default class is used to classify traffic that does not fall into one of the defined classes. Even though the class-default class is predefined when you create the policy map, you still have to configure it. If a default class is not configured, traffic that does not match any of the configured classes is given best-effort treatment, which means that the network will deliver the traffic if it can, without any assurance of reliability, delay prevention, or throughput.

If you configure shaping in addition to queueing on the interface, use the class-default class to configure the shaping policy. The shaping policy will serve as the parent in a hierarchical traffic policy. The queueing policy will serve as the child policy. The class-default class is used for the shaping policy so that all traffic for the entire interface is shaped and a bandwidth-limited stream can be created.

To configure the shaping policy in the class-default class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map*
4. **class class-default**
5. **shape** [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]]
6. **service-policy** *policy-map-name*
7. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map* **Example:** `Router(config)# policy-map policy1`	Specifies the name of the policy map to be created or modified. • Use this command to define the shaping policy.
Step 4	**class class-default** **Example:** `Router(config-pmap)# class class-default`	Specifies the default class so that you can configure or modify its policy.

	Command or Action	Purpose
Step 5	**shape** [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]] **Example:** `Router(config-pmap-c)# shape peak 10`	(Optional) Shapes traffic to the indicated bit rate according to the algorithm specified.
Step 6	**service-policy** *policy-map-name* **Example:** `Router(config-pmap-c)# service-policy policy1`	Specifies the name of a policy map to be used as a matching criterion (for nesting traffic policies [hierarchical traffic policies] within one another). • Use this command to attach the policy map for the priority queue (the child policy) to the shaping policy (the parent policy).
Step 7	**exit** **Example:** `Router(config-pmap-c)# exit`	Exits the current configuration mode.

Configuring Queueing and Fragmentation on the Frame Relay Interface

To configure low-latency queueing and FRF.12 end-to-end fragmentation on a Frame Relay interface, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface** *type number*
2. Router(config-if)# **encapsulation frame-relay**
3. Router(config-if)# **frame-relay interface-dlci** *dlci*
4. Router(config-if-dlci)# **class** *name*
5. Router(config-if-dlci)# **exit**
6. Router(config)# **map-class frame-relay** *name*
7. Router(config-map-class)# **frame-relay fragment** *fragment-size* **end-to-end**
8. Router(config-map-class)# **no frame-relay adaptive-shaping**
9. Router(config-map-class)# **service-policy output** *policy-map-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number*	Configures an interface type and enters interface configuration mode.
Step 2	Router(config-if)# **encapsulation frame-relay**	Enables Frame Relay encapsulation.

	Command or Action	Purpose
Step 3	Router(config-if)# **frame-relay interface-dlci** *dlci*	Assigns a DLCI to a specified Frame Relay subinterface on the router.
Step 4	Router(config-if-dlci)# **class** *name*	Associates a map class with a specified DLCI.
Step 5	Router(config-if-dlci)# **exit**	Exits configuration mode.
Step 6	Router(config)# **map-class frame-relay** *name*	Specifies a map class to define QoS values for a Frame Relay SVC or PVC.
Step 7	Router(config-map-class)# **frame-relay fragment** *fragment-size* **end-to-end**	Enables fragmentation of Frame Relay frames. • To maintain low latency and low jitter for priority queue traffic, configure the fragment size to be greater than the largest high-priority frame that would be expected.
Step 8	Router(config-map-class)# **no frame-relay adaptive-shaping**	Disables Frame Relay adaptive traffic shaping.
Step 9	Router(config-map-class)# **service-policy output** *policy-map-name*	Attaches a policy map to an output interface, to be used as the service policy for that interface. • If shaping is being used, use this command to attach the shaping policy (which includes the nested queueing policy) to the interface. • Interleaving of high-priority frames will not work if shaping is configured on the interface. • If shaping is not being used, use this command to attach the queueing policy to the interface.

Verifying Frame Relay Queueing and Fragmentation at the Interface

To verify the configuration and performance of Frame Relay queueing and fragmentation at the interface, perform the following steps:

SUMMARY STEPS

1. Enter the **show running-config** command to verify the configuration.
2. Enter the **show policy-map interface** command to display low-latency queueing information, packet counters, and statistics for the policy map applied to the interface. Compare the values in the "packets" and the "pkts matched" counters; under normal circumstances, the "packets" counter is much larger than the "pkts matched" counter. If the values of the two counters are nearly equal, then the interface is receiving a large number of process-switched packets or is heavily congested.
3. Enter the **show interfaces serial**command to display information about the queueing strategy, priority queue interleaving, and type of fragmentation configured on the interface. You can determine whether the interface has reached a congestion condition and packets have been queued by looking at the "Conversations" fields. A nonzero value for "max active" counter shows whether any queues have been

active. If the "active" counter is a nonzero value, you can use the **show queue** command to view the contents of the queues.

DETAILED STEPS

Procedure

Step 1 Enter the **show running-config** command to verify the configuration.

Example:

```
Router# show running-config
Building configuration...
.
.
.
class-map match-all voice
  match ip precedence 5
!
!policy-map llq
  class voice
    priority 64
policy-map shaper
  class class-default
   shape peak 96000
   service-policy llq
!
!interface Serial1/1
 ip address 16.0.0.1 255.255.255.0
 encapsulation frame-relay
 service-policy output shaper
 frame-relay fragment 80 end-to-end
!
```

Step 2 Enter the **show policy-map interface** command to display low-latency queueing information, packet counters, and statistics for the policy map applied to the interface. Compare the values in the "packets" and the "pkts matched" counters; under normal circumstances, the "packets" counter is much larger than the "pkts matched" counter. If the values of the two counters are nearly equal, then the interface is receiving a large number of process-switched packets or is heavily congested.

The following sample output for the **show policy-map interface command** is based on the configuration in Step 1:

Example:

```
Router# show policy-map interface serial 1/1
 Serial1/1
  Service-policy output:shaper
    Class-map:class-default (match-any)
      12617 packets, 1321846 bytes
      5 minute offered rate 33000 bps, drop rate 0 bps
      Match:any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           192000/96000     1992   7968      7968      83        1992
        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         12586     1321540   0         0         no
```

```
Service-policy :llq
  Class-map:voice (match-all)
    3146 packets, 283140 bytes
    5 minute offered rate 7000 bps, drop rate 0 bps
    Match:ip precedence 1
    Weighted Fair Queueing
      Strict Priority
      Output Queue:Conversation 24
      Bandwidth 64 (kbps) Burst 1600 (Bytes)
      (pkts matched/bytes matched) 0/0
      (total drops/bytes drops) 0/0
  Class-map:class-default (match-any)
    9471 packets, 1038706 bytes
    5 minute offered rate 26000 bps
    Match:any
```

Step 3 Enter the **show interfaces serial**command to display information about the queueing strategy, priority queue interleaving, and type of fragmentation configured on the interface. You can determine whether the interface has reached a congestion condition and packets have been queued by looking at the "Conversations" fields. A nonzero value for "max active" counter shows whether any queues have been active. If the "active" counter is a nonzero value, you can use the **show queue** command to view the contents of the queues.

The following sample output for the **show interfaces serial**command is based on the configuration in Step 1:

Example:

```
Router# show interfaces serial 1/1
Serial1/1 is up, line protocol is up
  Hardware is M4T
  Internet address is 16.0.0.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 5/255, rxload 1/255
  Encapsulation FRAME-RELAY, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  LMI enq sent  40, LMI stat recvd 40, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Fragmentation type:end-to-end, size 80, PQ interleaves 0
  Broadcast queue 0/64, broadcasts sent/dropped 0/0, interface broadcasts 0
  Last input 00:00:03, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:06:34
  Input queue:0/75/0/0 (size/max/drops/flushes); Total output drops:0
  Queueing strategy:weighted fair
  Output queue:0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/1/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 33000 bits/sec, 40 packets/sec
     40 packets input, 576 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     15929 packets output, 1668870 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Monitoring and Maintaining Frame Relay Queueing and Fragmentation at the Interface

To monitor and maintain Frame Relay queueing and fragmentation at the interface, use the following commands in privileged EXEC mode:

Command	Purpose
`Router# debug frame-relay fragment [event | interface type number dlci]`	Displays information related to Frame Relay fragmentation on a PVC.
`Router# show frame-relay fragment [interface type number [dlci]]`	Displays information about Frame Relay fragmentation.
`Router# show interfaces serial number`	Displays information about a serial interface.
`Router# show queue interface-type interface-number`	Displays the contents of packets inside a queue for a particular interface.
`Router# show policy-map interface number [input | output]`	Displays the packet statistics of all classes that are configured for all service policies on the specified interface.

Configuration Examples for Frame Relay Queueing and Fragmentation at the Interface

Example Frame Relay Queueing Shaping and Fragmentation at the Interface

The following example shows the configuration of a hierarchical policy for low-latency queueing, FRF.12 fragmentation, and shaping on serial interface 3/2. Note that traffic from the priority queue will not be interleaved with fragments from the class-default queue because shaping is configured.

```
class-map voice
 match access-group 101

policy-map llq
 class voice
  priority 64

policy-map shaper
 class class-default
  shape average 96000
  service-policy llq
interface serial 3/2
 ip address 10.0.0.1 255.0.0.0
 encapsulation frame-relay
 bandwidth 128
 clock rate 128000
```

```
 service-policy output shaper
 frame-relay fragment 80 end-to-end

 access-list 101 match ip any host 10.0.0.2
```

Example Frame Relay Queueing and Fragmentation at the Interface

The following example shows the configuration of low-latency queueing and FRF.12 fragmentation on serial interface 3/2. Because shaping is not being used, a hierarchical traffic policy is not needed and traffic from the priority queue will be interleaved with fragments from the other queues. Without shaping, the output rate of the interface is equal to the line rate or configured clock rate. In this example, the clock rate is 128,000 bps.

```
class-map voice
 match access-group 101

policy-map llq
 class voice
  priority 64
 class video
  bandwidth 32
interface serial 3/2
 ip address 10.0.0.1 255.0.0.0
 encapsulation frame-relay
 bandwidth 128
 clock rate 128000
 service-policy output llq
 frame-relay fragment 80 end-to-end
 access-list 101 match ip any host 10.0.0.2
```

Additional References

Related Documents

Related Topic	Document Title
Frame Relay configuration	*Cisco IOS Wide-Area Networking Configuration Guide , Release 12.4T*
Frame Relay commands	*Cisco IOS Wide-Area Networking Command Reference, Release 12.4T*

Standards

Standard	Title
FRF.16.1	*Multilink Frame Relay UNI/NNI Implementation Agreement,* May 2002

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

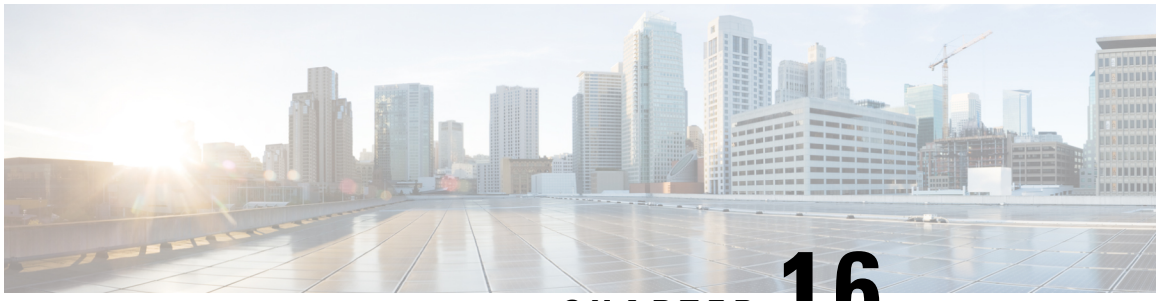
Feature Information for Frame Relay Queueing and Fragmentation at the Interface

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 29: Feature Information for Frame Relay Queueing and Fragmentation at the Interface

Feature Name	Releases	Feature Information
Frame Relay Queueing and Fragmentation at the Interface	Cisco IOS XE Release 2.1	The Frame Relay Queueing and Fragmentation at the Interface feature introduces support for low-latency queueing (LLQ) and FRF.12 end-to-end fragmentation on a Frame Relay interface.



CHAPTER 16

Frame Relay MIB Enhancements

The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using Simple Network Management Protocol (SNMP). Frame Relay fragmentation is supported in the MIB.

- Prerequisites for Frame Relay MIB Enhancements, on page 319
- Restrictions for Frame Relay MIB Enhancements, on page 319
- Information About Frame Relay MIB Enhancements, on page 320
- How to Configure Frame Relay MIB Enhancements, on page 321
- Configuration Examples for Frame Relay MIB Enhancements, on page 321
- Additional References, on page 322
- Feature Information for Frame Relay MIB Enhancements, on page 323

Prerequisites for Frame Relay MIB Enhancements

The tasks in this document assume that you have configured Frame Relay and SNMP on your devices.

To access the information introduced by the Frame Relay MIB enhancements, you must have the Cisco Frame Relay MIB in the MIB file called CISCO-FRAME-RELAY-MIB.my compiled in your network management system (NMS) application. You can find this MIB on the Web at Cisco's MIB website at

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

Restrictions for Frame Relay MIB Enhancements

- Frame Relay-ATM Network Interworking (FRF.5)
- Frame Relay-ATM Service Interworking (FRF.8)
- Frame Relay switching

Information About Frame Relay MIB Enhancements

Feature Overview

The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using SNMP. The Frame Relay MIB Enhancements feature extends the Cisco Frame Relay MIB by adding MIB objects to monitor the following Frame Relay functionality:

- Frame Relay fragmentation
- Input and output rates of individual virtual circuits (VCs)

The table below describes the MIB tables and objects that are introduced by the Frame Relay MIB enhancements. For a complete description of the MIB, see the Cisco Frame Relay MIB file CISCO-FRAME-RELAY-MIB.my, available through Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

Table 30: MIB Tables and Objects Introduced by the Frame Relay MIB Enhancements

Table or Object	Description
cfrFragTable	Table of Frame Relay fragmentation information.
cfrFRF5ConnectionTable	Table of Frame Relay-ATM Network Interworking connection information.
cfrFRF8ConnectionTable	Table of Frame Relay-ATM Service Interworking connection information.
cfrSwitchingTable	Table of Frame Relay switching entries.
cfrExtCircuitTxDataRate	Average rate, in bytes per second, at which data is transmitted on a circuit.
cfrExtCircuitTxPktRate	Average number of packets sent per second on a circuit.
cfrExtCircuitRcvDataRate	Average rate, in bytes per second, at which data is received on a circuit.
cfrExtCircuitRcvPktRate	Average number of packets received per second on a circuit.

The Frame Relay MIB Enhancements feature also modifies the **load-interval** command to enable you to configure the load interval per permanent virtual circuit (PVC). The load interval is the length of time for which data is used to compute load statistics, including input rate in bits and packets per second, output rate in bits and packets per second, load, and reliability. Before the introduction of this feature, the load interval could be configured only for the interface.

Benefits

The strict priority queueing scheme allows delay-sensitive data such as voice to be dequeued and sent first--that is, before packets in other queues are dequeued. Delay-sensitive data is given preferential treatment over other traffic. This process is performed on a per-PVC basis, rather than at the interface level.

How to Configure Frame Relay MIB Enhancements

Setting the Load Interval for a PVC

You can change the period of time over which a set of data is used for computing load statistics. Decisions, such as for dial backup, depend on these statistics. If you decrease the load interval, the average statistics are computed over a shorter period of time and are more responsive to bursts of traffic.

To change the length of time for which a set of data is used to compute load statistics for a PVC, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. Router(config-if)# **frame-relay interface-dlci** *dlci*
2. router(config-fr-dlci)# **load-interval** *seconds*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay interface-dlci** *dlci*	Assigns a specific PVC to a DLCI[4], and enters Frame Relay DLCI configuration mode.
Step 2	router(config-fr-dlci)# **load-interval** *seconds*	Changes the length of time for which data is used to compute load statistics. The seconds argument must be a multiple of 30. The range is from 30 to 300 seconds. The default is 300 seconds.

Verifying the Load Interval

Use the **show running-config** command to verify that you have configured the load interval correctly.

Configuration Examples for Frame Relay MIB Enhancements

Example Setting the Load Interval for a PVC

In the following example, the load interval is set to 60 seconds for a Frame Relay PVC with the DLCI 100:

```
interface serial 1/1
 frame-relay interface-dlci 100
  load-interval 60
```

Additional References

Related Documents

Related Topic	Document Title
WAN commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Wide-Area Networking Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this functionality.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this functionality.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

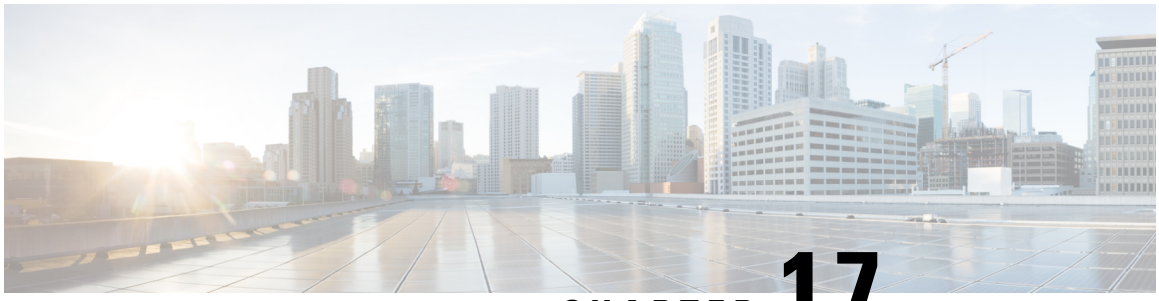
Feature Information for Frame Relay MIB Enhancements

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 31: Feature Information for Frame Relay MIB Enhancements

Feature Name	Releases	Feature Information
Frame Relay MIB Enhancements	Cisco IOS XE Release 2.1	The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using SNMP.



CHAPTER 17

Frame Relay PVC Interface Priority Queueing

The Frame Relay PVC Interface Priority Queueing feature provides an interface-level priority queueing scheme in which prioritization is based on destination permanent virtual circuit (PVC) rather than packet contents.

- Prerequisites for Frame Relay PVC Interface Priority Queueing, on page 325
- Restrictions for Frame Relay PVC Interface Priority Queueing, on page 325
- Information About Frame Relay PVC Interface Priority Queueing, on page 326
- How to Configure Frame Relay PVC Interface Priority Queueing, on page 327
- Configuration Examples for Frame Relay PVC Interface Priority Queuing, on page 329
- Additional References, on page 330
- Feature Information for Frame Relay PVC Interface Priority Queueing, on page 331
- Glossary, on page 332

Prerequisites for Frame Relay PVC Interface Priority Queueing

- PVCs should be configured to carry a single type of traffic.
- The network should be configured with adequate call admission control to prevent starvation of any of the priority queues.

Restrictions for Frame Relay PVC Interface Priority Queueing

- FR PIPQ is not supported on loopback or tunnel interfaces, or interfaces that explicitly disallow priority queueing.
- FR PIPQ is not supported with hardware compression.
- FR PIPQ cannot be enabled on an interface that is already configured with queueing other than FIFO queueing. FR PIPQ can be enabled if WFQ is configured, as long as WFQ is the default interface queueing method.

Information About Frame Relay PVC Interface Priority Queueing

Feature Overview

The Cisco Frame Relay MIB describes managed objects that enable users to remotely monitor Frame Relay operations using Simple Network Management Protocol (SNMP). The Frame Relay MIB Enhancements feature extends the Cisco Frame Relay MIB by adding MIB objects to monitor the following Frame Relay functionality:

- Frame Relay fragmentation
- Frame Relay-ATM Network Interworking (FRF.5)
- Frame Relay-ATM Service Interworking (FRF.8)
- Frame Relay switching
- Input and output rates of individual virtual circuits (VCs)

The table below describes the MIB tables and objects that are introduced by the Frame Relay MIB enhancements. For a complete description of the MIB, see the Cisco Frame Relay MIB file CISCO-FRAME-RELAY-MIB.my, available through Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

Table 32: MIB Tables and Objects Introduced by the Frame Relay MIB Enhancements

Table or Object	Description
cfrFragTable	Table of Frame Relay fragmentation information.
cfrFRF5ConnectionTable	Table of Frame Relay-ATM Network Interworking connection information.
cfrFRF8ConnectionTable	Table of Frame Relay-ATM Service Interworking connection information.
cfrSwitchingTable	Table of Frame Relay switching entries.
cfrExtCircuitTxDataRate	Average rate, in bytes per second, at which data is transmitted on a circuit.
cfrExtCircuitTxPktRate	Average number of packets sent per second on a circuit.
cfrExtCircuitRcvDataRate	Average rate, in bytes per second, at which data is received on a circuit.
cfrExtCircuitRcvPktRate	Average number of packets received per second on a circuit.

The Frame Relay MIB Enhancements feature also modifies the **load-interval** command to enable you to configure the load interval per permanent virtual circuit (PVC). The load interval is the length of time for which data is used to compute load statistics, including input rate in bits and packets per second, output rate in bits and packets per second, load, and reliability. Before the introduction of this feature, the load interval could be configured only for the interface.

Benefits

FR PIPQ provides four levels of PVC priority: high, medium, normal, and low. This method of queueing ensures that time/delay-sensitive traffic such as voice has absolute priority over signalling traffic, and that signalling traffic has absolute priority over data traffic, providing different PVCs are used for the different types of traffic.

How to Configure Frame Relay PVC Interface Priority Queueing

Configuring PVC Priority in a Map Class

To configure PVC priority within a map class, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **map-class frame-relay** *map-class-name*
2. Router(config-map-class)# **frame-relay interface-queue priority** {**high** | **medium**| **normal** | **low**}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **map-class frame-relay** *map-class-name*	Specifies a Frame Relay map class.
Step 2	Router(config-map-class)# **frame-relay interface-queue priority** {**high** | **medium**| **normal** | **low**}	Assigns a PVC priority level to a Frame Relay map class.

Enabling FR PIPQ and Setting Queue Limits

To enable FR PIPQ and set the priority queue sizes, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **interface** *type number* [*name-tag*]
2. Router(config-if)# **encapsulation frame-relay**[**cisco** | **ietf**]
3. Router(config-if)# **frame-relay interface-queue priority** [*high-limit medium-limit normal-limit low-limit*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config)# **interface** *type number* [*name-tag*]	Configures an interface type and enters interface configuration mode.
Step 2	Router(config-if)# **encapsulation frame-relay**[**cisco** | **ietf**]	Enables Frame Relay encapsulation.
Step 3	Router(config-if)# **frame-relay interface-queue priority** [*high-limit medium-limit normal-limit low-limit*]	Enables FR PIPQ and sets the priority queue limits.

Assigning a Map Class to a PVC

To assign a map class to a specific PVC, use the following commands beginning in interface configuration mode:

SUMMARY STEPS

1. Router(config-if)# **frame-relay interface-dlci** *dlci*
2. Router(config-fr-dlci)# **class** *map-class-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Router(config-if)# **frame-relay interface-dlci** *dlci*	Specifies a single PVC on a Frame Relay interface.
Step 2	Router(config-fr-dlci)# **class** *map-class-name*	Associates a map class with a specified PVC.

Verifying FR PIPQ

To verify the configuration of FR PIPQ, use one or more of the following commands in privileged EXEC mode:

Command	Purpose
`Router# show frame-relay pvc [interface interface][dlci]`	Displays statistics about PVCs for Frame Relay interfaces.
`Router# show interfaces [type number][first][last]`	Displays the statistical information specific to a serial interface.

Command	Purpose
`Router# show queueing [custom | fair | priority | random-detect [interface atm_subinterface [vc [[vpi/] vci]]]]`	Lists all or selected configured queueing strategies.

Monitoring and Maintaining FR PIPQ

To monitor and maintain FR PIPQ, use one or more of the following commands in privileged EXEC mode:

Command	Purpose
`Router# debug priority`	Debugs priority output queueing.
`Router# show frame-relay pvc [interface interface][dlci]`	Displays statistics about PVCs for Frame Relay interfaces.
`Router# show interfaces [type number][first][last]`	Displays the statistical information specific to a serial interface.
`Router# show queue interface-name interface-number [vc [vpi/] vci][queue-number]`	Displays the contents of packets inside a queue for a particular interface or VC.
`Router# show queueing [custom | fair | priority | random-detect [interface atm_subinterface [vc [[vpi/] vci]]]]`	Lists all or selected configured queueing strategies.

Configuration Examples for Frame Relay PVC Interface Priority Queuing

• Monitoring and Maintaining FR PIPQ, on page 329

FR PIPQ Configuration Example

This example shows the configuration of four PVCs on serial interface 0. DLCI 100 is assigned high priority, DLCI 200 is assigned medium priority, DLCI 300 is assigned normal priority, and DLCI 400 is assigned low priority.

The following commands configure Frame Relay map classes with PVC priority levels:

```
Router(config)# map-class frame-relay HI
Router(config-map-class)# frame-relay interface-queue priority high
```

```
Router(config-map-class)# exit
Router(config)# map-class frame-relay MED
Router(config-map-class)# frame-relay interface-queue priority medium
Router(config-map-class)# exit
Router(config)# map-class frame-relay NORM
Router(config-map-class)# frame-relay interface-queue priority normal
Router(config-map-class)# exit
Router(config)# map-class frame-relay LOW
Router(config-map-class)# frame-relay interface-queue priority low
Router(config-map-class)# exit
```

The following commands enable Frame Relay encapsulation and FR PIPQ on serial interface 0. The sizes of the priority queues are set at a maximum of 20 packets for the high priority queue, 40 for the medium priority queue, 60 for the normal priority queue, and 80 for the low priority queue.

```
Router(config)# interface Serial0
Router(config-if)# encapsulation frame-relay
Router(config-if)# frame-relay interface-queue priority 20 40 60 80
```

The following commands assign priority to four PVCs by associating the DLCIs with the configured map classes:

```
Router(config-if)# frame-relay interface-dlci 100
Router(config-fr-dlci)# class HI
Router(config-fr-dlci)# exit
Router(config-if)# frame-relay interface-dlci 200
Router(config-fr-dlci)# class MED
Router(config-fr-dlci)# exit
Router(config-if)# frame-relay interface-dlci 300
Router(config-fr-dlci)# class NORM
Router(config-fr-dlci)# exit
Router(config-if)# frame-relay interface-dlci 400
Router(config-fr-dlci)# class LOW
Router(config-fr-dlci)# exit
```

Additional References

Related Documents

Related Topic	Document Title
WAN commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Wide-Area Networking Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this functionality.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this functionality.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Frame Relay PVC Interface Priority Queueing

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 33: Feature Information for Frame Relay PVC Interface Priority Queueing

Feature Name	Releases	Feature Information
Frame Relay PVC Interface Priority Queueing	Cisco IOS XE Release 2.1	The FR PIPQ feature provides an interface-level priority queueing scheme in which prioritization is based on destination permanent virtual circuit (PVC) rather than packet contents. For example, FR PIPQ allows you to configure a PVC transporting voice traffic to have absolute priority over a PVC transporting signalling traffic, and a PVC transporting signalling traffic to have absolute priority over a PVC transporting data.

Glossary

DLCI --data-link connection identifier. Value that specifies a permanent virtual circuit (PVC) or switched virtual circuit (SVC) in a Frame Relay network.

FIFO queueing -- First-in, first-out queueing. FIFO involves buffering and forwarding of packets in the order of arrival. FIFO embodies no concept of priority or classes of traffic. There is only one queue, and all packets are treated equally. Packets are sent out an interface in the order in which they arrive.

Frame Relay traffic shaping --See FRTS.

FRF.12 --The FRF.12 Implementation Agreement was developed to allow long data frames to be fragmented into smaller pieces and interleaved with real-time frames. In this way, real-time voice and nonreal-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic.

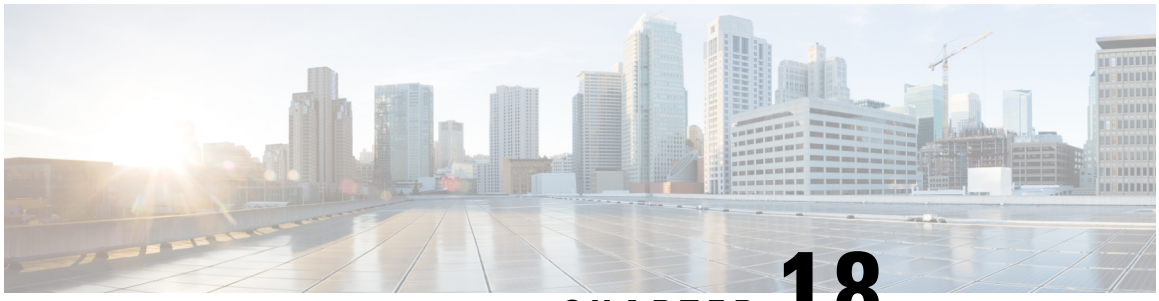
FRTS --Frame Relay traffic shaping. FRTS uses queues on a Frame Relay network to limit surges that can cause congestion. Data is buffered and then sent into the network in regulated amounts to ensure that the traffic will fit within the promised traffic envelope for the particular connection.

PIPQ --Permanent virtual circuit (PVC) interface priority queueing. An interface-level priority queueing scheme in which prioritization is based on destination PVC rather than packet contents.

quality of service --Measure of performance for a transmission system that reflects its transmission quality and service availability.

WFQ --weighted fair queueing. Congestion management algorithm that identifies conversations (in the form of traffic streams), separates packets that belong to each conversation, and ensures that capacity is shared fairly among these individual conversations. WFQ is an automatic way of stabilizing network behavior during congestion and results in increased performance and reduced retransmission.

WRED --Weighted Random Early Detection. Combines IP Precedence and standard Random Early Detection (RED) to allow for preferential handling of voice traffic under congestion conditions without exacerbating the congestion. WRED uses and interprets IP Precedence to give priority to voice traffic over data traffic, dropping only data packets.



CHAPTER 18

ASR1K Frame Relay - Multilink (MLFR-FRF.16)

The ASR1K Frame Relay - Multilink (MLFR-FRF.16) feature is based on the Frame Relay Forum Multilink Frame Relay User-to-Network Interface/Network-to-Network Interface (UNI/NNI) Implementation Agreement (FRF.16.1) on Cisco Aggregation Services Routers. This feature provides a cost-effective way to increase the bandwidth for particular applications by enabling multiple serial links to be aggregated into a single bundle of bandwidth. Multilink Frame Relay (MFR) is supported on UNI in Frame Relay networks.

- Prerequisites for ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 333
- Restrictions for ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 333
- Information About ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 334
- How to Enable ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 338
- Configuration Examples for ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 345
- Additional References, on page 347
- Feature Information for ASR1K Frame Relay - Multilink (MLFR-FRF.16), on page 347
- Glossary, on page 348

Prerequisites for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

- MFR must be configured on the peer device.

Restrictions for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

- Only the 2-octet Frame Relay format is supported.
- Only T1 and E1 speed members are supported in a bundle.
- All member links of a bundle must be of the same type.
- The following Shared Port Adapter (SPA) types are supported:
  - SPA-2XCT3/DS0
  - SPA-4XCT3/DS0
  - SPA-8XCHT1/E1
  - SPA-1XCHOC12/DS0
  - SPA-1XCHSTM1/OC3

- The following features are not supported with the ASR1K Frame Relay - Multilink (MLFR-FRF.16) feature:
  - 3- or 4-octet headers
  - Data-link connection identifier (DLCI) address mapping
  - Discard Eligibility (DE) bit manipulation
  - E1/T1 fractional links within the bundle
  - Frame Relay broadcast queue
  - Frame Relay backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN) counting
  - Frame Relay Permanent Virtual Circuit (PVC) interface priority queuing (PIPQ) including DLCI prioritization
  - Frame Relay switching including NNI and FRF2.1
  - Frame Relay Traffic Policing (FRTP)
  - Frame Relay Traffic Shaping (FRTS)
  - FRF.16.1 Fragmentation
  - Generic Traffic Shaping (GTS)
  - Inverse Address Resolution Protocol (ARP)
  - PVC configuration over MFR bundle interface
  - Point-to-multipoint subinterfaces
  - Switched Virtual Circuits (SVC)
- An ISDN interface and any type of virtual interface cannot be a bundle link.
- The Multilink Frame Relay MIB (RFC 3020) is not supported.
- FRF.9 hardware compression over MFR is not supported.

Information About ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Benefits of ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Flexible Pool of Bandwidth

By combining multiple physical interfaces into a bundle, you can design a Frame Relay interface that has more bandwidth than is available from any single physical interface. For example, many new network applications require more bandwidth than is available on a T1 line. One option is to invest in a T3 line; however, T3 lines can be expensive and are not available in some locations. MFR provides a cost-effective solution to this problem by allowing multiple T1 lines to be aggregated into a single bundle of bandwidth.

Increased Service Resilience

When multiple physical interfaces are provisioned as a single bundle, they provide more service resilience than a single physical interface. If a link fails, the bundle continues to support the Frame Relay service by transmitting across the remaining bundle links.

Scalability

ASR1K supports up to 992 MFR bundles.

- MFR bundles—The following table shows the maximum number of MFR bundles supported on ASR1K based on the number of links in a bundle:

Table 34: Maximum MFR Bundles

Links per Bundle	Number of MFR Bundles
1	992
2	496
3	330
4	248

- Frame Relay DLCI—The number of Frame Relay DLCIs that can be configured on MFR subinterfaces equals the maximum number of MFR bundles on ASR1K. The maximum number of Frame Relay DLCIs that you can configure on MFR subinterfaces and in one MFR bundle is 992.
- MFR subinterface—Because only point-to-point interfaces are supported, the number of DLCIs supported is equal to the number of MFR subinterfaces. Therefore, the maximum number of MFR subinterfaces and the maximum number of MFR interfaces supported in one bundle is 992.
- Physical Links—The maximum number of physical links supported in a bundle is 10.

Link Integrity Protocol Control Messages

For link management, each end of a bundle link follows the MFR Link Integrity Protocol and exchanges link control messages with its peer (the other end of the bundle link). To bring up a bundle link, both ends of the link must complete an exchange of ADD_LINK and ADD_LINK_ACK messages. To maintain the link, both ends periodically exchange HELLO and HELLO_ACK messages. This exchange of hello messages and acknowledgments serve as a keepalive mechanism for the link. If a router is sending hello messages but not receiving acknowledgments, it will resend the hello message up to a configured maximum number of times. If the router exhausts the maximum number of retries, the bundle link line protocol is considered down (unoperational).

The bundle link interface's line protocol status is considered up (operational) when the peer device acknowledges that it will use the same link for the bundle. The line protocol remains up when the peer device acknowledges the hello messages from the local router.

The bundle interface's line status becomes up when at least one bundle link has its line protocol status up. The bundle interface's line status goes down when the last bundle link is no longer in the up state. This behavior complies with the class A bandwidth requirement defined in FRF.16.

The bundle interface's line protocol status is considered up when the Frame Relay data-link layer at the local router and peer device synchronize using the Local Management Interface (LMI), when LMI is enabled. The bundle line protocol remains up as long as the LMI keepalives are successful.

Variable Bandwidth Class Support

MFR FRF.16 variable bandwidth class support allows you to specify the criterion used to activate or deactivate a Frame Relay bundle.

Class A Single Link

The Frame Relay bundle is provisioned when one or more bundle links issue a BL_ACTIVATE message to indicate that an operational bandwidth is available. When this occurs, the bundle emulates a physical link by issuing a PH_ACTIVATE message to the data link layer.

When the operational bandwidth of a bundle link fails to meet operational requirements (for instance, if a bundle link is in rollback mode), the bundle link issues a BL_DEACTIVATE message. When all bundle links are down in a class A bundle, a PH_DEACTIVATE message is sent to the data link layer, indicating that the Frame Relay bundle cannot accept frames.



Note Activate and deactivate messages are implementation-oriented messages only. They are not visible in the output of the debug commands.

Class B All Links

The Frame Relay bundle is provisioned when all bundle links issue a BL_ACTIVATE message to indicate that an operational bandwidth is available. When this occurs, the bundle emulates a physical link by issuing a PH_ACTIVATE message to the data link layer.

When the operational bandwidth of a bundle link fails to meet operational requirements (for instance, if it is in loopback mode), the bundle link issues a BL_DEACTIVATE message. When any bundle link is down in a class B bundle, a PH_DEACTIVATE message is sent to the data link layer, indicating that the Frame Relay bundle cannot accept frames.

Class C Threshold

A Frame Relay bundle is provisioned when a minimum number of links in the configured bundle issue a BL_ACTIVATE message, causing the bundle to emulate a physical link by issuing a PH_ACTIVATE message to the data link layer.

When the number of bundle links issuing a BL_ACTIVATE message falls below the configured threshold value, a PH_DEACTIVATE message is sent to the data link layer, indicating that the Frame Relay bundle cannot accept frames.

Load Balancing

MFR provides load balancing across bundle links within a bundle. If a bundle link that is chosen for transmission happens to be busy transmitting a long packet, the load-balancing mechanism can try another link, thus solving problems encountered when delay-sensitive packets have to wait.

ASR1K FRF.12 Support on MFR Interfaces

The ASR1K FRF.12 Support on MFR Interfaces feature enables the transport of realtime, delay-sensitive (voice) and nonrealtime, delay-insensitive (data) packets over the same, relatively slow-speed PVC.

During the transmission of packets, the larger, nonrealtime packets are fragmented into a sequence of smaller, mostly fixed-sized packets, also called fragments. The realtime packets are interleaved among the fragments. While receiving the packets, the nonrealtime fragments are reassembled and the resulting packets are forwarded along with the realtime packets. This approach minimizes the delay that can occur when nonrealtime and realtime traffic flow over the same PVC.

Benefits of ASR1K FRF.12

The ASR1K FRF.12 functionality prevents delay in Frame Relay networks by allowing edge routers to fragment large data packets before transmitting them across the network.

Limitations of ASR1K FRF.12

If a Frame Relay access device does not support FRF.12 fragmentation, the ASR1K FRF.12 Support on MFR Interfaces feature will not benefit the interface between the Frame Relay access device and the edge router. Fragmentation and reassembly occur on the interface between the edge router and the Frame Relay network.

If the Frame Relay access device is sending voice and unfragmented data on the same PVC, voice quality will suffer. The edge router will not reorder packets on PVCs.

Selecting a Fragment Size

You should set the fragment size based on the lowest port speed between routers. For example, for a hub-and-spoke Frame Relay topology, where the hub has a T1 speed and the remote routers have 64 kb/s port speeds, the fragmentation size must be set for 64 kb/s speed on both routers. Any other PVCs that share the same physical interface must use the same fragmentation size used by the voice PVC.

With pure end-to-end FRF.12 fragmentation, you should select a fragment size that is larger than the voice packet size.

The following table shows the recommended fragmentation sizes for a serialization delay of 10 ms:

Table 35: Recommended Fragment Size for 10 ms Serialization Delay

Lowest Link Speed in Path	Recommended Fragment Size
56 kb/s	70 bytes
64 kb/s	80 bytes
128 kb/s	160 bytes
256 kb/s	320 bytes
512 kb/s	640 bytes
768 kb/s	1000 bytes
1536 kb/s	1600 bytes

How to Enable ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Configuring an MFR Bundle

Perform this task to configure an MFR bundle.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface mfr***interface-number*
4. **frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]]
5. **frame-relay intf-type** [**dce** | **dte**]
6. **frame-relay multilink bid** *name*
7. **exit**
8. **interface mfr***interface-number.subinterface-number* **point-to-point**
9. **ip address** *ip-address mask*
10. **frame-relay interface-dlci** *dlci*
11. **end**
12. **show frame-relay multilink**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface mfr***interface-number* **Example:** `Router(config)# interface mfr1`	Configures an MFR bundle interface and enters interface configuration mode.
Step 4	**frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]] **Example:** `Router(config-if)# frame-relay multilink bandwidth-class a`	(Optional) Specifies the bandwidth class criterion used to activate or deactivate a Frame Relay bundle. • Class A (single link)—The bundle will activate when any bundle link is up and deactivate when all bundle links are down (default).

	Command or Action	Purpose
		• Class B (all links)—The bundle will activate when all bundle links are up and deactivate when any bundle link is down. • Class C (threshold)—The bundle will activate when the minimum configured number of bundle links is up (the threshold) and deactivate when the minimum number of configured bundle links fails to meet the threshold. **Note** If no bandwidth class criterion is specified by using the **frame-relay multilink bandwidth-class** command, the Frame Relay bundle will default to class A (single link).
Step 5	**frame-relay intf-type** [**dce** | **dte**] **Example:** `Router(config-if)# frame-relay intf-type dce`	Configures a device to function as data communication equipment (DCE). • **dce**—(Optional) The router or access server functions as a switch connected to a router. • **dte**—(Optional) The router or access server is connected to a Frame Relay network. **Note** Only one end of a link should be configured as DCE. The other end will function as data terminal equipment (DTE), which is the default setting.
Step 6	**frame-relay multilink bid** *name* **Example:** `Router(config-if)# frame-relay multilink bid router1`	(Optional) Assigns a bundle identification name to an MFR bundle. • The bundle identification (BID) will not go into effect until the interface has gone from the "down" state to the "up" state. One way to bring the interface down and back up again is by using the **shutdown** and **no shutdown** commands in interface configuration mode (assuming that the physical state of the link is always up).
Step 7	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**interface mfr***interface-number***.***subinterface-number* **point-to-point** **Example:** `Router(config)# interface mfr1.1 point-to-point`	Configures a point-to-point MFR subinterface and enters subinterface configuration mode.

	Command or Action	Purpose
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.0.1.1 255.255.255.0`	Configures an IP address for the subinterface.
Step 10	**frame-relay interface-dlci** *dlci* **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a DLCI to a Frame Relay subinterface and enters Frame Relay DLCI configuration mode. • The DLCI range is from 16 to 1007.
Step 11	**end** **Example:** `Router(config-fr-dlci)# end`	Exits Frame Relay DLCI configuration mode and returns to privileged EXEC mode.
Step 12	**show frame-relay multilink** **Example:** `Router# show frame-relay multilink`	(Optional) Displays the current Frame Relay multilink configuration.

Configuring an MFR Bundle Link

Tip To minimize the latency that results from the arrival order of packets, Cisco recommends bundling physical links of the same line speed in one bundle.

Perform this task to configure an MFR bundle link.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface serial** *number*
4. **encapsulation frame-relay mfr***number* [*name*]
5. **frame-relay multilink lid** *name*
6. **frame-relay multilink hello** *seconds*
7. **frame-relay multilink ack** *seconds*
8. **frame-relay multilink retry** *number*
9. **end**
10. **show frame-relay multilink**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface serial** *number* **Example:** `Router(config)# interface serial 5/0`	Selects a physical interface and enters interface configuration mode.
Step 4	**encapsulation frame-relay mfr***number* [*name*] **Example:** `Router(config-if)# encapsulation frame-relay mfr1`	Creates an MFR bundle link and associates the link with a bundle.
Step 5	**frame-relay multilink lid** *name* **Example:** `Router(config-if)# frame-relay multilink lid first-link`	(Optional) Assigns a bundle link identification name to an MFR bundle link. • The bundle link identification (LID) is not functional until the interface has gone from the "down" state to the "up" state. One way to bring the interface down and back up again is by using the **shutdown** and **no shutdown** commands in interface configuration mode.
Step 6	**frame-relay multilink hello** *seconds* **Example:** `Router(config-if)# frame-relay multilink hello 9`	(Optional) Configures the interval in seconds after which a bundle link will send out hello messages. • The default value is 10 seconds.
Step 7	**frame-relay multilink ack** *seconds* **Example:** `Router(config-if)# frame-relay multilink ack 6`	(Optional) Configures the interval (in seconds) for which a bundle link will wait for a hello message acknowledgment before resending the hello message. • The default value is 4 seconds.
Step 8	**frame-relay multilink retry** *number* **Example:** `Router(config-if)# frame-relay multilink retry 3`	(Optional) Configures the maximum number of times a bundle link will resend a hello message while waiting for an acknowledgment. • The default value is 2 tries.
Step 9	**end** **Example:**	Ends the configuration session and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Router(config-if)# end`	
Step 10	**show frame-relay multilink** **Example:** `Router# show frame-relay multilink`	(Optional) Displays the current Frame Relay multilink configuration.

Configuring FRF.12 on an MFR Bundle Interface

Before you begin

You must create a class map and a policy map before enabling FRF.12 fragmentation of Frame Relay frames. For the class map, define a differentiated services code point (DSCP) value as the match criterion.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface mfr***interface-number*
4. **no ip address**
5. **frame-relay fragment** *fragment-size* **end-to-end**
6. **service-policy output** *policy-map-name*
7. **exit**
8. **interface mfr***interface-number***.***subinterface-number* **point-to-point**
9. **ip address** *ip-address mask*
10. **frame-relay interface-dlci** *dlci-value*
11. **end**
12. **show frame-relay fragment** [**interface** *interface* [*dlci*]]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface mfr***interface-number* **Example:** `Router(config)# interface mfr1`	Configures an MFR bundle interface and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**no ip address** **Example:** `Router(config-if)# no ip address`	Disables IP processing.
Step 5	**frame-relay fragment** *fragment-size* **end-to-end** **Example:** `Router(config-if)# frame-relay fragment 300 end-to-end`	Enables FRF.12 end-to-end fragmentation of Frame Relay frames. • The valid size range is from 16 to 1600. • To maintain low latency and low jitter for priority queue traffic, configure the fragment size to be greater than the largest high-priority frame that would be expected.
Step 6	**service-policy output** *policy-map-name* **Example:** `Router(config-if)# service-policy output pmap1`	Attaches a policy map to an output interface that is to be used as the service policy for that interface.
Step 7	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**interface mfr***interface-number***.***subinterface-number* **point-to-point** **Example:** `Router(config)# interface mfr1.1 point-to-point`	Configures a point-to-point MFR subinterface and enters subinterface configuration mode.
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.1.1.1 255.255.255.0`	Configures the IP address of the subinterface.
Step 10	**frame-relay interface-dlci** *dlci-value* **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a DLCI to the MFR subinterface and enters Frame Relay DLCI configuration mode. • The DLCI range is from 16 to 1007.
Step 11	**end** **Example:** `Router(config-fr-dlci)# end`	Exits Frame Relay DLCI configuration mode and returns to privileged EXEC mode.
Step 12	Required: **show frame-relay fragment** [**interface** *interface* [*dlci*]] **Example:** `Router# show frame-relay fragment`	(Optional) Displays statistics about Frame Relay fragmentation.

Monitoring and Maintaining MFR Bundles and Bundle Links

SUMMARY STEPS

1. **enable**
2. **debug frame-relay multilink** [**control** [**mfr***number* | **serial** *number*]]
3. **show frame-relay multilink** [**mfr***number* | **serial** *number*] [**detailed**]
4. **show interfaces mfr***number*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug frame-relay multilink** [**control** [**mfr***number* | **serial** *number*]] **Example:** `Router# debug frame-relay multilink control mfr1`	(Optional) Displays debug messages for MFR bundles and bundle links.
Step 3	**show frame-relay multilink** [**mfr***number* | **serial** *number*] [**detailed**] **Example:** `Router# show frame-relay multilink mfr1 detailed`	(Optional) Displays configuration information and statistics about MFR bundles and bundle links.
Step 4	**show interfaces mfr***number* **Example:** `Router# show interfaces mfr1`	(Optional) Displays information and packet statistics for the bundle interface.

Examples

The following is sample output from the **show frame-relay multilink** command. Because a particular bundle or bundle link is not specified, information about all bundles and bundle links is displayed.

```
Router# show frame-relay multilink

Bundle: mfr1, State = down, class = A, fragmentation disabled
 BID = router1
 Bundle links:
  Serial3/1, HW state = Administratively down, link state = Down, LID = second-link
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
Bundle: mfr1, State = down, class = B, fragmentation disabled
 BID = router1
 Bundle links:
  Serial3/0, HW state = Administratively down, link state = Down, LID = third-link
```

```
  Serial3/1, HW state = Administratively down, link state = Down, LID = second-link
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
```

The following is sample output from the **show frame-relay multilink** command when a Frame Relay bundle is configured as bandwidth class C (threshold):

```
Router# show frame-relay multilink

Bundle: mfr2, State = down, class = C (threshold 100), fragmentation disabled
 BID = router2
 Bundle links:
  Serial3/1, HW state = Administratively down, link state = Down, LID = cisco2
  Serial3/0, HW state = Administratively down, link state = Down, LID = cisco1
```

The following is sample output from the **show frame-relay multilink** command when the **serial** *number* keyword and argument pair is specified. It displays information about the specified bundle link.

```
Router# show frame-relay multilink Serial 3/2

Bundle links:
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
  Bundle interface = mfr1,  BID = router1
```

The following is sample output from the **show frame-relay multilink** command when the **serial** *number* keyword and argument pair and the **detailed** option are specified. Detailed information about the specified bundle links is displayed.

```
Router# show frame-relay multilink Serial 3/2 detail

Bundle links:
  Serial3/2, HW state = up, link state = Add_sent, LID = first-link
  Bundle interface = mfr1,  BID = router1
    Cause code = none, Ack timer = 6, Hello timer = 9,
    Max retry count = 3, Current count = 0,
    Peer LID = , RTT = 0 ms
    Statistics:
    Add_link sent = 110, Add_link rcv'd = 0,
    Add_link ack sent = 0, Add_link ack rcv'd = 0,
    Add_link rej sent = 0, Add_link rej rcv'd = 0,
    Remove_link sent = 0, Remove_link rcv'd = 0,
    Remove_link_ack sent = 0, Remove_link_ack rcv'd = 0,
    Hello sent = 0, Hello rcv'd = 0,
    Hello_ack sent = 0, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
```

Configuration Examples for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Example: Configuring Multilink Frame Relay

The following example shows the configuration of bundle MFR1, where serial interfaces 3/0 and 3/2 are configured as bundle links:

```
interface MFR1
 no ip address
 frame-relay intf-type dce
```

```
 frame-relay multilink bid router1
!
interface MFR1.1 point-to-point
 ip address 10.0.0.1 255.255.255.0
 frame-relay interface-dlci 100
interface Serial3/0
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 9
 frame-relay multilink retry 3
 frame-relay multilink ack 4
interface Serial3/2
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 8
 frame-relay multilink ack 3
 frame-relay multilink retry 2
```

Example: Configuring Variable Bandwidth Class Support

The following example shows how to configure Frame Relay bundle MFR2 to use the class B (all links) criterion to get activated or deactivated:

```
interface MFR2
 frame-relay multilink bandwidth-class b
 frame-relay intf-type dce
 frame-relay multilink bid router2
 exit
interface MFR2.2 point-to-point
 ip address 10.1.1.10 255.255.255.0
 frame-relay interface-dlci 145
  end
```

Example: Configuring FRF.12 on an MFR Interface

The following example shows how to configure FRF.12 on an MFR interface:

```
class-map match-any tos_111
 match dscp cs7
policy-map voip
 class tos_111
  priority percent 100
interface mfr1
 frame-relay multilink bid 1
 frame-relay multilink bandwidth-class a
 frame-relay fragment 100 end-to-end
 service-policy output voip
interface mfr1.1 point-to-point
 ip address 70.1.1.1 255.255.255.0
 frame-relay interface-dlci 100
```

The following output shows the result of the above configuration:

```
Router# show frame-relay fragment

interface  dlci  frag-type   size  in-frag  out-frag  dropped-frag
mfr1.1     100   end-to-end  100   0        0         0
```

The size column displays the configured fragment size in bytes.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WAN commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS Wide-Area Networking Command Reference
Frame Relay configuration	Configuring Frame Relay

Standards and RFCs

Standard/RFC	Title
FRF.16.1	*Multilink Frame Relay UNI/NNI Implementation Agreement*, May 2002

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 36: Feature Information for ASR1K Frame Relay - Multilink (MLFR-FRF.16)

Feature Name	Releases	Feature Information
ASR1K Frame Relay - Multilink (MLFR-FRF.16)	Cisco IOS XE Release 3.4S	The ASR1K Frame Relay - Multilink (MLFR-FRF.16) feature is based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1) on Aggregation Service Routers. The following commands were introduced or modified: **debug frame-relay multilink**, **encapsulation frame-relay mfr**, **frame-relay multilink ack**, **frame-relay multilink bandwidth-class**, **frame-relay multilink bid**, **frame-relay multilink hello**, **frame-relay multilink lid**, **frame-relay multilink retry**, **interface mfr**, **show frame-relay multilink**.
ASR1K FRF.12 Support on MFR Interfaces	Cisco IOS XE Release 3.5S	The following sections provide information about this feature: • ASR1K FRF.12 Support on MFR Interfaces • Configuring FRF.12 on an MFR Bundle Interface

Glossary

BID --Bundle identification. The BID is the name used to identify the bundle. The BID can be assigned, or the default can be used.

BL_ACTIVATE --A message that controls the addition of a bundle link to a Frame Relay bundle.

BL_DEACTIVATE --A message that controls the removal a bundle link from a Frame Relay bundle.

bundle --A logical grouping of one or more physical interfaces using the formats and procedures of multilink Frame Relay. A bundle emulates a physical interface to the Frame Relay data-link layer. The bundle is also referred to as the *MFR interface* .

bundle link --An individual physical interface that is a member of a bundle.

DLCI --data-link connection identifier. A value that identifies a permanent virtual circuit (PVC) in a Frame Relay network.

HELLO message --A message that notifies a peer endpoint that the local endpoint is in the operational state (up).

HELLO_ACK --A message that notifies a peer endpoint that a hello message has been received.

LID --link identification. The LID is the name used to identify a bundle link. The LID can be assigned, or the default can be used.

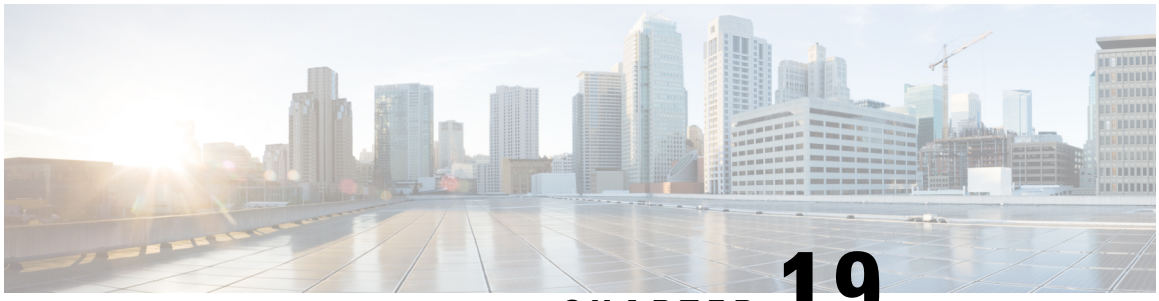
LMI --Local Management Interface. A set of enhancements to the basic Frame Relay specification. LMI includes support for a keepalive mechanism, which verifies that data is flowing; a multicast mechanism, which provides the network server with its local DLCI and the multicast DLCI; global addressing, which gives DLCIs global rather than local significance in Frame Relay networks; and a status mechanism, which provides an ongoing status report on the DLCIs known to the switch.

NNI --Network-to-Network Interface. The interface between two Frame Relay devices that are both located in a private network or both located in a public network.

PH_ACTIVATE --A message that indicates that the Frame Relay bundle is up.

PH_DEACTIVATE --A message that indicates that the Frame Relay bundle is down.

UNI --User-to-Network Interface. The interface between a Frame Relay device in a public network and a Frame Relay device in a private network.



CHAPTER 19

Frame Relay show Command and debug Command Enhancements

The Frame Relay show Command and debug Command Enhancements feature provides the ability to filter the output of certain Frame Relay **show** and **debug** commands on the basis of the interface and data-link connection identifier (DLCI). These enhancements facilitate network scalability and simplify network management and troubleshooting.

- Information About Frame Relay show Command and debug Command Enhancements, on page 351
- Additional References, on page 352
- Feature Information for Frame Relay show Command and debug Command Enhancements, on page 353

Information About Frame Relay show Command and debug Command Enhancements

Overview of the Frame Relay show Command and debug Command Enhancements

This feature introduces the following enhancments:

- The **show frame-relay map** command has been enhanced to allow map information to be displayed for specific interfaces and DLCIs.
- The **show frame-relay ip tcp header-compression** and **show frame-relay ip rtp header-compression** commands have been enhanced to allow header-compression information to be displayed for specific DLCIs.
- The **summary** keyword was added to the **show frame-relay pvc** command, allowing a summary of all PVCs on the system to be displayed.
- Conditional debugging support, which allows debug output to be filtered on the basis of interface and DLCI, was introduced for the following commands:
  - **debug frame-relay end-to-end**
  - **debug frame-relay events**
  - **debug frame-relay fragment**

- **debug frame-relay fragment event**
- **debug frame-relay ip**
- **debug frame-relay ppp**
- **debug frame-relay verbose**

Note Conditional debugging for Frame Relay **debug** commands is configured by using the **debug condition** command.

Benefits of the Frame Relay Show Command and Debug Command Enhancements

The Frame Relay show Command and debug Command Enhancements allow the output for some Frame Relay **show** commands and **debug** commands to be filtered on the basis of interface and DLCI. This enhancement saves network administrators time and frustration by eliminating the need to look through a large amount of output for information about a specific interface or DLCI. These enhancements can also reduce the amount of CPU processing time that is required to generate large amounts of **show** and **debug** output.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS XE Wide-Area Networking configuration tasks	*Cisco IOS XE Wide-Area Networking Configuration Guide*, Release 2
Wide-Area networking commands	*Cisco IOS Wide-Area Networking Command Reference*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

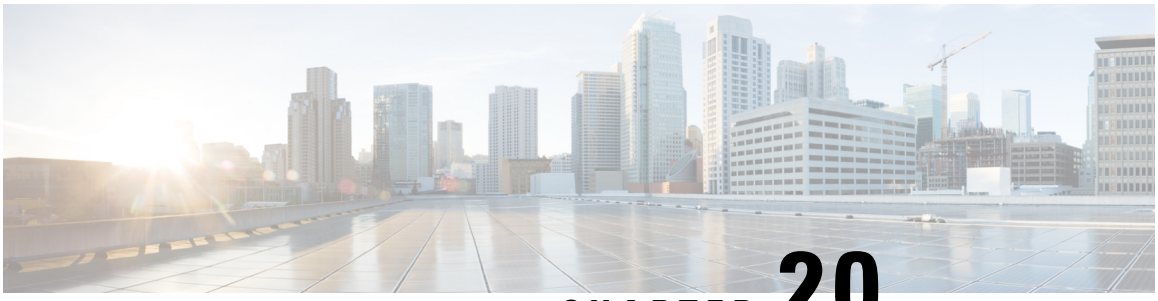
Feature Information for Frame Relay show Command and debug Command Enhancements

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 37: Feature Information for Frame Relay show Command and debug Command Enhancements

Feature Name	Releases	Feature Information
Frame Relay show Command and debug Command Enhancements	Cisco IOS XE Release 2.1	The Frame Relay show Command and debug Command Enhancements feature provides the ability to filter the output of certain Frame Relay **show** and **debug** commands on the basis of the interface and DLCI.



CHAPTER 20

L2VPN Local Switching—Frame Relay-Ethernet/VLAN

L2VPN Local Switching—Frame Relay-Ethernet/VLAN feature allows you to switch Frame Relay and Ethernet frames between two interfaces on the same device.

- Restrictions for L2VPN Local Switching—Frame Relay-Ethernet/VLAN , on page 355
- Information About L2VPN Local Switching—Frame Relay-Ethernet/VLAN , on page 355
- How To Configure L2VPN Local Switching—Frame Relay-Ethernet/VLAN, on page 358
- Configuration Examples for L2VPN Local Switching—Frame Relay-Ethernet/VLAN , on page 361
- Additional References for L2VPN Local Switching—Frame Relay-Ethernet/VLAN, on page 363
- Feature Information for L2VPN Local Switching—Frame Relay-Ethernet/VLAN, on page 364

Restrictions for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

The following functions are not supported:

- Frame Relay-to-Ethernet IP-Mode local switching
- Frame Relay-to-Ethernet VLAN-Mode local switching
- Frame Relay Multilink Frame Relay (MFR)

Information About L2VPN Local Switching—Frame Relay-Ethernet/VLAN

L2VPN Local Switching—Frame Relay-Ethernet/VLAN Overview

The L2VPN Local Switching—Frame Relay-Ethernet/VLAN feature switches a Frame Relay frame to an Ethernet VLAN/QinQ frame over the same provider edge (PE) device. Only Ethernet (bridged) interworking mode is supported to switch packets between Frame Relay link and Ethernet VLAN/QinQ. In a bridged interworking mode, the MAC header is considered as the payload of Frame Relay frames.

The L2VPN Local Switching—Frame Relay-Ethernet/VLAN supports the following functions:

- The Frame Relay-Ethernet bridge mode local switching in data-link connection identifier (DLCI) mode.
- Port interface and subinterface Ethernet attachment circuit (AC) type with single tag or double tags (Q-in-Q).
- Cisco and IETF Frame Relay encapsulation.

The Frame Relay-Ethernet local switching topology is illustrated in the figure below.

Figure 42: Frame Relay-Ethernet Local Switching Topology

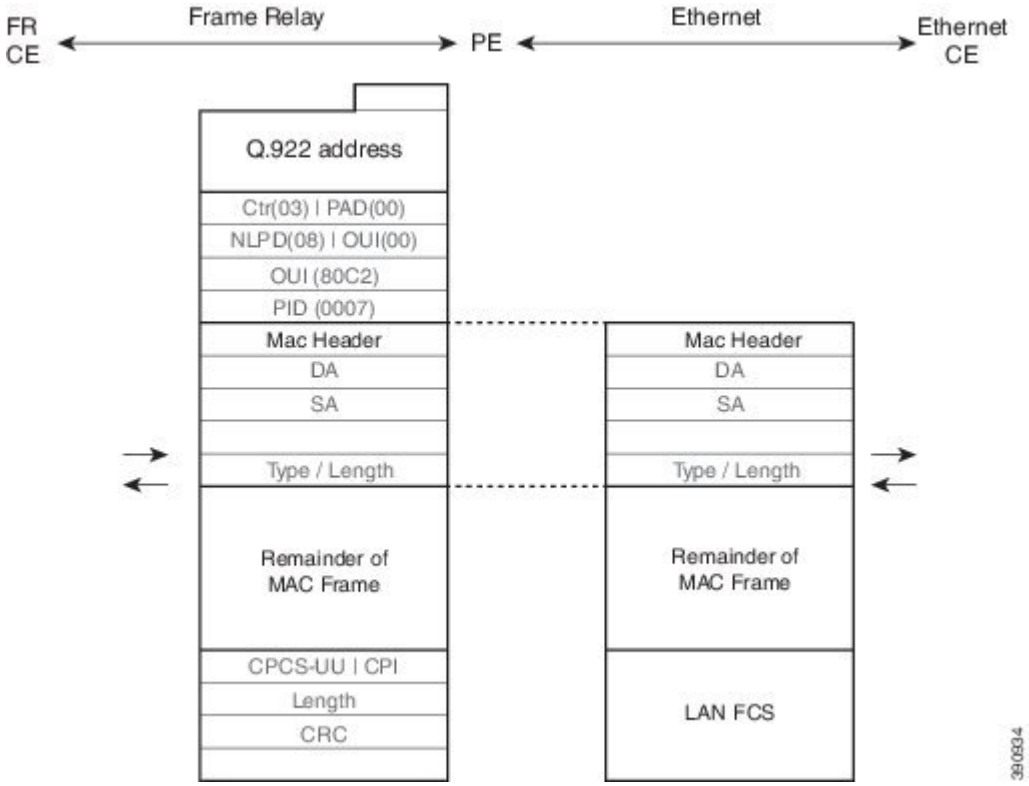


Frame Relay to Ethernet Port-Bridged Interworking

Frame Relay-Ethernet port-bridged interworking provides interoperability between a Frame Relay attachment virtual circuit (VC) and an Ethernet attachment VC connected to the same provider edge (PE) device. The bridged encapsulation is used that corresponds to the bridged (Ethernet) interworking mechanism.

Based on RFC 2427, *Multiprotocol Interconnect over Frame Relay*, the interworking is done at the PE connected to the Frame Relay attachment VC as shown in the figure below.

Figure 43: Protocol Stack for Frame Relay to Ethernet Port Bridged Interworking



The processing of Frame Relay-Ethernet port local switching is described as follows:

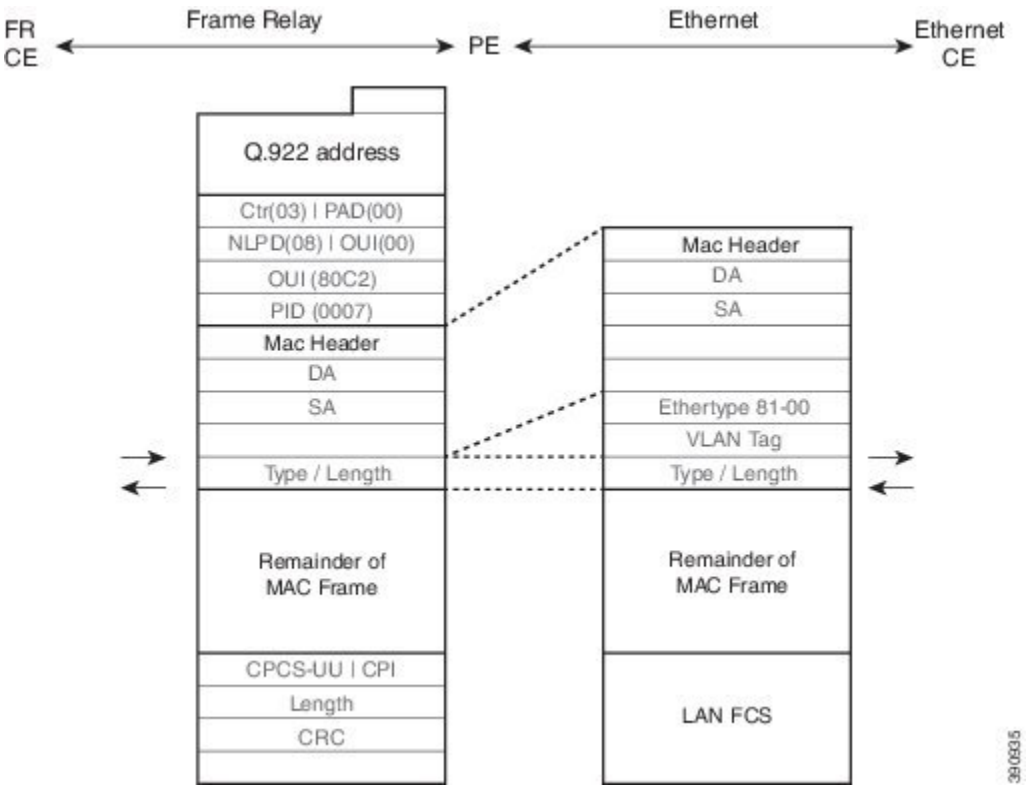
- In the direction from Frame Relay to Ethernet:
  - On the Frame Relay side, the Frame Relay header and trailer are removed. The packet is forwarded to Ethernet side.
  - On the Ethernet side, the MAC header is ignored.
- In the direction from Ethernet to Frame Relay:
  - On the Ethernet side, the MAC header is ignored.
  - On the Frame Relay side, the Frame Relay header is generated and added to the packet that is sent to the Frame Relay customer edge (CE) device.

Frame Relay to Ethernet VLAN/QinQ–Bridged Interworking

Frame Relay to Ethernet VLAN/QinQ bridged interworking provides interoperability between a Frame Relay attachment virtual circuit (VC) and an Ethernet VLAN attachment VC connected to the same provider edge (PE) device. The bridged encapsulation is used that corresponds to the bridged (Ethernet) interworking mechanism.

Based on RFC 2427, *Multiprotocol Interconnect over Frame Relay*, the interworking function is implemented on the PE connected to the Frame Relay attachment VC as shown in the figure below.

Figure 44: Protocol Stack for Frame Relay to Ethernet VLAN/QinQ Bridged Interworking



The process of Frame Relay to VLAN/QinQ bridged interworking is described as follows:

- In the direction from Frame Relay to Ethernet:
  - On the Frame Relay side, the Frame Relay header and trailer are removed. The packet is forwarded to Ethernet side.
  - On the Ethernet side, one or two VLAN tags are generated per the configuration and inserted into L2 header, which is referred as VLAN tag push.
- In the direction from Ethernet to Frame Relay:
  - On the Ethernet side, the one or two VLAN tags are removed. The packet is then forwarded to Frame Relay side.
  - On the Frame Relay side, the Frame Relay header is generated and added to the packet that is sent to the Frame Relay customer edge (CE) device.

How To Configure L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Configuring Frame Relay-Ethernet Port-Bridged Interworking

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no ip address**
5. **exit**
6. **interface** *type number*
7. **encapsulation frame-relay**
8. **frame-relay interface-dlci** *dlci* **switched**
9. **exit**
10. **connect** *connection-name type number dlci* **interworking ethernet**
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0`	Specifies the interface type and number and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface Serial 0/0/0:0`	Specifies the subinterface type and number and enters subinterface configuration mode.
Step 7	**encapsulation frame-relay** **Example:** `Device(config-subif)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 8	**frame-relay interface-dlci** *dlci* **switched** **Example:** `Device(config-subif)# frame-relay interface-dlci 57 switched`	Indicates that a Frame Relay data-link connection identifier (DLCI) is switched. The range is from 16 to 1007.
Step 9	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 10	**connect** *connection-name type number dlci* **interworking ethernet** **Example:** `Device(config)# connect Eth-Ser GigabitEthernet0/0/0 Serial0/0/0:0 57 interworking ethernet`	Creates Layer 2 data connections between two ports on the same device.
Step 11	**end** **Example:** `Device# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring Frame Relay-Ethernet VLAN/QinQ Interworking

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **encapsulation dot1q** *vlan-id* **second-dot1q** *second vlan-id*
5. **exit**
6. **interface** *type number*
7. **encapsulation frame-relay**
8. **frame-relay interface-dlci** *dlci* **switched**
9. **exit**
10. **connect** *connection-name type number dlci* **interworking ethernet**
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0.1`	Specifies a subinterface type and number and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **second-dot1q** *second vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 3 second-dot1q 4`	Specifies QinQ as the encapsulation method.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits subinterface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 6	**interface** *type number* **Example:** `Device(config)# interface Serial 0/0/0:0`	Specifies a subinterface type and number and enters subinterface configuration mode.
Step 7	**encapsulation frame-relay** **Example:** `Device(config-subif)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 8	**frame-relay interface-dlci** *dlci* **switched** **Example:** `Device(config-subif)# frame-relay interface-dlci 58 switched`	Indicates that a Frame Relay data-link connection identifier (DLCI) is switched.
Step 9	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 10	**connect** *connection-name type number dlci* **interworking ethernet** **Example:** `Device(config)# connect Eth-FR GigabitEthernet0/0/0.1 Serial0/0/0:0 58 interworking ethernet`	Creates Layer 2 data connections between two ports on the same device.
Step 11	**end** **Example:** `Device# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuration Examples for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Example: Configuring Frame Relay-Ethernet Port Mode Bridged Interworking

The following example shows how to configure the Frame Relay-Ethernet port mode bridged interworking:

PE configuration:

```
interface GigabitEthernet0/0/1
   no ip address
   end
interface Serial0/1/2:0
   no ip address
   encapsulation frame-relay
   no keepalive
   frame-relay interface-dlci 60 switched
```

```
   end
connect FR-ETHQinQ Serial0/1/2:0 60 GigabitEthernet0/0/1 interworking ethernet
```

CE configuration: Frame-Relay-CE

```
 bridge irb
 bridge 16 protocol ieee
 bridge 16 route ip
 interface Serial2/0:0
   no ip address
   encapsulation frame-relay IETF
   no keepalive
  interface Serial2/0:0.1 point-to-point
   frame-relay interface-dlci 60
   bridge-group 60
  interface BVI16
   ip address 172.16.1.0 255.255.0.0
```

Ethernet-CE

```
interface GigabitEthernet0/0/1
 ip address 172.16.2.1 255.255.0.0
```

Example: Configuring Frame Relay-Ethernet VLAN 802.1Q Bridged Interworking

The following example shows how to configure Frame Relay-Ethernet VLAN 802.1Q bridged interworking:

PE configuration:

```
interface GigabitEthernet0/0/1.10
  encapsulation dot1Q 10
  end
interface Serial0/1/2:0
   no ip address
   encapsulation frame-relay
   no keepalive
   frame-relay interface-dlci 58 switched
   end
 connect FR-ETH1Q Serial0/1/2:0 58 GigabitEthernet0/0/1.10 interworking Ethernet
```

CE configuration: Frame Relay-CE

```
bridge irb
bridge 16 protocol ieee
bridge 16 route ip
interface Serial2/0:0
  no ip address
  encapsulation frame-relay IETF
  no keepalive
interface Serial2/0:0.1 point-to-point
  frame-relay interface-dlci 58
  bridge-group 16
interface BVI16
  ip address 172.18.1.2 255.255.0.0
```

Ethernet-CE

```
interface GigabitEthernet0/0/1.10
  encapsulation dot1Q 10
```

```
  ip address 172.17.2.1 255.255.0.0
```

Example: Configuring Frame Relay-VLAN QinQ Bridged Interworking

The following example shows how to configure Frame Relay-VLAN QinQ bridged interworking:

PE configuration:

```
interface GigabitEthernet0/0/1.11
  encapsulation dot1Q 11 second-dot1q 100
  end
interface Serial0/1/2:0
  no ip address
  encapsulation frame-relay
  no keepalive
  frame-relay interface-dlci 100 switched
  end
connect FR-ETHQinQ Serial0/1/2:0 100 GigabitEthernet0/0/1.11 interworking ethernet
```

CE configuration: Frame-Relay-CE

```
 bridge irb
 bridge 16 protocol ieee
 bridge 16 route ip
 interface Serial2/0:0
   no ip address
   encapsulation frame-relay IETF
   no keepalive
 interface Serial2/0:0.1 point-to-point
   frame-relay interface-dlci 100
   bridge-group 16
 interface BVI16
   ip address 172.18.1.3 255.255.0.0
```

Ethernet-CE

```
interface GigabitEthernet0/0/1.10
  encapsulation dot1Q 10 sec 10
  ip address 172.19.1.1 255.255.0.0
```

Additional References for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
WAN commands	Cisco IOS Wide Area Network Command Reference

RFCs

RFC	Title
RFC 2427	Multiprotocol Interconnect over Frame Relay

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 38: Feature Information for L2VPN Local Switching—Frame Relay-Ethernet/VLAN

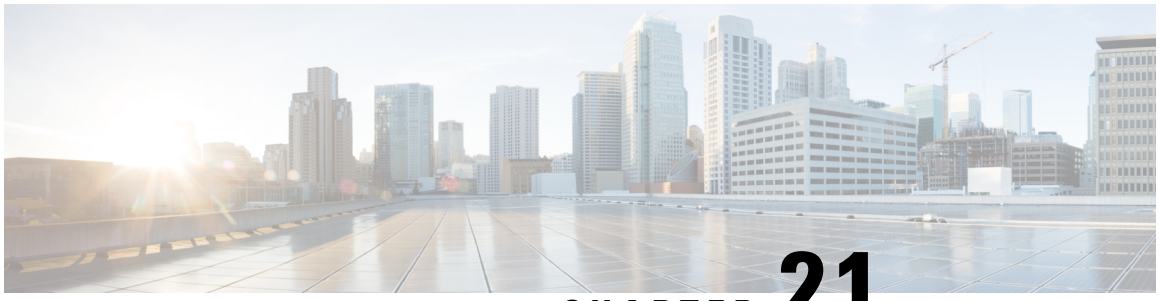
Feature Name	Releases	Feature Information
L2VPN Local Switching—Frame Relay-Ethernet/VLAN		The L2VPN Local Switching—Frame Relay-Ethernet/VLAN feature allows you to switch Frame Relay and Ethernet frames between two interfaces on the same device.



PART IV

WAAS

- mDNS for kWAAS, on page 367



CHAPTER 21

mDNS for kWAAS

The mDNS for kWAAS feature implements the multicast Domain Name System (mDNS) Service Discovery protocol, provides the API layer to the network, and enables Kernel-based Virtual Machine Wide-Area Application Services (kWAAS) devices to easily discover wide-area services advertised by other Cisco IOS devices.

This concept module provides a brief overview of mDNS service discovery for kWAAS and IP networking on mDNS for kWAAS.

- Information About mDNS for kWAAS, on page 367
- Additional References for mDNS for kWAAS, on page 368
- Feature Information for mDNS for kWAAS, on page 369

Information About mDNS for kWAAS

Overview of Service Discovery on mDNS for kWAAS

Software modules within the Cisco software need to be able to discover services of interest. These services are announced via the multicast Domain Name System (mDNS) Service Discovery protocol. The Layer 2 domain DNS helps with the auto-discovery of these services with its capability of multicasting and decentralizing the mDNS service discovery.

mDNS service discovery is performed using IP multicast. IP multicast is the process of sending multiple IP data packets to a receiver in a single transmission.

The multicast Domain Name System (mDNS) for kWAAS implements the mDNS Service Discovery protocol, provides the API layer, and enables kWAAS to easily discover wide-area services advertised by other Cisco IOS devices.

After service discovery, the mDNS Service Discovery protocol lets you retrieve information about the cached mDNS resource records, the different mDNS requests in queue, and the mDNS statistics about the packet relays.

Note The mDNS for kWAAS feature is enabled by default on Cisco devices. There are no configuration tasks, and this feature cannot be disabled.

Overview of IP Networking on mDNS for kWAAS

The mDNS for kWAAS feature utilizes the service discovery protocol, which provides a way to announce and discover services on the local network, thereby enabling wireless Cisco software device clients to access services advertised in a different IP network.

multicast Domain Name System (mDNS) performs DNS queries over IP multicast. mDNS supports zero configuration IP networking. Zero configuration IP networking consists of processes that let you devise an automated IP network, without the need for any intervention by a network administrator. The zero configuration IP networking method does not need any special configuration servers.

As a standard, mDNS uses multicast IP address 224.0.0.251 as the destination address and 5353 as the UDP destination port.



Note The mDNS for kWAAS feature is enabled by default on Cisco devices. There are no configuration tasks, and this feature cannot be disabled.

Additional References for mDNS for kWAAS

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Command List, All Releases*
WAN commands	Cisco IOS Wide-Area Networking Command Reference
Cisco Wide-Area Application Services	• *Cisco Wide Area Application Services Quick Configuration Guide* • *Cisco Wide Area Application Services Configuration Guide*
Wide-Area Networking Overview	*Wide-Area Networking Configuration Guide: Wide-Area Application Services*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for mDNS for kWAAS

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 39: Feature Information for mDNS for kWAAS

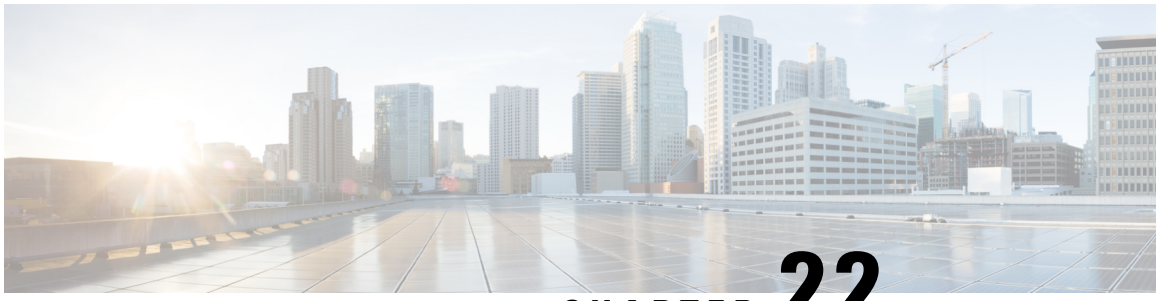
Feature Name	Releases	Feature Information
mDNS for kWAAS	Cisco IOS XE Release 3.9S	The multicast Domain Name System (mDNS) for kWAAS implements the mDNS Service Discovery protocol, provides the API layer to the network, and enables Kernel-based Virtual Machine Wide-Area Application Services (kWAAS) devices to easily discover wide-area services advertised by other Cisco IOS devices. The following commands were introduced or modified: **debug mdns, show mdns cache, show mdns requests, show mdns statistics.**



PART V

Multilink PPP

- Multilink PPP Support, on page 373
- Configuring Multilink PPP Connections for Broadband and Serial Topologies, on page 389
- MLPoE at PTA, on page 437
- Configurable CHAP Challenge Length, on page 447



CHAPTER 22

Multilink PPP Support

First Published: October 2012

Last Updated: August 23, 2016

Multilink Point-to-Point Protocol (MLP) provides support to aggregate the bandwidth of low-speed WAN and broadband links into a single entity, referred to as a bundle interface. A bundle interface is a logical entity that provides a single point in which other features (for example, Quality of Service [QoS]) can be attached. MLP provides incremental bandwidth on demand, by adding additional links to the bundle, as needed. MLP also enables interleaving of latency-sensitive priority traffic with fragmented nonpriority traffic using link fragmentation and interleaving (LFI).

Member links that are a part of an MLP bundle can be bundled across ports on:

- The same shared port adapter (SPA)
- Different SPAs on the same SPA interface processor (SIP)
- Different SPAs on different SIPs

The Cisco IOS XE software supports MLP links for serial (T1, E1, NxDS0) and broadband topologies such as Multilink PPP over ATM (MLPoA), Multilink PPP over Ethernet (MLPoE), Multilink PPP over Ethernet over ATM (MLPoEoA), and Multilink PPP over LNS (MLPoLNS). Additionally, the Cisco IOS XE software allows the device to operate as an L2TP Access Concentrator (LAC), L2TP Network Server (LNS), or PPP Termination and Aggregation (PTA) device.

This document describes the features, limitations, and scaling of MLP on the Cisco ASR 1000 Series Aggregation Services Routers running the Cisco IOS XE software. For information about the configuration and operation of MLP in the Cisco IOS XE software, see the “Configuring Multilink PPP Connections” chapter in the Wide-Area Networking Configuration Guide: Multilink PPP, Cisco IOS XE Release 3S (Cisco ASR 1000) .

- Cisco IOS XE Scaling Limits for MLP Bundles, on page 373
- Information About Multilink PPP Support, on page 378
- Additional References for Multilink PPP Support, on page 385
- Feature Information for Multilink PPP Support, on page 387

Cisco IOS XE Scaling Limits for MLP Bundles

This section lists the scaling limits for MLP bundles in different releases of Cisco IOS XE, in which scaling limits were either introduced or enhanced.

Release 2.2.(0)S

In Cisco IOS XE Release 2.2.(O)S, the MLP feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. MLPoSerial was the first supported transport. In this release, MLP bundles can consist of up to 10 serial links. The bandwidth of each link interface does not have to be the same as the other links in the bundle. The Cisco ASR 1000 Series Aggregation Services Routers support links of types T1, E1, and NxDS0. MLP LFI is fully supported with MLPoSerial in this release.

Release 3.4.(0)S

In Cisco IOS XE Release 3.4.(O)S, the MLP feature was enhanced to enable the Cisco ASR 1000 Series Aggregation Services Routers to act as LAC, LNS, or PTA devices. Support for tunneling bundles between the LAC device and the LNS device was added. In this release, transport between the LAC device and the LNS device is Layer 2 Tunnel Protocol (L2TP). The L2TP tunnels can operate on either 1-Gbps or 10-Gbps interfaces. When ASR 1000 Series Aggregation Services Router acts as an LNS device, it terminates the MLP bundles coming through the L2TP tunnel from the LAC. In this release, support was added for MLP upstream fragment reassembly, but not for MLP downstream fragmentation.

Release 3.7.1S

In Cisco IOS XE Release 3.7.1S, the existing support for the MLP feature in a broadband topology was enhanced. The scaling limits were increased for the Ethernet transports, and downstream fragmentation support was added for the broadband topologies.

In this release, when a Cisco ASR 1000 Series Aggregation Services Router acts as an LNS device, it terminates the MLP bundles coming through the L2TP tunnel from the LAC. The scaling targets mentioned for MLP over broadband are based on RP2/ESP40 and 2RU-VE hardware configurations. The scaling capabilities are less for RP1 and ESP5, ESP10, or ESP20.

The implementation of MLP on a Cisco ASR 1000 Series Aggregation Services Router does not support all the Cisco IOS XE interoperability features.

Release 3.12.(0)S

In Cisco IOS XE Release 3.12.(O)S, the multi-member-link MLPoA or MLPoEoA, including Downstream , is introduced. The scaling limits are increased for the member links in MLPoA or MLPoEoA scenarios.

Table 40: MLP Features and Maximum Scale Numbers, on page 374 shows the maximum scale numbers for various MLP functionalities on the Cisco ASR 1000 Series Aggregation Services Routers.

Table 40: MLP Features and Maximum Scale Numbers

Transport	Maximum Number of Members per Bundle	Maximum Number of Bundles per System	Maximum Number of Member Links per System	Downstream LFI	Upstream Fragment Reassembly	Cisco IOS XE Release
MLPoSerial	10	1232	1232	Yes	Yes	2.2.0S
MLPoA AAL5MUX	1	1000	1000	No	Yes	3.4.0S
MLPoA AAL5MUX	8	4000	4000	Yes	Yes	3.12.0S
MLPoA AAL5SNAP	1	1000	1000	No	Yes	3.4.0S

Transport	Maximum Number of Members per Bundle	Maximum Number of Bundles per System	Maximum Number of Member Links per System	Downstream LFI	Upstream Fragment Reassembly	Cisco IOS XE Release
MLPoA AAL5SNAP	8	4000	4000	Yes	Yes	3.12.0S
MLPoE	1	4000	4000	No	Yes	3.4.0S
MLPoE	8	4000	4000	Yes	Yes	3.7.1S
MLPoEoA AAL5SNAP	1	1000	1000	No	Yes	3.4.0S
MLPoEoA AAL5SNAP	8	4000	4000	Yes	Yes	3.12.0S
MLPoEoQinQ	1	4000	4000	No	Yes	3.4.0S
MLPoEoQinQ	8	4000	4000	Yes	Yes	3.7.1S
MLPoEoVLAN	1	4000	4000	No	Yes	3.4.0S
MLPoEoVLAN	8	4000	4000	Yes	Yes	3.7.1S
MLPoLNS	1	4000	4000	No	Yes	3.4.0S
MLPoLNS	8	4000	4000	Yes	Yes	3.7.1S

Restrictions for MLP over Serial Interfaces

The following restrictions apply to MLP over Serial Interfaces:

- The MLP over Serial Interfaces feature supports a maximum of ten member links per bundle. The member links can be any combination of T1/E1 or fractional T1s/E1s (for example, NxDS0). Member-link interfaces no faster than E1 speeds (DS0, T1, and E1) are only supported in the MLP over Serial Interfaces feature. For better MLP performance, all the member links in a bundle must be of the same bandwidth.
- Member links in a bundle cannot be of different encapsulation types.
- You cannot manually configure the bandwidth of an MLP bundle by using the bandwidth command on the multilink interface. The bandwidth of an MLP bundle is managed based on the aggregate bandwidth of all the active member links on the bundle. As the links are dynamically added or removed from an MLP bundle, the bandwidth is updated to reflect the aggregate of the active links. The bandwidth can be rate limited by applying an hierarchical QoS policy on the multilink interface and applying a shaper to the parent class-default class.
- MLP over Frame Relay is not supported; only MLP over Serial PPP link is supported. Customers who require multilink support in a frame relay environment can use the Multilink Frame Relay (MLFR-FRF.16) feature.
- The legacy IOS compression feature compress [mppc | stac | predictor] is not supported.
- LFI is supported on MLP bundles with any number of links in the bundle. However, when using a bundle with more than one member link, the order of the priority packets (PPP encapsulated) is not guaranteed. Priority-packet distribution is handled in a manner similar to IP per-packet load sharing. MLP guarantees

nonpriority packet ordering that manages reordering at the peer device, based on the MLP packet sequence number.

- Order issues with the LFI multiple-member link in case of priority traffic can be addressed in some platforms using Multiclass Multilink Protocol (MCMP-RFC 2686), which is an extension of the MLP. The Cisco ASR 1000 Series Aggregation Services Routers do not support MCMP.
- Only the MLP long-sequence number format is supported for the packet header format option.
- PPPoE is not supported on SVI interface for Cisco 1000 Series Integrated Services Routers and Cisco 4000 Series Integrated Services Routers.

Restrictions for MLP over Ethernet at PTA and LAC

The following restrictions apply to MLP over Ethernet at PTA and LAC:

- MLPoE using EtherChannel is not supported.
- For MLP virtual access bundles, the default Layer 3 (that is IP and IPv6) maximum transmission unit (MTU) value is 1500. For more information about MTU, see the **MTU** section.
- For MLPoE PTA variations (MLPoE, MLPoVLAN, and MLPoQinQ), the default bandwidth of the member-link session is 1 Gbps instead of the data rate communicated by the DSLAM to the PTA router. If a bandwidth statement is added to the virtual template, the bandwidth is applied to the bundle instead of the member link. This is not the desired behavior. (To define the data rate of an MLPoE PTA-type bundle, apply a QoS policy on the bundle session that includes a parent shaper on the class-default class with an explicit data rate defined. Do not use the shape percent command in this parent shaper because the shape percent command uses the default data rate of 1 Gbps as the base rate for percent calculation. However, the percent-based rates can be defined in the child (nested) policy, if an hierarchical policy is being defined.
- If the DSLAM between the CPE and PTA communicates the link rate through the PPPoE dsl-sync-rate tags (Actual Data-Rate Downstream [0x82/130d] tag), the PTA device passes this data to the RADIUS server, but the Cisco ASR 1000 Series Aggregation Services Routers do not act upon it. The data rate of the session remains as described in the previous list item.

Restrictions for MLP over ATM at PTA and LAC

The following restrictions apply to MLP over ATM at PTA and LAC:

- ATM Autosense is supported to allow the dynamic selection of MLPoA or MLPoEoA.
- For ATM, the link-level bandwidth is a part of the ATM Permanent Virtual Circuits (PVC) configuration based on the unspecified bit rate (UBR) or variable bit rate (VBR) configurations. The bundle bandwidth is the aggregate of the member-link session bandwidth.



Note The MLP over Ethernet over ATM at PTA and LAC has the same restrictions as the MLP over ATM at PTA and LAC.

Restrictions for MLP at LAC

In case of MLP over LNS (Ethernet) LAC switching, the MLP member-link session and the packet payload is transparent at the LAC device because it does not terminate the MLP session or the bundle interface. Hence, the LAC device does not bind the number of member-link sessions associated with a bundle. Similarly, the LFI functionality is transparent at the LAC device because the traffic is switched or passed through traffic.

Restrictions for MLP over LNS

The following restrictions apply to MLP over LNS:

- MLPoLNS bundles are supported with only Ethernet as the trunk between the LAC and LNS.
- Layer 2 Tunnel Protocol (L2TP) over IPsec is not supported.
- QoS (other than downstream Model-F shaping) on interfaces and tunnels towards the customer premise equipment (CPE) is not supported.
- When the CPE client initiates the PPP LCP connection, the multilink negotiation included as part of the LCP negotiation may fail if the LAC has not yet established connection with the LNS (which is typically the case). The LNS renegotiates the Multilink LCP options with the CPE client when the LAC initiates the connection to the LNS. (To allow this renegotiation of LCP options, the **lcp renegotiation always** command must be configured in the VPDN group at the LNS).
- Although per-packet load balancing is not supported, the configuration is not blocked and the functionality is operational (but not tested). Per-packet load balancing cannot be used with MLPoLNS because MLPoLNS requires a single-path per-destination IP address.
- Unlike the MLP over Serial mode or the MLP PTA mode, packets may traverse several network hops between the CPE and LNS devices in an MLPoLNS network. As a result of this multihop topology, even on a single-link bundle, MLP encapsulated packets may arrive at the receiver in an out-of-order state. Hence, the MLPoLNS receiver operates in a loose, lost-fragment detection mode. In this mode, if an MLP fragment is lost, the received MLP waits for a short time to receive the lost fragment. In addition, the MLP receiver limits the amount of out-of-order MLP data received before the fragment is declared lost. In Cisco IOS XE software, the default timeout value is 1 second. This may create problems in an environment with high packet loss and scaled MLP configurations because it requires the receiver to potentially buffer large amounts of data for each MLP bundle. Since the buffer space that is available is a finite resource, worst-case depletion of buffers can bleed over and begin affecting packet buffering on other MLP bundles. (The MLP lost-fragment timeout can be configured on the multilink virtual template interface using the **ppp timeout multilink lost-fragment** (*seconds*) (*milliseconds*) configuration command).

By default, in MLPoLNS, the Cisco IOS XE software informs the MLP that packets may arrive out of order. This works well for upstream traffic, but does not address the order issue at the peer CPE device. The peer CPE device should also be configured to allow for receipt of out-of-order packets. In Cisco devices, this can be managed by configuring the **ppp link reorders** command at the bundle interface.

- When the Cisco ASR 1000 Series Aggregation Services Routers function as both a PTA device and an LNS device simultaneously, locally terminated member links (PTA) and member links that are forwarded from the LAC are not supported within the same bundle.

Restrictions for Broadband MLP at PTA and LNS

The following restrictions apply to all variations of broadband MLP at PTA and LNS modes:

- When defining an MLP bundle with multiple member-link sessions, we recommend that all the member-link sessions utilize the same physical interface or subinterface. If other broadband sessions are sharing the same interface, ensure that all the member-link sessions utilize the same physical interface or subinterface.
- The following issues might occur because of splitting links across separate physical interfaces or subinterfaces:
  - MLP is a sequenced protocol and all the packets and fragments must be reordered and reassembled at the receiver, based on the MLP sequence number before the receiver forwards them. In such a scenario, packets traversing separate physical interfaces may cause additional packet latency disparity between links due to transmission delays and other issues associated with using multiple physical paths. The reordering and reassembly processing may require additional MLP buffering at the receiver.
  - MLP on the Cisco ASR 1000 Series Aggregation Services Routers performs congestion management of the MLP bundle based on the congestion state of the member-link sessions that make up the bundle. If member-links are distributed across multiple interfaces and sufficient congestion is detected in one or more member links, the bundle may be back pressured due to the congestion even if all the links in the bundle are not congested. By keeping all the links on the same physical interface or subinterface, the chance of back pressure due to one link being congested is reduced.

Information About Multilink PPP Support

The Multilink PPP feature provides the load-balancing functionality over multiple WAN links, while providing multivendor interoperability, packet fragmentation, proper sequencing, and load calculation for both inbound and outbound traffic. Cisco implementation of MLP supports the fragmentation and packet-sequencing specifications described in RFC 1990.

Some Cisco IOS platforms use the **interface multilink** command for both MLP over Serial and MLP over ATM (MLPoA) to configure multilink bundle interfaces. On the Cisco ASR 1000 Series Aggregation Services Routers, multilink bundle interfaces are configured using the **interface multilink** command for MLP over Serial and the **interface Virtual-Template** command for MLPoA.

On the Cisco ASR 1000 Series Aggregation Services Routers, all broadband MLP configurations use the **interface Virtual-Template** command to define the multilink bundle configuration. A virtual access interface is created dynamically from the virtual template when the session is negotiated with the peer device.

Quality of Service

QoS refers to the ability of a network to provide improved service to selected network traffic over various underlying technologies, including Frame Relay, ATM, Ethernet and 802.1 networks, SONET, and IP-routed networks. In particular, QoS features provide improved and more predictable network service.

For serial deployments, QoS is applied to an MLP bundle using the **multilink** configuration command. For broadband deployments, QoS is applied to an MLP bundle using the **virtual-template** command. When a router dynamically creates the virtual access interface from the virtual template, the QoS policy is applied to the corresponding bundle.

QoS is characterized by the following features and restrictions:

- To rate limit a broadband MLP bundle session, use a hierarchical QoS (HQoS) policy with a parent shaper in the class-default class.

- The Cisco ASR 1000 Series Aggregation Services Routers support HQoS queuing only in the egress (output) direction, and not in the ingress direction.
- The Cisco IOS XE software supports Model-F QoS with MLP. Model-F QoS on the L2TP tunnel is not supported on the Cisco ASR 1002-X Router and the FP100 line card.
  - In Cisco IOS XE Release 3.7.1S, support was added for Model-F QoS on the L2TP tunnel when the device acts as an LNS. A parent shaper policy can be applied to the physical subinterface that connects the LNS to the LAC device. This enables the shaping of the aggregate traffic going downstream to the LAC device.
  - If a Model-F shaper is attached to the LAC-facing interface after the sessions are established through that interface, the sessions must be bounced to handle the priority traffic appropriately.
- In Cisco IOS XE Release 3.4S, the **shape average** *shape-rate* **account user-defined** *offset* **atm** command supports only the broadband MLP interface and not the MLP over serial interface. The range for the **offset** argument is from -48 to 48 bytes. In Cisco IOS XE Release 3.6S, the **shape average** *shape-rate* **account user-defined** *offset* **atm** command supports MLP over serial interface.
- ATM cell loss priority (CLP) Match (classification) and Set (marking) are not supported with broadband MLP.
- When packets transit the MLP transmit path, they are subject to two separate stages of queuing. The first stage is at the MLP bundle interface, where QoS may be applied, and the second one is at the MLP member-link interface. At the MLP bundle interface, the packets are processed according to the applied QoS policy. Packets classified as priority are given preferential treatment over nonpriority traffic. For the priority classification to be honored at the MLP member-link interface, the bundle must have ppp multilink interleave enabled. Interleaving allows a packet to be queued to a separate priority queue at the member-link. If interleaving is not enabled on the bundle, the priority packet is placed in the member link session default queue and the knowledge that it is a priority packet will be lost. This is especially important if there are other PPP or MLP sessions sharing the same physical interface or subinterface. Without interleaving, priority traffic on the other sessions are given preferential treatment over the MLP priority packets that were reclassified as nonpriority packets at the MLP member-link queuing stage. For additional information on interleaving, see the *Downstream LFI* section.

Multilink PPP Packet Overhead Accounting for Shaping and Policing

On the Cisco ASR 1000 Series Aggregation Services Routers, Multilink PPP adjusts the packet length presented for shaping and policing to include the additional Layer 2 overhead added by Multilink PPP. For MLP over Serial, overhead accounting includes the MLP and PPP Layer 2 overhead. For Broadband MLPs such as MLPoE, MLPoEoVLAN, MLPoEoQinQ, MLPoEoA, MLPoA, and MLPoLNS, overhead accounting includes the MLP, PPP, Ethernet, ATM, and L2TP (LNS) Layer 2 overhead. If the output interface is ATM, such as the MLPoA or MLPoEoA, the Cisco ASR 1000 Series Aggregation Services Routers also take into account the ATM Cell overhead for the shaper. The ATM Cell overhead is not accounted for policing.

Shaping and policing overhead accounting does not include the additional overheads added by a SPA such as, Ethernet CRC, preamble, IPG, serial interface CRC, start of packet (SOP) delimiter, end of packet (EOP) delimiter, and serial-bit stuffing (the only exception being the ATM Cell overhead for the shaper referred to earlier). The overhead added by a SPA can be included in the shaper using the QoS **shape accounting user-defined** option.

If you do not define a QoS shaper for the multilink bundle interface, a default shaper is applied to the bundle based on the aggregate bandwidth of all the links that make up the multilink bundle. The information contained

in this section applies to both the default shaper and a QoS user-defined shaper, which the user may explicitly configure and apply to a multilink bundle.

The priority packets that are interleaved are sent PPP encapsulated and the MLP Layer 2 overhead is not included because MLP encapsulation is not included in these packets. During overhead accounting for link fragmentation, overhead accounting calculations are performed prior to the actual link fragmentation and link selection for Multilink PPP load balancing.

If all the member links in the corresponding multilink bundle use the same fragment size, the number of fragments are calculated and the overhead is adjusted to include the additional per-fragmentation Layer 2 header overhead for the shaper and policer. If one or more links in the bundle use different fragment sizes, the number of fragments cannot be calculated with 100 percent accuracy because link selection for load balancing and fragment size is not known until QoS processing is completed at the bundle level (after shaping and policing). For links with unequal fragment size, a best effort attempt is made using the largest link fragment size on the bundle. By using the largest fragment size, MLP avoids undersubscribing the member-link interfaces. If the links become oversubscribed, MLP will backpressure the bundle to avoid sustained oversubscription of the member links.

In Cisco IOS XE Release 3.4S on the Cisco ASR 1000 Series Aggregation Services Routers, support for shaping and policing overhead accounting was added for Broadband Multilink PPP. In addition, support was added for the Shape User-Defined Overhead Accounting feature using the following QoS command:

shape [average | peak] mean-rate [*burst-size***] [***excess-burst-size***] account {{{qinq | dot1q} {aal5 | aal3} {subscriber-encapsulation}} | {***user-defined offset* **[atm]}}**

This command enables you to include the additional overhead added by a SPA using the **user-defined** option. For example, the Ethernet SPA adds an additional 24 bytes per packet so that a user-defined value of 24 covers Ethernet IPG (12) + Preamble (8) + CRC32 (4). Another interesting scenario is when deploying MLPoLNS in an ATM topology. The physical link between the LNS and the LAC is Ethernet, and the physical link between the LAC and the CPE is ATM. In such a scenario, you can add the **atm** keyword to include the ATM Cell overhead between the LAC and the CPE.

In Cisco IOS XE Release 3.6S, shaping and policing overhead accounting support was added for Serial Multilink PPP and Multilink PPP LFI.

For more information on shaping and policing, see the IOS XE Ethernet Overhead Accounting documentation at:
http://www.cisco.com/en/US/docs/ios-xml/ios/qos_plcshp/configuration/xe-3s/qos-plcshp-ether-ohead-actg.html

Downstream Model-F Shaper on LNS

From Cisco IOS XE Release 3.7.1S, Model-F downstream shaping support for MLPoLNS is available to the Cisco ASR 1000 Series Aggregation Services Routers when these routers function as an LNS device.



Note Model-F downstream shaping for MLPoLNS is not supported on the Cisco ASR 1002-X Router and the FP100 line card.

This section provides an example of a Model-F policy with a parent shaper policy attached to a VLAN interface on the LNS device. The VLAN interface is used for the L2TP tunnel between the LAC device and the LNS device. The following configuration example shows an aggregate shaper applied to a VLAN, which shapes all the MLP sessions going downstream to the LAC device:

```
policy-map lns_downstream_shaper_out
```

```
class class-default
shape average 5000000
interface GigabitEthernet0/1/0.2
encapsulation dot1Q 2
ip address 90.0.0.1 255.255.255.0
service-policy output lns_downstream_shaper_out
```



Note Model-F QoS allows a parent shaper on the class-default class by using a flat policy. No additional QoS functionalities are supported in the Model-F policy.

Bandwidth

The interface-level **bandwidth** command must not be used to define the bandwidth at the bundle level on the virtual template interface or the multilink interface. By default, the bundle bandwidth is the aggregate of the bandwidth of the individual member links that make up the bundle.

For ATM, the link-level bandwidth is part of the ATM Permanent Virtual Circuits (PVC) configuration based on the unspecified bit rate (UBR) or variable bit rate (VBR) configurations. The member-link bandwidth cannot be set for an MLPoE session on a PTA device. To define the bandwidth for an MLPoE-type bundle on a PTA device, a QoS policy must be applied to the bundle interface that shapes the bundle bandwidth at the class-default class with a parent shaper.

In PPPoE and MLPoE broadband networks, the DSL access multiplexer (DSLAM) placed between the customer premises equipment (CPE) and LAC or PTA, inserts a PPPoE vendor tag. This tag includes information such as, media rate, characteristics, and identification pertaining to the circuit or session.

For more information about Ethernet-based networks, see DSL Forum TR-101 Migration to Ethernet-Based DSL Aggregation April 2006 at:

http://www.broadband-forum.org/technical/download/TR-101.pdf
http://www.broadband-forum.org/technical/download/TR-101.pdf

The PTA passes media-rate information to the RADIUS server for selecting an appropriate QoS policy to the bundle session based on the reported bandwidth. In the context of MLP over LNS, the LAC passes media-rate information to both the RADIUS server and the LNS router. The LNS router uses the media-rate information to define the bandwidth of the corresponding member-link session. If the upstream connection at the LAC is MLPoE, MLPoEoVLAN, or MLPoEoQinQ, the DSLAM may provide the media rate information to the LAC. If the DSLAM does not provide the media rate, the member-link session bandwidth can be configured using the **l2tp tx-speed** *rate* and **l2tp rx-speed** *rate* commands within the **vpdn-group** configuration command or downloaded from the RADIUS server using the l2tp-tx-speed and l2tp-rx-speed attributes.

MTU

For MLP Virtual Access bundles (IP and IPv6), the default Layer 3 MTU value is 1500. When the MLP bundle's member links are Ethernet, as in MLPoE, MLPoEoVLAN, and MLPoEoQinQ, the default MTU value of 1500 may cause an issue when sending IP packets that are close to this size.

For example, when a router sends a 1500-byte IP packet over MLPoE, the actual packet size transmitted is 1528: 14 (Ethernet header) + 8 (PPPoE header) + 6 (MLP header) + 1500 (IP) = 1528. The peer router drops the incoming packet as a **giant** because it exceeds the default expected maximum packet size.

The 1500-byte MTU size does not take into account any PPPoE or MLP header overhead, and hence, causes packets greater than 1493 bytes to be dropped by the peer. To address this issue, perform one of the following tasks:

- Lower the MTU on the MLP bundle to 1492.
- Increase the MTU on the Ethernet interface to 9216, the maximum MTU size supported on Cisco ASR 1000 Series Aggregation Services Routers.



Note In Cisco ASR 1000 Series Route Processor 1 (RP1), 2RU, and 2RU-Fixed chassis, the MTU size for the Management Ethernet interface (interface gigabitethernet 0) is up to 2370 bytes.

Downstream LFI

Although LFI is thought of as a single feature, it is actually two independent features within MLP. MLP link fragmentation allows larger packets to be Layer 2 fragmented by MLP, and the fragments to be distributed across the various member links in the MLP bundle. These fragments are MLP encapsulated and sequenced. These fragments are then collected, reordered, and reassembled at the peer termination point for the MLP bundle interface.



Note For more information about interleaving with QoS, see the *Quality of Service* section..

Interleaving enables you to reduce transmission delay on delay-sensitive voice, video, and interactive application data by interleaving it with the MLP fragments. When interleaving is configured, the packets on the bundle interface that QoS classifies as priority packets are interleaved. These priority packets are PPP encapsulated and interleaved with the MLP-encapsulated fragments or packets. When the peer router receives the PPP packets, they can be immediately forwarded, whereas, the received MLP-encapsulated packets have to be reordered and reassembled before being forwarded. While link fragmentation and interleaving can be configured on any multilink bundle, this LFI functionality is beneficial only on bundles of 1 Mbps or less. Packet transmission delays of higher bandwidth bundles are such that QoS prioritization of priority traffic should be sufficient to guarantee preferential treatment of the priority traffic without the need for LFI.

One downside of interleaving is that when there are two or more links in an MLP bundle, the order of the PPP-encapsulated packets cannot be guaranteed. In most applications sending data, such as, voice, video, and Telnet, this is not an issue because the gap between the packets on a given flow is large enough that the packets must not pass each other on the multiple links in the bundle. Since the order cannot be guaranteed for the priority PPP-encapsulated packets that are interleaved, IP Header Compression (IPHC) is skipped on any packet that is classified as priority-interleaved packet. IPHC continues to occur for nonpriority packets that are sent as MLP encapsulated because MLP guarantees reordering before the packets are forwarded to IPHC.

The Multi-Class Multilink Protocol (MCMP) (RFC-2686) addresses the issues related to ordering of priority-interleaved packets. Currently, the MCMP is not supported on the Cisco ASR 1000 Series Aggregation Services Routers.

MLP LFI must be configured on the Cisco ASR 1000 Series Aggregation Services Routers to enable LFI.

In the context of interface multilink or interface virtual template, use any of the following commands to enable link fragmentation:

- **ppp multilink fragment delay** (delay in milliseconds)
- **ppp multilink fragment delay** (maximum fragment size, in bytes)
- **ppp multilink interleave**

For MLP using serial links, link fragmentation can also be enabled by configuring the **ppp multilink fragment size** (maximum fragment size, in bytes) command on the member-link serial interface.

If the MLP bundle has only one active member link and interleaving is not enabled, MLP fragmentation is disabled. In addition, all the packets are sent PPP encapsulated instead of MLP encapsulated. When a second link in the bundle becomes active or interleaving is enabled, MLP and fragmentation is enabled.

If the **ppp multilink interleave** command is not configured, only MLP link fragmentation is enabled. To enable interleaving, you must also configure the **ppp multilink interleave** command at the interface multilink level or the interface virtual template level. In addition to configuring interleaving as indicated here, you must also define a QoS policy with one or more priority classes, and attach the QoS to this interface using the **service-policy output** *policy-map-name* command. This command classifies the priority traffic, that is interleaved by the MLP.

See the QoS and LFI configuration examples in the "Configuring Multilink PPP Connections" chapter in the *Wide-Area Networking Configuration Guide: Multilink PPP*.

When configuring MLP fragmentation on the various Cisco platforms, the functionality of MLP fragmentation and interleaving support on the various platforms may differ. This section explains the configuration options and their interpretation in the context of the Cisco ASR 1000 Series Aggregation Services Routers.

Based on the values of the MLP fragmentation configuration commands, the MLP feature calculates two values that are used during MLP fragmentation: link weight and maximum fragment size. These parameters are calculated for each member link in the bundle.

First, a link weight must be determined for each member link. The link weight indicates the number of bytes, and the MLP uses this value to balance the data amongst the links in the bundle. This parameter is especially important when the links in a bundle are of unequal bandwidth. The link weight is based on a combination of the bandwidth of the member link and the PPP multilink fragment delay value. If you do not configure the fragment delay value, a default delay value of 30 milliseconds is used:

Link Weight = (Member Link Interface Bandwidth in bps/8) * Fragment Delay



Caution Configuring the fragment delay to a smaller value results in smaller fragment size because the fragment delay value determines the default fragment size on the member link. This, in turn, implies loss of bandwidth due to the added Layer 2 header overhead. This is important for broadband MLP, which can have Layer 2 headers of 4 to 58 bytes in length.

The default maximum fragment size must be calculated per member link. The default maximum fragment size used will be the lesser value obtained from either of the following calculations:

- Link Weight – Multilink PPP + PPP Header Overhead (8)
- Interface MTU – Multilink PPP Header Overhead (4)

After the default maximum fragment size is calculated, if you have configured the **ppp multilink fragment size** (maximum) command at the multilink, virtual template, or serial interface level, the default maximum fragment size is compared against the configured maximum value and is capped accordingly. If the fragment

size is configured at the serial interface level and the multilink interface level, the serial interface configuration takes precedence.

MLP Fragmentation Model

Earlier, some Cisco platforms supported a legacy MLP fragmentation model that was enabled by default if all the following criteria were met:

- Two or more active member links exist in the bundle.
- All the member links have equal bandwidth.
- No other form of multilink fragmentation or interleave commands are configured on the bundle or member-link interface.

In the legacy model, there were many instances when fragmentation was enabled by default without users being aware that it was configured. In addition, packets of moderate length could be fragmented. This did not provide the expected throughput on the bundle due to the added packet Layer 2 overhead introduced by MLP fragmentation.

On the Cisco ASR 1000 Series Aggregation Services Routers, this model of MLP fragmentation was supported until Cisco IOS XE Release 3.7.0. Effective from Cisco IOS XE Release 3.7.1, the Cisco ASR 1000 Series Aggregation Services Routers do not support this mode of MLP fragmentation. Therefore, you must now explicitly configure the multilink fragmentation or interleaving to enable MLP fragmentation.

Effective from Cisco IOS XE Release 3.7.1, the following MLP configuration commands are ignored by the Cisco ASR 1000 Series Aggregation Services Routers:

- **ppp multilink fragment disable**
- **ppp multilink fragment maximum** *maximum number of fragments per packet*

IP Type of Service Reflect

Effective from Cisco IOS XE Release 3.7.(0)S, support for the IP Type of Service (ToS) Reflect feature was added on the VPDN group or VPDN template for the L2TP tunnel when the Cisco ASR 1000 Series Aggregation Services Routers act as LNS devices for broadband MLP sessions. Later, this feature was also added to the following maintenance releases: Cisco IOS XE 3.4.2, 3.5.1, and 3.6.2.

The IP Type of Service (ToS) Reflect feature allows the IP header ToS value from the inner IP header to be reflected in the ToS of the outer L2TP IP header.



Caution To prevent MLP packet reordering and fragment or packet holes, the ToS data should not be used to reclassify and requeue or drop packets at the LAC. Any drops or reordering of MLP packets may cause MLP reordering or reassembly delays and additional packet loss in the receiving CPE device.

The following example shows how to configure IP ToS reflect:

```
vpdn-group vpdn-1
accept-dialin
protocol l2tp
virtual-template 1
session-limit 100
terminate-from hostname VPDN-1
lcp renegotiation always
```

```
no l2tp tunnel authentication
ip tos reflect
```

IP Tunnel Marking

Effective from Cisco IOS XE Release 3.7.1, support was added for setting the ToS value in the outer L2TP IP header using the QoS set tunnel action or the policer set tunnel action.

The following configuration options of the set actions are supported when applied to the output QoS policy of the multilink virtual template interface. This functionality is not supported in the Model-F QoS policy attached to the member-link parent subinterface.

- set ip dscp tunnel xx
- set ip prec tunnel xx
- set dscp tunnel xx
- set prec tunnel xx
- police set-dscp-tunnel-transmit xx
- police set-prec-tunnel-transmit xx

The following example shows how to set the ToS value using the police set-prec-tunnel-transmit option:

```
policy-map ppp
class class-default
police cir 4000000 conform-action set-prec-tunnel-transmit 3
Set action example:
policy-map ppp
class gold
set ip prec tunnel 1
```

Unsupported Features

The Cisco ASR 1000 Series Aggregation Services Routers do not support the following MLP features:

- In-Service Software Upgrade (ISSU) and Stateful Switchover (SSO) for MLP bundles
- The broadband L4 Redirect feature and the Intelligent Services Gateway feature
- Per-user firewall
- Lawful intercept
- MLP with MPLS-TE FRR
- Change of Authorization (CoA)
- Layer 2 input QoS classification
- The Multiclass Multilink Protocol (MCMP) RFC 2686 extension to LFI
- Per-user Access Control Lists (ACLs) applied through the RADIUS server are not supported. However, ACLs applied through the virtual template definition for the bundle are supported.
- Only the MLP long-sequence number format is supported for the packet header format option.

Additional References for Multilink PPP Support

The following sections provide references related to the Multilink ppp protocol connections.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Configuring Multilink PPP Connections for Broadband and Serial Topologies	Configuring Multilink PPP Connections for Broadband and Serial Topologies
MLP	Wide-Area Networking Configuration Guide: Multilink PPP, Cisco IOS XE Release 3S
PPP commands	Cisco IOS Dial Technologies Command Reference
Broadband Configuration	Cisco IOS XE Broadband and DSL Configuration Guide
Cisco IOS Configuration Fundamentals	Cisco IOS Configuration Fundamentals Command Reference

Standards

Standard	Title
None	—

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use the Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1990	The PPP Multilink Protocol (MP)
RFC 2686	The Multi-Class Extension to Multi-Link PPP

Technical Assistance

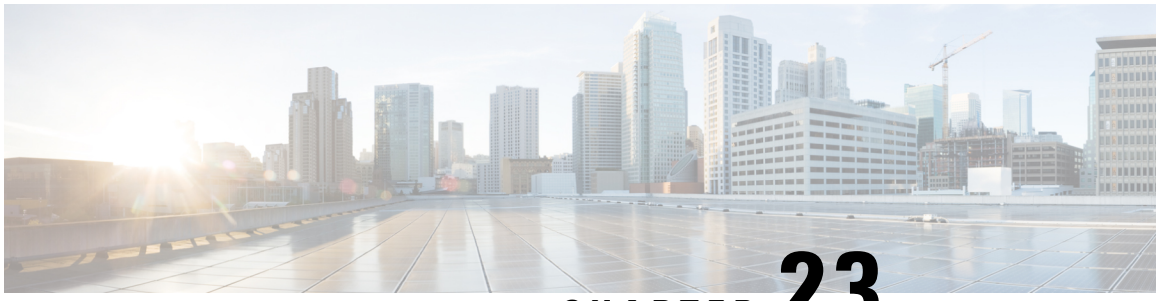
Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Multilink PPP Support

Table 41: Feature Information for Multilink PPP Support for Cisco ASR 1000 Series Routers

Feature Name	Releases	Feature Information
MLPoSerial	Cisco IOS XE Release 2.2.0S	In Cisco IOS XE Release 2.2.(0)S, support for MLPoSerial was introduced on the Cisco ASR 1000 Series Aggregation Services Routers.
MLPoBroadband with single-link bundles	Cisco IOS XE Release 3.4.0S	In Cisco IOS XE Release 3.4.(0)S, support for MLPoBroadband with single-link bundles was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. Support for MLP upstream fragment reassembly was also added, but not for downstream fragmentation.
MLPoLNS and MLPoE/MLPoVLAN/MLPoQinQ with up to 8 links per bundle	Cisco IOS XE Release 3.7.1S	In Cisco IOS XE Release 3.7.1S, support for MLPoLNS, MLPoE, MLPoVLAN, and MLPoQinQ with up to eight links per bundle was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. Support for downstream MLP LFI for all broadband MLPs was also added.
MLPoLNS Model F and IP Tunnel Marking	Cisco IOS XE Release 3.7.1S	In Cisco IOS XE Release 3.7.1S, support for MLPoLNS Model F and IP Tunnel Marking was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. **Note** Model-F downstream shaping for MLPoLNS is not supported on the Cisco ASR 1002-X Router and the FP100 line card.

Feature Name	Releases	Feature Information
Multi member-link MLPPPoA/MLPPPoEoA (including DS LFI)	Cisco IOS XE Release 3.7.12S	In Cisco IOS XE Release 3.7.12S, support for Multi member-link MLPPPoA/MLPPPoEoA (including DS LFI) was introduced on the Cisco ASR 1000 Series Aggregation Services Routers.



CHAPTER 23

Configuring Multilink PPP Connections for Broadband and Serial Topologies

This module describes how to configure Multilink PPP over broadband and serial interfaces. Configuring Multilink PPP over broadband includes configuring Multilink PPP over ATM (MLPoA), Multilink PPP over Ethernet (MLPoE), Multilink PPP over Ethernet over ATM (MLPoEoA), Multilink PPP over Queue-in-Queue (MLPoQinQ), and Multilink PPP over VLAN (MLPoVLAN).

- Restrictions for Multilink PPP Connections for Broadband and Serial Topologies, on page 389
- Information About Multilink PPP Connections for Broadband and Serial Topologies, on page 390
- How to Configure Multilink PPP Connections for Broadband and Serial Topologies, on page 401
- Configuration Examples for Multilink PPP Connections for Broadband and Serial Topologies, on page 426
- Additional References for Multilink PPP Connections for Broadband and Serial Topologies, on page 435
- Feature Information for Multilink PPP Connections for Broadband and Serial Topologies, on page 436

Restrictions for Multilink PPP Connections for Broadband and Serial Topologies

Multilink PPP over Broadband Restrictions

This section lists the common limitations and caveats for all broadband Multilink PPP models supported by Cisco software. For information about basic broadband restrictions, limitations, and caveats, see the *Broadband Access Aggregation and DSL Configuration Guide*.

- Some Cisco broadband Multilink PPP implementations support the creation of a Multilink PPP bundle for some broadband configurations either through a virtual template interface or through an interface multilink. Cisco software supports virtual template interfaces only for broadband Multilink PPP bundle types. Interface multilink and multilink groups are supported on Multilink PPP over serial interfaces (that is, nonbroadband Multilink PPP).

Note A virtual access interface (VAI) will be created for a session that is based on a virtual template configuration.

- Cisco software supports a maximum of 4096 virtual template interfaces. In many cases, a single common virtual template configuration may be used to create multiple bundle VAI instances, especially for single-link bundle sessions.
- Depending on your release, there may be a limit of one link per Multilink PPP bundle. The **ppp multilink links maximum** command should be configured on the virtual template to ensure that requests for additional links in a bundle are rejected.
- Cisco In-Service Software Upgrade (ISSU) and stateful switchover (SSO) for broadband Multilink PPP sessions are not supported.

QoS Restrictions

- Quality of service (QoS) is supported only on a Multilink PPP bundle. QoS is not supported on PPP sessions that make up the Multilink PPP bundle. Multilink PPP does not function properly if queuing policies are attached to PPP sessions that make up the Multilink PPP bundle.
- QoS Bandwidth Remaining Ratio (BRR) is not supported with Multilink PPP. BRR will require QoS policies to be applied to PPP sessions that make up the Multilink PPP bundle, which will cause Multilink PPP to not function properly.

Information About Multilink PPP Connections for Broadband and Serial Topologies

Multilink PPP

The Multilink PPP feature provides load balancing functionality over multiple WAN links while providing multivendor interoperability and support for packet fragmentation, proper sequencing, and load calculation on both inbound and outbound traffic. The Multilink PPP feature supports the fragmentation and packet sequencing specifications described in RFC 1990.

Multilink PPP allows packets to be fragmented and fragments to be sent at the same time over multiple point-to-point links to the same remote address. Multiple links come up in response to a defined dialer load threshold. The load can be calculated on inbound or outbound traffic, as required, for the traffic between specific sites. Multilink PPP provides bandwidth on demand and reduces transmission latency across WAN links.

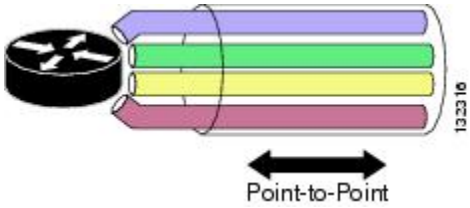
Multilink PPP can work over synchronous and asynchronous serial type of single or multiple interfaces that have been configured to support both dial-on-demand rotary groups and PPP encapsulation.

Multilink PPP Bundles

Multilink PPP combines multiple physical links into a logical bundle called a Multilink PPP bundle. A Multilink PPP bundle is a single, virtual interface that connects to the peer system. Having a single interface (Multilink PPP bundle interface) provides a single point to apply hierarchical queueing, shaping, and policing to traffic flows. Individual links in a bundle do not perform any hierarchical queueing. None of the links have any knowledge about the traffic on parallel links. Hierarchical queueing and quality of service (QoS) cannot be applied uniformly to the entire aggregate traffic between a system and its peer system. A single, virtual interface

also simplifies the task of monitoring traffic to the peer system (for example, all traffic statistics run on one interface).

Figure 45: Multilink PPP Bundle



Multilink PPP Bundles and PPP Links

Multilink PPP works with fully functional PPP interfaces. A Multilink PPP bundle can have multiple links connecting peer devices. These links can be serial links or broadband links (Ethernet or ATM). As long as each link behaves like a standard serial interface, mixed links work properly in a bundle.

To designate a link to a specified bundle, use the **ppp multilink group** command for configuring the link. This command restricts the link to join only the specified bundle. When a link negotiates to join a Multilink PPP bundle, the link must provide proper identification that is associated with the Multilink PPP bundle. If the negotiation is successful, the link is assigned to the requested Multilink PPP bundle. If the link provides identification that coincides with the identification associated with a different Multilink PPP bundle in the system or if the link fails to match the identity of a Multilink PPP bundle that is already active on the multilink group interface, the connection terminates.

A link joins a Multilink PPP bundle only if it negotiates to use the bundle when a connection is established and the identification information exchanged matches that of an existing bundle.

When you configure the **ppp multilink group** command on a link, the command applies the following restrictions on the link:

- The link is not allowed to join any bundle other than the indicated group interface.
- The PPP session must be terminated if the peer device attempts to join a different bundle.

A link joins a bundle only when the identification keys for that link match the identification keys for an existing bundle. (See the “Factors that Govern a Link Joining a Bundle” section.) Merely configuring the **ppp multilink group** command on a link does not allow the link to join the corresponding bundle; the link must have matching identification keys to join the corresponding bundle. Identification keys are always used as the determining factor for matching links with bundles.

Because the **ppp multilink group** command merely places a restriction on a link, any Multilink-PPP-enabled link that is not assigned to a particular multilink group can join the dedicated bundle interface if the Multilink-PPP-enabled link provides correct identification keys for that dedicated bundle. Removing the **ppp multilink group** command from an active link that is currently a member of a multilink group does not make that link leave the bundle because the link is still a valid member of the multilink group. However, the link is no longer restricted to this one bundle.

The table below lists the different configurations of Multilink PPP and the number of links supported by each one of them.

Feature Name	Number of Links Supported Per Bundle
Multilink PPP over Serial	10

Multilink PPP over Ethernet	1 or 8 (depending on your release)*
Multilink PPP over Ethernet over VLAN	1 or 8 (depending on your release)*
Multilink PPP over Ethernet over QinQ	1 or 8 (depending on your release)*
Multilink PPP over Ethernet over ATM	1 or 8 (depending on your release)*
Multilink PPP over ATM (AAL5MUX, AL5SNAP)	1 or 8 (depending on your release)*
Multilink PPP over LNS	1 or 8 (depending on your release)*

Link Fragmentation and Interleaving

The main benefit of Multilink PPP is the support for link fragmentation and interleaving (LFI). LFI minimizes packet latency on delay-sensitive voice, video, and interactive applications when data is sent over low-speed interfaces.

With LFI, the latency of delay-sensitive traffic is minimized because Multilink PPP breaks the nonpriority or nonlatency-sensitive traffic into smaller fragments. The delay-sensitive traffic is then PPP encapsulated and interleaved with nonpriority Multilink PPP fragments or packets. At the receiver, Multilink PPP fragments or packets are reordered and reassembled while the PPP-encapsulated packets are received and immediately forwarded. (Multilink PPP is bypassed; no reordering or reassembly is performed.)

Types of Multilink PPP Bundle Interfaces

Multilink PPP bundle interfaces can be one of the following types:

- Multilink group interfaces
- Virtual access interfaces (VAIs)

Both these types of interfaces provide the same level of PPP and multilink functionality after a bundle is established. All PPP and multilink-related features run identically on a bundle.

Multilink Group Interface

A multilink group interface is a static interface that exists whether or not it is being used at a particular point in time. A static interface is defined in the startup configuration file on a device. This type of interface is created automatically when a device boots up. Multilink group interfaces are dedicated to specific remote users. These interfaces are used in leased-line environments, where you have information about all physical links connections and where the number of users is defined by the number of physical connections in the system.

Multilink group interfaces allow you to track a specific user's activity. By examining a user's associated interface, you can easily see whether a user is connected and how much traffic the user has sent or received. You can monitor the state of a multilink group interface for issues such as network outages.

Virtual Access Interface

A virtual access interface (VAI) is a type of interface that is used for Multilink PPP bundles, specifically for Multilink PPP over broadband. A VAI is created dynamically for a multilink connection and released as soon

as a multilink connection is torn down. A bundle interface of this type exists only as long as a user is connected. As soon as a user disconnects, a virtual access interface no longer exists.

A VAI is the default type of bundle interface when Multilink PPP is running in a broadband topology. For broadband topologies, a VAI replaces multilink group interfaces, which are used in serial topologies.

The use of a VAI has the following advantages:

- The number of bundle interfaces depends only on the number of currently active multilink users and not on the size of the user database.
- Because a local configuration source does not exist for per-user information, this information is derived from another source, such as an authentication, authorization, and accounting (AAA) server.

The disadvantage of using a VAI lies in its monitoring capability. A multilink group interface, used in serial deployments, allows the monitoring of each user's traffic through a serial link. With a VAI, there is no dedicated interface defined in a device's configuration file. Therefore, you must track a user's activity by using other means, such as the accounting mechanism of an AAA server.

Factors that Govern a Link Joining a Bundle

A link joins a bundle when identification keys for that link match identification keys for an existing bundle.

The following two keys define the identity of a remote system: the PPP username and Multilink PPP endpoint discriminator. PPP authentication mechanisms (for example, password authentication protocol [PAP] or Challenge-Handshake Authentication Protocol [CHAP]) learn the PPP username. The endpoint discriminator is an option negotiated by the Link Control Protocol (LCP). Therefore, a bundle consists of links that have the same PPP usernames and endpoint discriminators.

A link that does not provide a PPP username or an endpoint discriminator is an anonymous link. Multilink PPP collects all anonymous links into a single bundle that is referred to as the anonymous bundle or the default bundle. Typically, there can be only one anonymous bundle. Any anonymous links that negotiate Multilink PPP join (or create) the anonymous bundle.

When you use multilink group interfaces, more than one anonymous bundle is allowed. When you preassign a link to a Multilink PPP bundle by using the **ppp multilink group** command and the link is anonymous, the link joins the bundle interface to which it is assigned if the interface is not already active and is associated with an anonymous user.

Multilink PPP determines the bundle a link joins by following these steps:

1. When a PPP session is initiated, Multilink PPP creates a bundle name identifier for the link.
2. Multilink PPP then searches for a bundle with the same bundle name identifier. The following scenarios are possible:
    - If a bundle with the same identifier exists, the link joins that bundle.
    - If a bundle with the same identifier does not exist, Multilink PPP creates a new bundle with the same identifier as the link, and the link becomes the first link to join the bundle.

The table below describes commands and associated algorithms that are used to generate a bundle name. In the table, "username" typically means an authenticated username; however, an alternate name can be used instead. The alternate name is usually an expanded version of the username (for example, virtual private dialup network [VPDN] tunnels might include the network access server name) or a name derived from other sources.

Table 42: Bundle Name Generation Commands

Command	Bundle Name Generation Algorithm
multilink bundle-name authenticated	The bundle name is derived from the username that is defined on a peer device. The username is passed from the peer device during PPP negotiation. If a peer does not provide a username, the algorithm uses the peer's endpoint discriminator. **Note** The **authenticated** keyword specifies that the bundle name should be derived by Cisco software. Because multiple bundles may exist on a device concurrently, Cisco software must ensure a unique name for each bundle. The endpoint discriminator is ignored entirely, unless it is the only name that can be found. The **multilink bundle-name authenticated** command is the default naming policy.
multilink bundle-name both	The bundle name is a concatenation of the username and the endpoint discriminator.
multilink bundle-name endpoint	The bundle name is a peer's endpoint discriminator. If there is no endpoint discriminator, the algorithm uses the peer's username.

Rate of Session Establishment for Multilink PPP Bundles

When devices running Cisco software begin negotiating a large number of broadband sessions, a peer device may be constrained by its processing capabilities. This limitation may cause an excessive number of timeouts (because the peer device may be renegotiating hundreds of sessions due to timeouts) while trying to negotiate PPP parameters. Cisco software provides an internal mechanism to control the rate of session establishment (to prevent excessive timeouts) for broadband sessions by using the Call Admission Control (CAC) functionality. CAC can be configured to control the number of sessions that can be negotiated in a given period of time. Controlling the rate of session establishment is also known as throttling. The mechanism of throttling helps to prevent unnecessary negotiation timeouts with slower devices. The following commands show how to configure the CAC functionality:

```
Device(config)# call admission new-model
Device(config)# call admission limit 500
Device(config)# call admission cpu-limit 80
Device(config)# call admission pppoe 10 1
Device(config)# call admission pppoa 10 1
Device(config)# call admission vpdn 10 1
```

- The **call admission new-model** command enables the new-model-based CAC, which regulates session establishment based on both CPU utilization and incoming session requests.
- The **call admission limit** command (also referred to as charge limit) sets the maximum total concurrent charge threshold. If this threshold is exceeded, any additional session requests are rejected.

- The **call admission cpu-limit** command specifies the CPU utilization threshold, as a percentage. If this threshold is exceeded, new sessions are rejected. CAC uses the 5-second CPU utilization of IOS daemon (IOSd) for this calculation.
- Session requests are rejected if either the **cpu-limit** (80% in the example above) or the charge **limit** (1000 in the example above) is exceeded.
- The **call admission pppoe 10 1** command specifies the charge values for a single PPP over Ethernet (PPPoE) session. In the above example, the session charge is 10, and the lifetime is 1 second. The charge values are set per call type (PPP over ATM [PPPoA], PPPoE, or virtual private dialup network [VPDN]). The extended lifetime is calculated as the sum of two lifetime values. For the above example, the extended lifetime is 1 + 1 = 2.
- You can calculate calls per second (CPS) as follows:

  ```
  CPS = Charge Limit / (Session Charge * Extended Lifetime)
  ```

  For the above example, CPS = 500 / (10 * 2) = 25

For a more detailed explanation, refer to the "Broadband Scalability and Performance" module in the *Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide*.

Multilink PPP Packet Overhead

Multilink PPP encapsulation adds six extra bytes (four header and two checksum) to each outbound packet. These overhead bytes reduce the effective bandwidth of a connection; therefore, the throughput for a Multilink PPP bundle is slightly less than that for an equivalent bandwidth connection that is not using Multilink PPP. If the average packet size is large, the extra Multilink PPP overhead is not readily apparent. However, if the average packet size is small, the extra overhead becomes more noticeable.

Multilink PPP fragmentation adds additional overhead to a packet. Each fragment contains six bytes of Multilink PPP header plus a link encapsulation header. The size of the link encapsulation header varies based on the topology of a network. The Layer 2 headers for Ethernet, ATM, and serial interfaces add different number of bytes to a packet.

Multilink PPP over Serial Interfaces

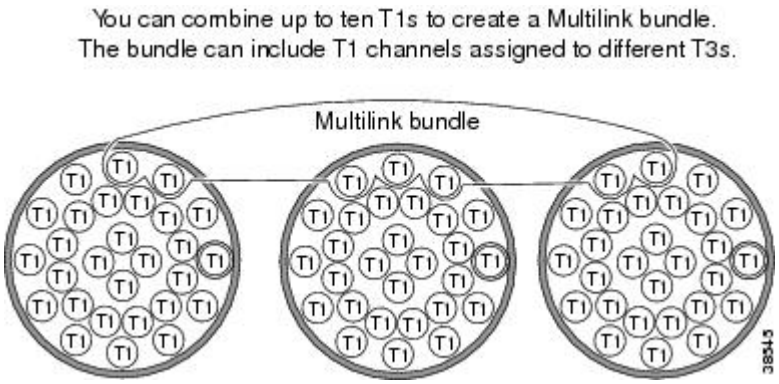
The Multilink PPP over Serial Interfaces feature enables you to bundle T1 interfaces into a single, logical connection called a Multilink PPP bundle. (See the "Multilink PPP Bundles" section.) The Multilink PPP over Serial Interfaces feature also provides the following functionalities:

- Load balancing—Multilink PPP provides bandwidth on demand and uses load balancing across all member links (up to ten) to transmit packets and packet fragments. Multilink PPP mechanisms calculate the load on inbound or outbound traffic between specific sites. Because Multilink PPP splits packets and fragments across all member links during transmission, Multilink PPP reduces transmission latency across WAN links. Ideally, all member links in a bundle would be of the same bandwidth (for example, T1s). Load balancing and fragmentation and interleaving also allow for a mix of unequal cost member links for situations where a small increment in the bundle bandwidth is required.
- Increased redundancy—Multilink PPP allows traffic to flow over remaining member links when a port fails. When you configure a Multilink PPP bundle that consists of T1 lines from more than one line card and if one line card stops operating, part of the bundle on other line cards continues to operate.

- Link fragmentation and interleaving (LFI)—The Multilink PPP fragmenting mechanism fragments large, nonreal-time packets and sends fragments at the same time over multiple point-to-point links to the same remote address. Smaller, real-time packets remain intact. The Multilink PPP interleaving mechanism sends real-time packets between fragments of nonreal-time packets, thus reducing real-time packet delay.

The figure below shows a Multilink PPP bundle that consists of T1 interfaces from three T3 interfaces.

Figure 46: Multilink PPP Bundle for Multilink PPP over Serial Interfaces



Multilink PPP over Broadband

The Multilink PPP over Broadband feature allows you to combine Multilink PPP over Ethernet (MLPoE) and Multilink PPP over ATM (MLPoA) links into a multilink bundle. Functionally, Multilink PPP over broadband is the same as Multilink PPP over serial interfaces, with the exception of interface management. In Multilink PPP over serial interfaces, interfaces are statically defined in the configuration database, that is, in the startup configuration. In Multilink PPP over broadband, link interfaces are created dynamically by Cisco software while negotiating a PPP session.

The Multilink PPP feature operates in the following two deployment schemes: “PTA mode” and “LNS mode.” In the PPP termination and aggregation (PTA) mode, a device acts as the PTA device and terminates Multilink PPP sessions coming from the customer premises equipment (CPE). In the Layer 2 Tunneling Protocol (L2TP) Network Server (LNS) mode, a device terminates Multilink PPP sessions (carried in a Layer 2 tunnel that originates on a L2TP Access Concentrator [LAC] device) coming from the CPE device.

Note Cisco software allows a device to function as a LAC switch. Therefore, in the LNS mode, Cisco software can run on both LAC and LNS devices. A Cisco device cannot act as a CPE device.

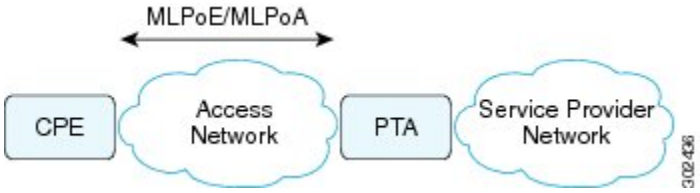
Cisco software can terminate Multilink PPP over ATM (MLPoA) ATM adaptation layer 5 (AAL5) multiplexer (MUX) or AAL5 Subnetwork Access Protocol (SNAP), Multilink PPP over Ethernet (MLPoE), Multilink PPP over Ethernet over ATM (MLPoEoA) AAL5 SNAP, Multilink PPP over Ethernet over Queue-in-Queue (MLPoEoQinQ), or Multilink PPP over Ethernet over VLAN (MLPoEoVLAN) sessions acting as the PTA node. In the LNS mode, Cisco software can terminate Multilink PPP over LNS (MLPoLNS) sessions by using Ethernet (Gigabit Ethernet or 10 Gigabit Ethernet) and ATM (AAL5 MUX and AAL5 SNAP) as the LAC-to-LNS connection to the LAC device. Cisco software also provides support to act as a LAC device to switch broadband Multilink PPP sessions between a CPE or Digital Subscriber Line Access Multiplexer (DSLAM) and an LNS device.

PTA Mode

In the PPP termination and aggregation (PTA) mode, Multilink PPP bundles are terminated at the customer premises equipment (CPE) and at the PTA device. A PTA device terminates the PPP or Multilink PPP session and assigns the network layer address to a client. The client data, present as payload in the Multilink PPP packet, is then forwarded to the backbone network. Depending on the access network, PPP or Multilink PPP frames can be transported to the PTA device by using PPP over ATM (PPPoA) or PPP over Ethernet (PPPoE).

The figure below shows the PTA mode.

Figure 47: PTA Mode



LNS Mode

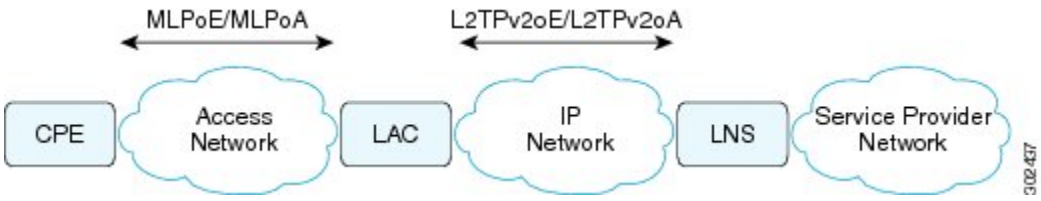
A customer premises equipment (CPE) provides access to a network through various types of access network topologies, including ISDN, asymmetric digital subscriber line (ADSL), and Fiber to the Home (FTTH). If the network protocol used by the CPE device is IPv4, the IP payload will be carried over PPP or Multilink PPP, and the L2TP Access Concentrator (LAC) device will carry the client PPP or Multilink PPP session data to the Layer 2 Tunneling Protocol (L2TP) Network Server (LNS) device by using Layer 2 Tunneling Protocol Version 2 (L2TPv2). The LNS device terminates the PPP or Multilink PPP session and assigns the client its network layer address. The client data, present as the payload in the IPv4 packet, is then forwarded to the backbone network. Depending on the access network, PPP or Multilink PPP frames can be transported to the LAC device by using PPP over ATM (PPPoA) or PPP over Ethernet (PPPoE).

Note Because the LAC device performs Layer 2 Forwarding (L2F) between a CPE and an LNS device, the LAC device does not physically terminate the Multilink PPP session; the LAC device forwards the Multilink PPP session traffic by using L2F.

CPE and LAC devices exchange PPP or Multilink PPP packets that are typically encapsulated using the PPPoE protocol and carried over an Ethernet network. The requirements of an LNS device are independent of protocols that are in use between clients and the LAC device. The LAC device creates L2TPv2 tunnels to all LNS devices to which clients need termination. Multiple tunnels can exist between a given LAC-to-LNS pair of devices for load sharing and redundancy considerations.

The figure below shows the LNS mode.

Figure 48: LNS Mode



For each PPP or Multilink PPP session that is initiated by the CPE device, the LAC device signals the LNS device to add another session to the tunnel. After a session is set up, all traffic from the client, including PPP control packets, is forwarded to the LNS device. After the decision to negotiate the initial Link Control Protocol (LCP) and forward packets to an LNS device has been made, the LAC device becomes Layer 2 transparent to subsequent packets that are received from the client and the LNS device, including further PPP control messages. PPP or Multilink PPP sessions are terminated at the LNS device, and IP packets from the client are routed on the attached ISP or corporate network towards their final destination. The LNS device performs authentication, authorization, and accounting (AAA) actions on PPP or Multilink PPP sessions.



Note In the case of PPP control packet exchanges at PPP or Multilink PPP session initiation time, the Multilink PPP session is terminated at the CPE device and the LNS device, not at the LAC device. The LAC device does not have the knowledge that this is a Multilink PPP session. As a result, when the client CPE first initiates a PPP-LCP connection, the Multilink (Maximum Receive Reconstructed Unit [MRRU]) negotiation included as part of the LCP negotiation may fail if the LAC device has not yet established the connection with the LNS device (which is typically the case). When a connection between the LAC device and the LNS device is established, the LNS device can renegotiate Multilink (MRRU) LCP options with the CPE device, and the two peer devices can then establish an end-to-end Multilink PPP connection.

Performance- and Scalability-Related Commands

When working in large-scale network configurations with many Multilink PPP (or PPP) subscribers, the traffic load when bringing up a scaled session can overwhelm the network and packets may be dropped. For example, such a scenario may occur after a full device reload or after an outage on a network interface that supports large numbers of users.

To minimize recovery issues due to dropped session establishment packets, we recommend that you configure the following commands on the multilink interface (that is, interface multilink, virtual template, or serial member link interfaces):

```
Device(config)# interface GigabitEthernet 1/0/0
Device(config-if)# ppp max-configure 30
Device(config-if)# ppp max-failure 30
Device(config-if)# ppp timeout retry 5
Device(config-if)# keepalive 30
```

The above commands are the recommended starting points. These values can be configured as required depending on the scale of the network.

Some other issues related to network packet drops, observed when bringing up large-scale networks, include PPP or Multilink PPP sessions coming up but missing adjacencies in IP forwarding tables (that is, IP addresses are not assigned to sessions).

The **keepalive** command listed above affects how quickly a link is flagged as disabled if it is no longer functional. If a Multilink PPP bundle is used with multiple member link sessions and if one or more links are removed, the higher the keepalive interval the longer Multilink PPP will require to detect the disabled link and remove from the list of member links. This delay can cause a period of packet loss and delays on the bundle until the disabled link has been detected. The default keepalive interval, if not specified by the user, is 10 seconds, and the default number of keepalive retries is 4.

Note When a link is disabled due to a loss of signal, shut down, or due to a major alarm type of condition, the disabled link may be detected outside of the keepalive mechanism and reported as disabled before the keepalive timeout. The keepalive mechanism applies wherever the link is disabled but appears functional at the physical layer.

When configuring Broadband Aggregation (BBA) groups over an Ethernet interface, there are limits that must be adjusted to match the scale of the configuration. The number of broadband sessions can be limited per MAC address, per VLAN, and per virtual circuit (VC). The following example shows how to use the **bba-group pppoe** command to configure a BBA group over Ethernet:

```
Device(config)# bba-group pppoe global
Device(config-bba-group)# sessions per-mac limit 2000
Device(config-bba-group)# sessions per-vlan limit 2000
Device(config-bba-group)# sessions per-vc limit 2000 threshold 2000
```

If these per-session numbers are too low, there will be fewer established sessions between devices.

Multilink PPP over ATM on the PTA Device

Cisco software supports Multilink PPP over ATM (MLPoA) by using single-link and multilink bundles. Cisco software supports MLPoA over ATM adaptation layer 5 (AAL5) multiplexer (MUX) or AAL5 Subnetwork Access Protocol (SNAP).

The network topology and functional support for MLPoA is similar to the Multilink PPP over Ethernet (MLPoE) topology on the PTA device. The difference is that the connection between the PTA device and downstream devices is ATM instead of Ethernet.

Multilink PPP over Ethernet over ATM on the PTA Device

Cisco software supports Multilink PPP over Ethernet over ATM (MLPoEoA) by using single-link and multilink bundles. Cisco software supports MLPoEoA over ATM adaptation layer 5 (AAL5) Subnetwork Access Protocol (SNAP).

The network topology and functional support for MLPoEoA is similar to the Multilink PPP over Ethernet (MLPoE) topology on the PTA device. The difference is that the connection between the PTA device and downstream devices is ATM instead of Ethernet.

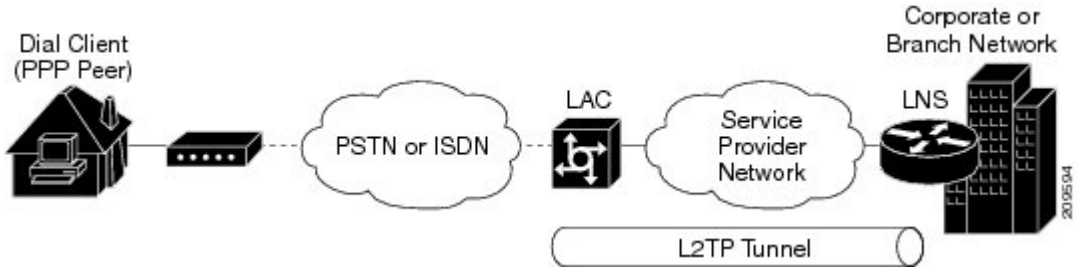
Multilink PPP over LNS

Cisco software supports Multilink PPP over LNS (MLPoLNS) by using single-link and multilink bundles. The data link layer for MLPoLNS is Layer 2 Tunneling Protocol Version 2 (L2TPv2). For MLPoLNS, Ethernet is used for transport between L2TP Network Server (LNS) and L2TP Access Concentrator (LAC) devices.

The Multilink PPP over LNS feature bundles one or more virtual private dialup network (VPDN) sessions in a single logical connection, which forms a Multilink PPP bundle on the LNS device. From a Multilink PPP perspective, Multilink PPP over LNS is similar to Multilink PPP over ATM or over Ethernet or over Ethernet over ATM, except that in this case, Multilink PPP or PPP packets are L2TPv2 encapsulated. The bandwidth of a member link session can be determined on a LAC device by using the *connect speed* attribute-value pair.

The L2TP tunnel between a LAC device and an LNS device carries both PPP and Multilink PPP traffic. The LAC device acts as a switch and forwards both PPP and Multilink PPP session packets between the CPE device and the LNS device. The figure below shows a LAC-to-LNS dialup network.

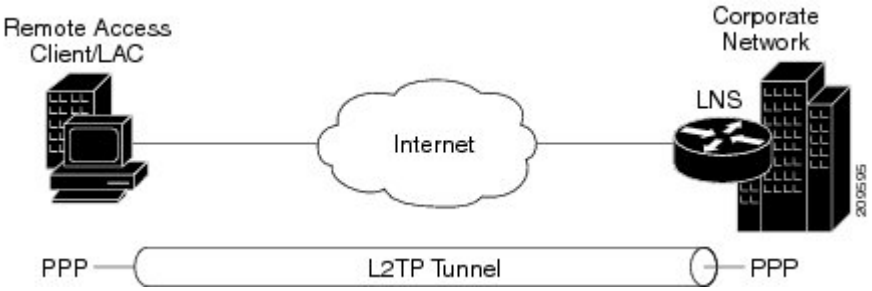
Figure 49: Dialup LAC to LNS



L2TP-client-initiated tunneling allows a client device to initiate L2TP tunnels. A client device can initiate an L2TP tunnel to the LNS device without the intermediate Network Access Server (NAS) participating in tunnel negotiation or establishment. The benefit of L2TP-client-initiated tunneling is that client devices can initiate L2TP tunnels.

The figure below shows an example of a client-initiated tunneling scenario. The client connects to the NAS through a medium, such as a dialup modem, Digital Subscriber Line (DSL), ISDN, or a cable modem, that supports PPP. The client can initiate an L2TP tunnel to the LNS device.

Figure 50: L2TPv2 Tunnel to an LNS Device



The L2TP Client-Initiated Tunneling feature uses a virtual PPP interface, which adds Layer 2 encapsulation to Layer 3 packets, allowing these packets to be sent to the LNS device over an L2TPv2 tunnel.

Switching Multilink PPP Traffic Through a LAC Device

Cisco software supports the switching of broadband Multilink PPP sessions between customer premises equipment (CPE) and LNS devices through a LAC device.

In the LAC switching mode, the LAC device provides the Multilink PPP over ATM (MLPoATM), Multilink PPP over Ethernet (MLPoE), or Multilink PPP over Ethernet over ATM (MLPoEoA) connection to the CPE device and the virtual private dialup network (VPDN) connection to the LNS device. The LAC-to-CPE connection may be any of the variations supported through MLPoA, MLPoE, or MLPoEoA in the PPP termination and aggregation (PTA) mode.

When switching Multilink PPP traffic through a LAC device, Multilink PPP sessions are not terminated. Multilink PPP sessions are terminated by CPE and LNS devices. The LAC device manages the establishment of the member link VPDN session between the CPE and LNS devices and the switching of the session data. Data from the CPE device is L2TPv2 encapsulated and passed to the LNS device. Data from the LNS device is stripped off of the L2TPv2 data and passed on to the CPE device.

QoS Traffic and Shaping

Quality of service (QoS) is applied to a Multilink PPP bundle by using a service policy. This policy is then applied to the Multilink PPP bundle interface, which appears as a PPP interface.



Note For Multilink PPP over serial interfaces, the service policy is applied to the multilink interface configuration. For broadband topologies, the service policy is applied to the virtual template configuration.

QoS allows you to manually account for the extra overhead imposed by the external interface and other Layer 2 encapsulations by using the **account** and **user-defined** keywords in the **shape** command. One such example includes ATM cell overhead (extra bits or bytes are added to packets when running traffic on an ATM network). This additional overhead is not accounted for because the extra bytes are removed by the external interface before the QoS policy is applied to the packets. For more information, see the *QoS: Policing and Shaping Configuration Guide*.

How to Configure Multilink PPP Connections for Broadband and Serial Topologies

Configuring Multilink PPP

Before configuring Multilink PPP connections between CPE and PTA devices, you must configure Multilink PPP by performing the following tasks:

Creating a Multilink Bundle

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface multilink** *group-number*
4. **ip address** *address mask*
5. **ppp multilink**
6. **ppp multilink group** *group-number*
7. **end**
8. **show ppp multilink**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface multilink** *group-number* **Example:** `Device(config)# interface multilink 10`	Assigns a multilink bundle group number and enters interface configuration mode.
Step 4	**ip address** *address mask* **Example:** `Device(config-if)# ip address 192.0.2.9 255.255.255.224`	Assigns an IP address to the multilink interface.
Step 5	**ppp multilink** **Example:** `Device(config-if)# ppp multilink`	Enables Multilink PPP.
Step 6	**ppp multilink group** *group-number* **Example:** `Device(config-if)# ppp multilink group 12`	Restricts a physical link to join only the designated multilink-group interface.
Step 7	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 8	**show ppp multilink** **Example:** `Device# show ppp multilink`	Displays Multilink PPP bundle information.

Assigning an Interface to a Multilink Bundle



Caution Do not install a route to the peer address while configuring a Multilink PPP lease line. The route can be disabled using the **no ppp peer-neighbor-route** command on the Multilink PPP bundle interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface serial** *slot/subslot/port*
4. **no ip address**
5. **encapsulation ppp**
6. **keepalive** *seconds*

7. **ppp multilink**
8. **ppp multilink group** *group-number*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface serial** *slot/subslot/port* **Example:** `Device(config)# interface serial 1/3/0`	Assigns a multilink bundle group number and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Removes any specified IP address.
Step 5	**encapsulation ppp** **Example:** `Device(config-if)# encapsulation ppp`	Enables PPP encapsulation.
Step 6	**keepalive** *seconds* **Example:** `Device(config-if)# keepalive 50`	Sets the frequency of keepalive packets.
Step 7	**ppp multilink** **Example:** `Device(config-if)# ppp multilink`	Enables Multilink PPP.
Step 8	**ppp multilink group** *group-number* **Example:** `Device(config-if)# ppp multilink group 12`	Restricts a physical link to join only the designated multilink-group interface.
Step 9	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Minimum Multilink PPP Links

Perform this task to configure the minimum number of links in a Multilink PPP bundle, which are required to keep that bundle active.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface multilink** *group-number*
4. **ppp multilink**
5. **ppp multilink min-links** *links* **mandatory**
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface multilink** *group-number* **Example:** `Device(config)# interface multilink 10`	Assigns a multilink bundle group number and enters interface configuration mode.
Step 4	**ppp multilink** **Example:** `Device(config-if)# ppp multilink`	Enables Multilink PPP.
Step 5	**ppp multilink min-links** *links* **mandatory** **Example:** `Device(config-if)# ppp multilink min-links 5 mandatory`	Specifies the required minimum number of links in a Multilink PPP bundle. • If the minimum number of links in the Multilink PPP bundle falls below the number specified by the *links* argument, the Multilink PPP bundle is disabled.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Changing the Default Endpoint Discriminator

By default, when the system negotiates the use of Multilink PPP with a peer device, the value that is supplied for the endpoint discriminator is the same as the username used for authentication. The username is configured for the interface by using the **ppp chap hostname** or **ppp pap sent-username** command. If not configured, the username defaults to the globally configured hostname (or stack group name if this interface is a Stack Group Bidding Protocol [SGBP] group member).

Perform this task to override or change the default endpoint discriminator.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface virtual-template** *number*
4. **ppp multilink endpoint** {**hostname** | **ip** *ip-address* | **mac** *lan-interface* | **none** | **phone** *telephone-number* | **string** *char-string*}
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface virtual-template** *number* **Example:** `Device(config)# interface virtual-template 1`	Creates a virtual template interface that can be configured and applied dynamically for creating VAIs and enters interface configuration mode.
Step 4	**ppp multilink endpoint** {**hostname** | **ip** *ip-address* | **mac** *lan-interface* | **none** | **phone** *telephone-number* | **string** *char-string*} **Example:** `Device(config-if)# ppp multilink endpoint ip 209.165.201.20`	Overrides or changes the default endpoint discriminator that the system uses when negotiating the use of Multilink PPP with a peer.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP Interleaving and Queueing

Multilink PPP support for interleaving can be configured on virtual templates. To configure interleaving, first configure a virtual template and then configure Multilink PPP and interleaving on the interface or template. These tasks are described in the "Configuring Multilink PPP Interleaving" section.

Note Fair queueing, which is enabled by default, must remain enabled on the interface.

This section covers the following tasks:

Configuring Multilink PPP Interleaving

Interleaving statistics can be displayed by using the **show interfaces** command, specifying the particular interface on which interleaving is enabled. Interleaving data is displayed only if there are interleaves. For example, the following line shows interleaves:

```
Output queue: 315/64/164974/31191 (size/threshold/drops/interleaves)
```

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface virtual-template** *number*
4. **ppp multilink**
5. **ppp multilink interleave**
6. **ppp multilink fragment delay** *milliseconds*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface virtual-template** *number* **Example:** `Device(config)# interface virtual-template 1`	Creates a virtual template interface that can be configured and applied dynamically for creating VAIs and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**ppp multilink** **Example:** `Device(config-if)# ppp multilink`	Enables Multilink PPP.
Step 5	**ppp multilink interleave** **Example:** `Device(config-if)# ppp multilink interleave`	Enables the interleaving of packets among fragments of larger packets on a Multilink PPP bundle.
Step 6	**ppp multilink fragment delay** *milliseconds* **Example:** `Device(config-if)# ppp multilink fragment delay 50`	Specifies the maximum size, in units of time, for packet fragments on a Multilink PPP bundle.
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Disabling PPP Multilink Fragmentation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface multilink** *group-number*
4. **ppp multilink fragment disable**
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface multilink** *group-number* **Example:** `Device(config)# interface multilink 10`	Assigns a multilink group number and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**ppp multilink fragment disable** **Example:** `Device(config-if)# ppp multilink fragment disable`	Disables PPP multilink fragmentation.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over Broadband

The following sections provide information about configuring Multilink PPP connections between CPE and PTA devices for MLPoA, MLPoE, MLPoEoA, and MLPoLNS:

Creating a Class Map

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map** *class-map-name* [**match-all** | **match-any**]
4. **match ip precedence** *precedence-criteria*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**class-map** *class-map-name* [**match-all**	**match-any**] **Example:** `Device(config)# class-map ip-prec-1 match-all`	Creates a class map to be used for matching traffic with the specified class and enters QoS class-map configuration mode.
Step 4	**match ip precedence** *precedence-criteria* **Example:** `Device(config-cmap)# match ip precedence 1`	(Optional) Specifies the match criteria in a class map.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-cmap)# end`	Exits QoS class-map configuration mode and returns to privileged EXEC mode.

Creating a Policy Map

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** {*class-name* | **class-default**}
5. **priority percent** *percentage*
6. **shape** {**average** | **peak**} *mean-rate*
7. **service-policy** *policy-map*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map mlp-parent-policy-10m`	Specifies the name of the policy map to be created and enters QoS policy-map configuration mode.
Step 4	**class** {*class-name*	**class-default**} **Example:** `Device(config-pmap)# class ip-prec-1`	Specifies the name of the class and enters QoS policy-map class configuration mode. • This class is associated with the class map created in the “Creating a Class Map” section.
Step 5	**priority percent** *percentage* **Example:** `Device(config-pmap-c)# priority percent 10`	(Optional) Specifies the percentage of the total available bandwidth to be set aside for the priority class.

	Command or Action	Purpose
Step 6	**shape** {**average** | **peak**} *mean-rate* **Example:** `Device(config-pmap-c)# shape average 250000`	Shapes traffic to the indicated bit rate according to the algorithm specified.
Step 7	**service-policy** *policy-map* **Example:** `Device(config-pmap-c)# service-policy mlp-parent-policy-10m`	Attaches a policy map to a class.
Step 8	**end** **Example:** `Device(config-pmap-c)# end`	Exits QoS policy-map class configuration mode and returns to privileged EXEC mode.

Defining a PPP over Ethernet Profile

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **bba-group pppoe** {*group-name* | **global**}
4. **virtual-template** *template-number*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**bba-group pppoe** {*group-name* | **global**} **Example:** `Device(config)# bba-group pppoe mlpppoe-bba-group-10m`	Defines a PPPoE profile and enters BBA group configuration mode.
Step 4	**virtual-template** *template-number* **Example:** `Device(config-bba-group)# virtual-template 18`	Specifies the virtual template to be used to clone VAIs for all PPPoE ports that use this PPPoE profile.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-bba-group)# end`	Exits BBA group configuration mode and returns to privileged EXEC mode.

Configuring a Virtual Template Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface virtual-template** *number*
4. **peer default ip address pool** [*pool-name*]
5. **ppp multilink**
6. **ppp multilink interleave**
7. **ppp multilink endpoint string** *char-string*
8. **ppp multilink retry** *seconds*
9. **service-policy output** *policy-map-name*
10. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface virtual-template** *number* **Example:** `Device(config)# interface virtual-template 1`	Creates a virtual template interface that can be configured and applied dynamically for creating VAIs and enters interface configuration mode.
Step 4	**peer default ip address pool** [*pool-name*] **Example:** `Device(config-if)# peer default ip address pool MLP-IPv4-Pool`	Specifies an IP address from an IP address pool to be returned to a remote peer that is connected to this interface.
Step 5	**ppp multilink** **Example:**	Enables Multilink PPP.

	Command or Action	Purpose
	`Device(config-if)# ppp multilink`	
Step 6	**ppp multilink interleave** **Example:** `Device(config-if)# ppp multilink interleave`	Enables interleaving of packets among fragments of larger packets on a multilink bundle.
Step 7	**ppp multilink endpoint string** *char-string* **Example:** `Device(config-if)# ppp multilink endpoint string Dialer2-CPE-MLPoE`	Overrides or changes the default endpoint discriminator that the system uses when negotiating the use of Multilink PPP with a peer.
Step 8	**ppp multilink retry** *seconds* **Example:** `Device(config-if)# ppp multilink retry 4`	Sets the maximum waiting period for a response during PPP negotiation.
Step 9	**service-policy output** *policy-map-name* **Example:** `Device(config-if)# service-policy output mlp-parent-10m`	Attaches the previously created traffic policy (policy map). • The policy map evaluates and applies QoS features for traffic leaving the interface.
Step 10	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over ATM on the CPE Device

Before configuring Multilink PPP over ATM, you must complete the following tasks:

- Creating a Class Map
- Creating a Policy Map
- Creating a Dialer Interface

This section covers the following tasks:

Configuring Multilink PPP over ATM Using AAL5 MUX Encapsulation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}]
4. **pvc** *vpi/vci*
5. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
6. **encapsulation aal5mux** *protocol*
7. **dialer pool-number** *number*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}] **Example:** `Device(config)# interface atm 6/0.20 point-to-point`	Specifies the ATM interface for which Multilink PPP must be configured and enters interface configuration mode.
Step 4	**pvc** *vpi/vci* **Example:** `Device(config-if)# pvc 20/10`	Specifies the encapsulation type on an ATM permanent virtual circuit (PVC) and enters ATM virtual circuit configuration mode.
Step 5	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-if-atm-vc)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.
Step 6	**encapsulation aal5mux** *protocol* **Example:** `Device(config-if-atm-vc)# encapsulation aal5mux ppp dialer`	Configures the ATM adaptation layer (AAL) and encapsulation type for an ATM virtual circuit (VC).
Step 7	**dialer pool-number** *number* **Example:** `Device(config-if-atm-vc)# dialer pool-number 4`	Configures a physical interface as a member of a dialer profile dialing pool.
Step 8	**end** **Example:** `Device(config-if-atm-vc)# end`	Exits ATM virtual circuit configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over ATM Using AAL5 SNAP Encapsulation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}]

4. **pvc** *vpi/vci*
5. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
6. **encapsulation aal5snap**
7. **protocol ppp dialer**
8. **dialer pool-number** *number*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}] **Example:** `Device(config)# interface atm 6/0.20 point-to-point`	Specifies the ATM interface for which Multilink PPP must be configured and enters interface configuration mode.
Step 4	**pvc** *vpi/vci* **Example:** `Device(config-if)# pvc 20/10`	Specifies the encapsulation type on an ATM permanent virtual circuit (PVC) and enters ATM virtual circuit configuration mode.
Step 5	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-if-atm-vc)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.
Step 6	**encapsulation aal5snap** **Example:** `Device(config-if-atm-vc)# encapsulation aal5snap`	Configures the ATM adaptation layer (AAL) and encapsulation type for an ATM virtual circuit (VC).
Step 7	**protocol ppp dialer** **Example:** `Device(config-if-atm-vc)# protocol ppp dialer`	Configures a static map for an ATM PVC.
Step 8	**dialer pool-number** *number* **Example:** `Device(config-if-atm-vc)# dialer pool-number 4`	Configures a physical interface as a member of a dialer profile dialing pool.

	Command or Action	Purpose
Step 9	**end** **Example:** `Device(config-if-atm-vc)# end`	Exits ATM virtual circuit configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over Ethernet over ATM at the CPE

Before you begin

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}]
4. **pvc** *vpi*/*vci*
5. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
6. **pppoe-client dial-pool-number** *number*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *interface-number*[**.***subinterface-number* {**mpls**	**multipoint**	**point-to-point**}] **Example:** `Device(config)# interface atm 6/0.20 point-to-point`	Specifies the ATM interface for which Multilink PPP must be configured and enters interface configuration mode.
Step 4	**pvc** *vpi*/*vci* **Example:** `Device(config-if)# pvc 20/10`	Specifies the encapsulation type on an ATM permanent virtual circuit (PVC) and enters ATM virtual circuit configuration mode.
Step 5	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-if-atm-vc)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.

	Command or Action	Purpose
Step 6	**pppoe-client dial-pool-number** *number* **Example:** `Device(config-if-atm-vc)# pppoe-client dial-pool-number 6`	Configures a PPPoE client.
Step 7	**end** **Example:** `Device(config-if-atm-vc)# end`	Exits ATM virtual circuit configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over ATM on the PTA Device

Before configuring Multilink PPP over ATM, you must complete the following tasks:

- Creating a Class Map
- Creating a Policy Map
- Defining a PPP over Ethernet Profile
- Configuring a Virtual Template Interface

This section covers the following tasks:

Configuring Multilink PPP over ATM Using AAL5 MUX Encapsulation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}]
4. **pvc** *vpi/vci*
5. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
6. **encapsulation aal5mux** *protocol* **virtual-template** *virtual-template*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}] **Example:** `Device(config)# interface atm 2/2/1.20 point-to-point`	Specifies the ATM interface for which Multilink PPP must be configured and enters interface configuration mode.
Step 4	**pvc** *vpi/vci* **Example:** `Device(config-if)# pvc 20/10`	Specifies the encapsulation type on an ATM PVC and enters ATM virtual circuit configuration mode.
Step 5	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-if-atm-vc)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.
Step 6	**encapsulation aal5mux** *protocol* **virtual-template** *virtual-template* **Example:** `Device(config-if-atm-vc)# encapsulation aal5mux ppp virtual-template 18`	Configures the ATM adaptation layer (AAL) and encapsulation type for an ATM virtual circuit (VC) and assigns the VC to the previously specified virtual template.
Step 7	**end** **Example:** `Device(config-if-atm-vc)# end`	Exits ATM virtual circuit configuration mode returns to privileged EXEC mode.

Configuring Multilink PPP over ATM Using AAL5 SNAP Encapsulation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}]
4. **pvc** *vpi/vci*
5. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
6. **encapsulation aal5snap**
7. **protocol** *protocol* **virtual-template** *virtual-template*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}] **Example:** `Device(config)# interface atm 2/2/1.22 point-to-point`	Specifies the ATM interface for which Multilink PPP must be configured and enters interface configuration mode.
Step 4	**pvc** *vpi*/*vci* **Example:** `Device(config-if)# pvc 20/10`	Specifies the encapsulation type on an ATM PVC and enters ATM virtual circuit configuration mode.
Step 5	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-if-atm-vc)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.
Step 6	**encapsulation aal5snap** **Example:** `Device(config-if-atm-vc)# encapsulation aal5snap`	Configures the ATM adaptation layer (AAL) and encapsulation type for an ATM virtual circuit (VC).
Step 7	**protocol** *protocol* **virtual-template** *virtual-template* **Example:** `Device(config-if-atm-vc)# protocol ppp virtual-template 18`	Specifies parameters that PPPoA sessions use.
Step 8	**end** **Example:** `Device(config-if-atm-vc)# end`	Exits ATM virtual circuit configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over Ethernet over ATM on the PTA Device

Before configuring Multilink PPP over Ethernet over ATM, you must complete the following tasks:

- Creating a Class Map
- Creating a Policy Map
- Defining a PPP over Ethernet Profile
- Configuring a Virtual Template Interface

This section covers the following tasks:

Configuring a Virtual Circuit Class

Before configuring MLPoEoA, you must create and configure a virtual circuit (VC) class.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vc-class atm** *name*
4. **protocol pppoe** [**group** *group-name*]
5. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
6. **encapsulation aal5snap**
7. **create-on demand**
8. **idle-timeout** *minutes*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vc-class atm** *name* **Example:** `Device(config)# vc-class atm mlpoeoa-vc-class`	Creates a VC class and enters VC-class configuration mode.
Step 4	**protocol pppoe** [**group** *group-name*] **Example:** `Device(config-vc-class)# protocol pppoe group mlpoeoa-bba-group`	Enables PPP over Ethernet (PPPoE) sessions to be established on permanent virtual circuits (PVCs).
Step 5	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-vc-class)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies the output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.
Step 6	**encapsulation aal5snap** **Example:** `Device(config-vc-class)# encapsulation aal5snap`	Configures the ATM adaptation layer (AAL) and encapsulation type for an ATM VC.

	Command or Action	Purpose
Step 7	**create-on demand** **Example:** `Device (config-vc-class)# create-on demand`	Configures ATM PVC autoprovisioning, which enables a PVC or range of PVCs to be created automatically on demand.
Step 8	**idle-timeout** *minutes* **Example:** `Device(config-vc-class)# idle-timeout 10`	Sets a time to keep the session alive in the absence of any data traffic.
Step 9	**end** **Example:** `Device(config-vc-class)# end`	Exits VC-class configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over Ethernet over ATM Using AAL5 SNAP Encapsulation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm** *interface-number*[**.***subinterface-number* {**mpls** | **multipoint** | **point-to-point**}]
4. **pvc** *vpi/vci*
5. **class-vc** *name*
6. **vbr-nrt** *output-pcr output-scr output-maxburstsize*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *interface-number*[**.***subinterface-number* {**mpls**	**multipoint**	**point-to-point**}] **Example:** `Router (config)# interface atm 2/2/1.24 point-to-point`	Specifies the ATM interface for which Multilink PPP must be configured and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**pvc** *vpi/vci* **Example:** `Device(config-if)# pvc 20/10`	Specifies the encapsulation type on an ATM permanent virtual circuit (PVC) and enters ATM virtual circuit configuration mode.
Step 5	**class-vc** *name* **Example:** `Device(config-if-atm-vc)# class-vc mlpppoeoa-vc-class`	Assigns a virtual circuit (VC) class to an ATM PVC.
Step 6	**vbr-nrt** *output-pcr output-scr output-maxburstsize* **Example:** `Device(config-if-atm-vc)# vbr-nrt 512 256 20`	Configures the variable bit rate-nonreal time (VBR-NRT) and specifies the output peak cell rate (PCR), output sustainable cell rate (SCR), and output maximum burst cell size for an ATM PVC.
Step 7	**end** **Example:** `Device(config-if-atm-vc)# end`	Exits ATM virtual circuit configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over LNS

Configuring an LNS Device to Initiate and Receive L2TP Traffic

Before performing this task, you must configure the virtual template interface as described in the "Configuring a Virtual Template Interface" section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vpdn enable**
4. **vpdn-group** *group-name*
5. **accept-dialin**
6. **protocol l2tp**
7. **virtual-template** *template-number*
8. **exit**
9. **terminate-from hostname** *hostname*
10. **lcp renegotiation** {**always** | **on-mismatch**}
11. **no l2tp authentication**
12. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vpdn enable** **Example:** `Device(config)# vpdn enable`	Enables virtual private dialup network (VPDN) on the device. • The device looks for tunnel definitions in a local database and on a remote authorization server (home gateway) if one is present.
Step 4	**vpdn-group** *group-name* **Example:** `Device(config)# vpdn-group group1`	Defines a local group name for which you can assign other VPDN variables and enters VPDN group configuration mode.
Step 5	**accept-dialin** **Example:** `Device(config-vpdn)# accept-dialin`	Configures the LNS device to accept tunneled PPP connections from the LAC device, creates an accept-dialin VPDN subgroup, and enters VPDN accept-dialin group configuration mode.
Step 6	**protocol l2tp** **Example:** `Device(config-vpdn-acc-in)# protocol l2tp`	Specifies the Layer 2 Tunnel Protocol (L2TP).
Step 7	**virtual-template** *template-number* **Example:** `Device(config-vpdn-acc-in)# virtual-template 18`	Specifies the virtual template to be used to clone virtual access interfaces (VAIs).
Step 8	**exit** **Example:** `Device(config-vpdn-acc-in)# exit`	Returns to VPDN group configuration mode.
Step 9	**terminate-from hostname** *hostname* **Example:** `Device(config-vpdn)# terminate-from hostname LAC3`	Specifies the hostname of the remote LAC device that will be required when accepting a VPDN tunnel.
Step 10	**lcp renegotiation** {**always** | **on-mismatch**} **Example:**	Allows the LNS device to renegotiate the PPP Link Control Protocol (LCP).

	Command or Action	Purpose
	`Device(config-vpdn)# lcp renegotiation always`	
Step 11	**no l2tp authentication** **Example:** `Device(config-vpdn)# no l2tp authentication`	Disables L2TP tunnel authentication.
Step 12	**end** **Example:** `Device(config-vpdn)# end`	Returns to privileged EXEC mode.

Configuring a LAC Device to Initiate and Receive L2TP Traffic

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vpdn enable**
4. **vpdn-group** *group-name*
5. **request-dialin**
6. **protocol l2tp**
7. **exit**
8. **initiate-to ip** *ip-address* [**priority** *priority-number*]
9. **local name** *hostname*
10. **no l2tp authentication**
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vpdn enable** **Example:** `Router(config)# vpdn enable`	Enables virtual private dialup network (VPDN) on the device.

	Command or Action	Purpose
Step 4	**vpdn-group** *group-name* **Example:** `Device(config)# vpdn-group group1`	Defines a local group name for which you can assign other VPDN variables and enters VPDN group configuration mode.
Step 5	**request-dialin** **Example:** `Device(config-vpdn)# request-dialin`	Enables the LAC device to request Layer 2 Tunneling Protocol (L2TP) tunnels and enters VPDN request-dialin group configuration mode.
Step 6	**protocol l2tp** **Example:** `Device(config-vpdn-req-in)# protocol l2tp`	Specifies the L2TP protocol.
Step 7	**exit** **Example:** `Device(config-vpdn-req-in)# exit`	Returns to VPDN group configuration mode.
Step 8	**initiate-to ip** *ip-address* [**priority** *priority-number*] **Example:** `Device(config-vpdn)# initiate-to ip 10.0.0.4`	Specifies the LNS IP address and optionally, the priority of the IP address (1 is the highest).
Step 9	**local name** *hostname* **Example:** `Device(config-vpdn)# local name LAC3`	Specifies a local hostname that the tunnel will use to identify itself.
Step 10	**no l2tp authentication** **Example:** `Device(config-vpdn)# no l2tp authentication`	Disables L2TP tunnel authentication.
Step 11	**end** **Example:** `Device(config-vpdn)# end`	Exits VPDN group configuration mode and returns to privileged EXEC mode.

Configuring Multilink PPP over Serial Interfaces

To configure Multilink PPP over serial interfaces, configure serial interfaces to support PPP encapsulation and Multilink PPP. Repeat the steps below for as many serial interfaces as required.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface serial** *number*
4. **no ip address**
5. **encapsulation ppp**

6. **ppp multilink**
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface serial** *number* **Example:** `Device(config)# interface serial 1`	Specifies an asynchronous interface and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Removes any specified IP address.
Step 5	**encapsulation ppp** **Example:** `Device(config-if)# encapsulation ppp`	Enables PPP encapsulation.
Step 6	**ppp multilink** **Example:** `Device(config-if)# ppp multilink`	Enables Multilink PPP.
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for Multilink PPP Connections for Broadband and Serial Topologies

Example: Configuring Multilink PPP

Multilink PPP provides characteristics most similar to hardware inverse multiplexers with good manageability and Layer 3 services support. The following example shows the configuration of Multilink PPP with traffic shaping and QoS. In this example, two bundles with four links each are configured between two devices. The **ppp chap hostname** command entries are required for originating and terminating multiple bundles on a single pair of devices.

```
controller T3 0/3/1
 framing c-bit
 cablelength 224
 t1 1 channel-group 0 timeslots 1-24
 t1 2 channel-group 0 timeslots 1-24
 t1 3 channel-group 0 timeslots 1-24
 t1 4 channel-group 0 timeslots 1-24
 t1 5 channel-group 0 timeslots 1-24
 t1 6 channel-group 0 timeslots 1-24
 t1 7 channel-group 0 timeslots 1-24
 t1 8 channel-group 0 timeslots 1-24
!
class-map match-all DETERMINISTICOUT
  match ip precedence 3
class-map match-all VOICEVIDEOCONTROLOUT
  match ip precedence 2
class-map match-all VOICEOUT
  match ip precedence 1
class-map match-all ROUTINGPROTOCOLS
  match ip precedence 5
class-map match-all CONTROLLEDLOADOUT
  match ip precedence 4
!
policy-map QOS304QCHILD
 class VOICEOUT
    priority level 1
    police cir percent 30
 class VOICEVIDEOCONTROLOUT
    priority level 2
    police cir percent 5
 class DETERMINISTICOUT
    bandwidth remaining ratio 20
 class CONTROLLEDLOADOUT
    bandwidth remaining ratio 18
 class ROUTINGPROTOCOLS
    bandwidth remaining ratio 4
 class class-default
    bandwidth remaining ratio 22
policy-map ASRMLP6MBPARENT
 class class-default
    shape average percent 98
   service-policy QOS304QCHILD
!
interface multilink 1
 ip address 192.168.1.1 255.255.255.0
 ppp chap hostname multilink_name-1
```

```
 ppp multilink
 ppp multilink group 1
 service-policy output ASRMLP6MBPARENT
!
interface multilink 2
 ip address 192.168.2.1 255.255.255.0
 ppp chap hostname multilink_name-2
 ppp multilink
 ppp multilink group 2
 service-policy output ASRMLP6MBPARENT
!
interface serial 0/3/1/1:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-1
 ppp multilink
 ppp multilink group 1
!
interface serial 0/3/1/2:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-1
 ppp multilink
 ppp multilink group 1
!
interface serial 0/3/1/3:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-1
 ppp multilink
 ppp multilink group 1
!
interface serial 0/3/1/4:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-1
 ppp multilink
 ppp multilink group 1
!
interface serial 0/3/1/5:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-2
 ppp multilink
 ppp multilink group 2
!
interface serial 0/3/1/6:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-2
 ppp multilink
 ppp multilink group 2
!
interface serial 0/3/1/7:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-2
```

```
 ppp multilink
 ppp multilink group 2
!
interface serial 0/3/1/8:0
 no ip address
 encapsulation ppp
 no keepalive
 ppp chap hostname multilink_name-2
 ppp multilink
 ppp multilink group 2
!
```

Example: Configuring Multilink PPP over ATM on the PTA Device

Example: Configuring Multilink PPP over ATM Using AAL5 MUX Encapsulation

The following example shows how to configure Multilink PPP over ATM (MLPoA) with AAL5 MUX encapsulation on the PTA device:

```
class-map match-all ip-prec-1
 match ip precedence 1
policy-map mlp-child-lfi-policy
 class ip-prec-1
  priority percent 10
policy-map mlp-parent-250K
 class class-default
  shape average 250000
  service-policy mlp-child-lfi-policy
policy-map mlp-parent-10M
 class class-default
  shape average 10000000
  service-policy mlp-child-lfi-policy
interface virtual-template 18
 description MLPoEoA/MLPoA aal5mux/aal5snap (single-link bundle) Virtual Template
 ip address negotiated
 peer default ip address pool MLP-IPv4-Pool
 ppp max-failure 30
 ppp chap password 0 password1
 ppp multilink
 ppp multilink interleave
 ppp multilink endpoint magic-number
 ppp timeout retry 4
 service-policy output mlp-parent-250K
bba-group pppoe mlpoeoa-bba-group-250K
 virtual-template 18
ip local pool MLP-IPv4-Pool 209.165.201.2 209.165.201.10
ip forward-protocol nd
no ip http server
no ip http secure-server
ip route 10.0.0.0 255.0.0.0 172.16.0.1
ip route 192.168.0.1 255.255.255.255 198.51.100.1
ip route 192.168.0.2 255.255.255.255 198.51.100.2
ip route 192.168.0.3 255.255.255.255 198.51.100.3
ip route 192.168.0.4 255.255.255.255 198.51.100.4
ip route 192.168.0.5 255.255.255.255 198.51.100.5
ip route 192.168.0.6 255.255.255.255 198.51.100.6
ip route vrf Mgmt-intf 10.0.0.0 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 172.16.0.0 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 209.165.202.128 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 192.168.0.0 255.255.240.0 192.168.0.1
interface atm 2/2/1.20 point-to-point
 description MLPoA aal5mux (Single-Link Bundles) Session (PTA Mode)
```

```
 no atm enable-ilmi-trap
 pvc 20/10
  vbr-nrt 512 256 1
  encapsulation aal5mux ppp Virtual-Template18
```

Example: Configuring Multilink PPP over ATM Using AAL5 SNAP Encapsulation

The following example shows how to configure Multilink PPP over ATM (MLPoA) with AAL5 SNAP encapsulation on the PTA device:

```
class-map match-all ip-prec-1
 match ip precedence 1
policy-map mlp-child-lfi-policy
 class ip-prec-1
  priority percent 10
policy-map mlp-parent-250K
 class class-default
  shape average 250000
  service-policy mlp-child-lfi-policy
policy-map mlp-parent-10M
 class class-default
  shape average 10000000
  service-policy mlp-child-lfi-policy
interface virtual-template 18
 description MLPoEoA/MLPoA aal5mux/aal5snap (single-link bundle) Virtual Template
 ip address negotiated
 peer default ip address pool MLP-IPv4-Pool
 ppp max-failure 30
 ppp chap password 0 password1
 ppp multilink
 ppp multilink interleave
 ppp multilink endpoint magic-number
 ppp timeout retry 4
 service-policy output mlp-parent-250K
bba-group pppoe mlpoeoa-bba-group-250K
 virtual-template 18
ip local pool MLP-IPv4-Pool 209.165.201.2 209.165.201.10
ip forward-protocol nd
no ip http server
no ip http secure-server
ip route 10.0.0.0 255.0.0.0 172.16.0.1
ip route 192.168.0.1 255.255.255.255 198.51.100.1
ip route 192.168.0.2 255.255.255.255 198.51.100.2
ip route 192.168.0.3 255.255.255.255 198.51.100.3
ip route 192.168.0.4 255.255.255.255 198.51.100.4
ip route 192.168.0.5 255.255.255.255 198.51.100.5
ip route 192.168.0.6 255.255.255.255 198.51.100.6
ip route vrf Mgmt-intf 10.0.0.0 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 172.16.0.0 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 209.165.202.128 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 192.168.0.0 255.255.240.0 192.168.0.1
interface atm 2/2/1.22 point-to-point
 description MLPoA aal5snap (Single-Link Bundles) Session (PTA Mode)
 no atm enable-ilmi-trap
 pvc 20/14
  vbr-nrt 512 256 20
  encapsulation aal5snap
  protocol ppp Virtual-Template18
```

Example: Configuring Multilink PPP over Ethernet over ATM on the PTA Device

The following example shows how to configure Multilink PPP over Ethernet over ATM (MLPoEoA) with AAL5 SNAP encapsulation on the PTA device:

```
class-map match-all ip-prec-1
 match ip precedence 1
policy-map mlp-child-lfi-policy
 class ip-prec-1
  priority percent 10
policy-map mlp-parent-250K
 class class-default
  shape average 250000
  service-policy mlp-child-lfi-policy
policy-map mlp-parent-10M
 class class-default
  shape average 10000000
  service-policy mlp-child-lfi-policy
interface virtual-template 18
 description MLPoEoA/MLPoA aal5mux/aal5snap (single-link bundle) Virtual Template
 ip address negotiated
 peer default ip address pool MLP-IPv4-Pool
 ppp max-failure 30
 ppp chap password 0 password1
 ppp multilink
 ppp multilink interleave
 ppp multilink endpoint magic-number
 ppp timeout retry 4
 service-policy output mlp-parent-250K
bba-group pppoe mlpoeoa-bba-group-250K
 virtual-template 18
vc-class atm mlpoeoa-vc-class-250K
  protocol pppoe group mlpoeoa-bba-group-250K
  vbr-nrt 512 256 20
  encapsulation aal5snap
  create on-demand
  idle-timeout 30
ip local pool MLP-IPv4-Pool 209.165.201.2 209.165.201.10
ip forward-protocol nd
no ip http server
no ip http secure-server
ip route 10.0.0.0 255.0.0.0 172.16.0.1
ip route 192.168.0.1 255.255.255.255 198.51.100.1
ip route 192.168.0.2 255.255.255.255 198.51.100.2
ip route 192.168.0.3 255.255.255.255 198.51.100.3
ip route 192.168.0.4 255.255.255.255 198.51.100.4
ip route 192.168.0.5 255.255.255.255 198.51.100.5
ip route 192.168.0.6 255.255.255.255 198.51.100.6
ip route vrf Mgmt-intf 10.0.0.0 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 172.16.0.0 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 209.165.202.128 255.0.0.0 192.168.0.1
ip route vrf Mgmt-intf 192.168.0.0 255.255.240.0 192.168.0.1
interface atm 2/2/1.24 point-to-point
 description MLPoEoA aal5snap (Single-Link Bundles) Session (PTA Mode)
 no atm enable-ilmi-trap
 pvc 23/32
  class-vc mlpoeoa-vc-class-250K
  vbr-nrt 512 256 20
```

Example: Configuring Multilink PPP over LNS

Example: Configuring an LNS Device to Initiate and Receive L2TP Traffic

The following example shows how to set up a tunnel on the Gigabit Ethernet interface on which virtual private dialup network (VPDN) member links are negotiated and added to the Multilink PPP bundle that is cloned from virtual template 500:

```
aaa new-model
!
!
aaa authentication ppp default local
aaa authentication ppp TESTME group radius
aaa authorization network default local
aaa authorization network TESTME group radius
!
aaa session-id common
buffers small perm 15000
buffers mid perm 12000
buffers big perm 8000
!
vpdn enable
!
vpdn-group LNS_1
 accept-dialin
  protocol l2tp
  virtual-template 500
 terminate-from hostname LAC1-1
 local name LNS1-1
 lcp renegotiation always
 l2tp tunnel receive-window 100
 L2tp tunnel password 0 password1
 l2tp tunnel nosession-timeout 30
 l2tp tunnel retransmit retries 7
 l2tp tunnel retransmit timeout min 2
 l2tp tunnel retransmit timeout max 8
!
!
interface GigabitEthernet 2/0/0
 ip address 209.165.202.140 255.255.255.0
 negotiation auto
 hold-queue 4096 in
!
!
interface virtual-template 500
 ip unnumbered Loopback1
 peer default ip address pool pool-1
 ppp mtu adaptive
 ppp timeout authentication 100
 ppp max-configure 110
 ppp max-failure 100
 ppp timeout retry 5
keepalive 30
 ppp authentication pap TESTME
 ppp authorization TESTME
 ppp multilink
!
ip local pool pool-1 209.165.201.3 209.165.201.30
radius-server host 10.0.0.10 auth-port 1645 acct-port 1646 key password1
radius-server retransmit 0
```

Example: Configuring a LAC Device to Initiate and Receive L2TP Traffic

The following example shows how to set up a tunnel on a LAC device to initiate and receive traffic:

```
vpdn enable
vpdn search-order domain
!
vpdn-group cisco-1.com
 request-dialin
  protocol l2tp
  domain cisco-1.com
 initiate-to ip 10.0.0.4
 local name LAC1
 no l2tp tunnel authentication
!
vpdn-group cisco-2.com
 request-dialin
  protocol l2tp
  domain cisco-2.com
 initiate-to ip 10.0.0.4
 local name LAC2
 no l2tp tunnel authentication
!
vpdn-group cisco-3.com
 request-dialin
  protocol l2tp
  domain cisco-3.com
 initiate-to ip 10.0.0.4
 local name LAC3
 no l2tp tunnel authentication
!
vpdn-group cisco-4.com
 request-dialin
  protocol l2tp
  domain cisco-4.com
 initiate-to ip 10.0.0.4
 local name LAC4
 no l2tp tunnel authentication
!
bba-group pppoe cpe-lac-lns-group
 virtual-template 99
!
interface GigabitEthernet 0/0/0
 description GE connection to cpp-rtp-7200-41 0/1 (PTA-to-CPE)
 no ip address
 negotiation auto
!
interface GigabitEthernet 0/0/0.23
 description PPPoEoVLAN-to-PPPoLNS (non-MLP) Sessions CPE-LAC-LNS
 encapsulation dot1Q 23
 pppoe enable group cpe-lac-lns-group
!
interface GigabitEthernet 0/0/0.24
 description PPPoEoQinQ-to-PPPoLNS (non-MLP) Sessions CPE-LAC-LNS
 encapsulation dot1Q 24 second-dot1q 240
 pppoe enable group cpe-lac-lns-group
!
interface GigabitEthernet 0/2/0
 description GE connection to cpp-rtp-7200-41 0/2  (PTA-to-CPE)
 no ip address
 negotiation auto
!
interface GigabitEthernet 0/2/0.23
 description MLPPPoEoVLAN-to-MLPPPoLNS (Single Link Bundles) Sessions CPE-LAC-LNS
```

```
 encapsulation dot1Q 23
 pppoe enable group cpe-lac-lns-group
!
interface GigabitEthernet 0/2/0.24
 description MLPPPoEoQinQ-to-MLPPPoLNS (Single Link Bundles) Sessions CPE-LAC-LNS
 encapsulation dot1Q 24 second-dot1q 240
 pppoe enable group cpe-lac-lns-group
!
interface TenGigabitEthernet 1/0/0
 description TenGig connection to cpp-rtp-mcp6ru-01 1/0/0 (LAC-to-LNS)
 ip address 10.0.0.3 255.255.0.0
!
interface virtual-template 99
 description VT for PPPoE/MLPPPoE sessions from CPE being forwarded to LNS CPE-LAC-LNS
 no ip address
 no peer default ip address
 ppp mtu adaptive
 ppp authentication chap pap
```

Example: Configuring Multilink PPP over Serial Interfaces

The figure below shows a typical inverse multiplexing application using two Cisco routers and Multilink PPP over four T1 lines.

Figure 51: Inverse Multiplexing Application Using Multilink PPP



The example below shows the configuration commands that are used to create the inverse multiplexing application.

Router A Configuration

```
hostname RouterA
!
!
username RouterB password passwordA
ip subnet-zero
multilink virtual-template 1
!
interface virtual-template 1
 ip unnumbered Ethernet0
 ppp authentication chap
 ppp multilink
!
interface serial 0
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface serial 1
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface serial 2
```

```
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface serial 3
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface GigabitEthernet 0/0/0
 ip address 10.17.1.254 255.255.255.0
!
router rip
network 10.0.0.0
!
end
```

Router B Configuration

```
hostname RouterB
!
!
username RouterB password passwordA
ip subnet-zero
multilink virtual-template 1
!
interface virtual-template 1
 ip unnumbered Ethernet0
 ppp authentication chap
 ppp multilink
!
interface serial 0
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface serial 1
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface serial 2
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface serial 3
 no ip address
 encapsulation ppp
 no fair-queue
 ppp multilink
!
interface GigabitEthernet 0/0/0
 ip address 10.17.2.254 255.255.255.0
!
router rip
network 10.0.0.0
!
end
```

Additional References for Multilink PPP Connections for Broadband and Serial Topologies

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
PPP commands	Dial Technologies Command Reference
Broadband configuration tasks	*Broadband and DSL Configuration Guide*
Multilink PPP	*Multilink PPP Feature Functionality on the ASR 1000 Series Aggregation Services Router*

Standards and RFCs

Standard/RFC	Title
RFC 1990	*The PPP Multilink Protocol (MP)*
RFC 2686	*The Multi-Class Extension to Multi-Link PPP*

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

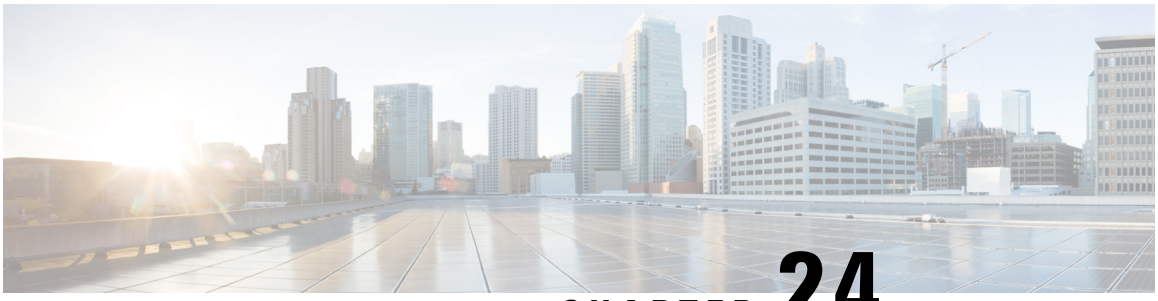
Feature Information for Multilink PPP Connections for Broadband and Serial Topologies

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 43: Feature Information for Multilink PPP Connections for Broadband and Serial Topologies

Feature Name	Releases	Feature Information
MLPoA and MLPoEoA—PTA	Cisco IOS XE Release 3.4S	Configuring Multilink PPP over broadband includes configuring Multilink PPP over ATM (MLPoA), Multilink PPP over Ethernet (MLPoE), Multilink PPP over Ethernet over ATM (MLPoEoA), Multilink PPP over Queue-in-Queue (MLPoQinQ), and Multilink PPP over VLAN (MLPoVLAN). For Multilink PPP over Serial, Multilink PPP over Ethernet, and Multilink PPP over LNS, see the Release Notes specific to your platform and release. This feature also provides support for Link Fragmentation and Interleaving (LFI) to meet additional needs of service providers to manage the latency of their delay-sensitive voice, video, and interactive application traffic on slower broadband links.



CHAPTER 24

MLPoE at PTA

The Multilink PPP over Ethernet (MLPoE) at PPP Termination and Aggregation (PTA) feature allows customer premises equipment (CPE) and PTA devices to interleave high-priority and low-latency packets (PPP encapsulated) between Multilink PPP fragments of lower-priority and higher-latency packets.

- Prerequisites for MLPoE at PTA, on page 437
- Restrictions for MLPoE at PTA, on page 437
- Information About MLPoE at PTA , on page 438
- How to Configure MLPoE at PTA, on page 439
- Configuration Examples for MLPoE at PTA, on page 442
- Additional References for MLPoE at PTA, on page 444
- Feature Information for MLPoE at PTA, on page 444

Prerequisites for MLPoE at PTA

Before configuring Multilink PPP over Ethernet (MLPoE) at PPP termination and aggregation (PTA), you must complete the following tasks:

- Creating a Class Map
- Creating a Policy Map
- Defining a PPP over Ethernet Profile
- Configuring a Virtual Template Interface

For more information see Configuring Multilink PPP over Broadband section.

Restrictions for MLPoE at PTA

- In-Service Software Upgrade (ISSU) and Stateful Switchover (SSO) for Broadband MLP sessions are not supported.
- Multilink PPP over Ethernet (MLPoE) using EtherChannel is not supported.
- Cisco IOS XE software supports a maximum of 4000 member links using MLPoE.
- For MLP virtual access bundles, the default Layer 3 (that is IP, IPv6) maximum transmission unit (MTU) value is 1500. When the member link of the MLPPP bundle are Ethernet-like in MLPoEoE, MLPoEoVLAN, and MLPoEoQinQ, the MTU value of 1500 can cause an issue when sending IP packets

close to this size. For example, when a 1500-byte IP packet is sent by a device over MLPoEoE, the actual packet size transmitted is 1522: 14 (Ethernet header) + 8 (PPPoE header) + 6 (MLP header) + 1500 (IP) = 1528. A device enforcing MRU might drop the incoming packet as a "giant" because it exceeds the default expected maximum packet size. The 1500-byte MTU size does not take into account any PPPoE or MLP header overhead and, hence, causes packets greater than 1492 bytes to be dropped by the peer. To address this issue, do one of the following:

- Lower the MTU on the MLP bundle to 1492.
- Increase the MTU on the Ethernet interface to 9216. Also, increase the MTU on the bundle by adjusting the MTU of the virtual template to 1508.

- Member Link Session bandwidth—For MLPoE PPP termination and aggregation (PTA) variations, by default the bandwidth of the member link session is that of the parent interface. If a bandwidth statement is added to the virtual template, the member link session uses that bandwidth as the member link session bandwidth. This bandwidth is in turn communicated to MLPPP in the bundle member link aggregate data rate bandwidth calculation.
- If the Digital Subscriber Line Access Multiplexer (DSLAM) between the CPE and PTA communicates the link rate via the PPPoE dsl-sync-rate tags (Actual Data-Rate Downstream [0x82/130d] tag), this data is passed by the PTA device to the RADIUS server but is not acted upon by the ASR 1000 device. The data rate of the session remains as described above in the previous bullet. Note that this behavior is specific to PTA mode; LAC/LNS behaves differently. Use the **dsl line info forwarding** command on the LAC to transport the LAC access speed to the LNS.

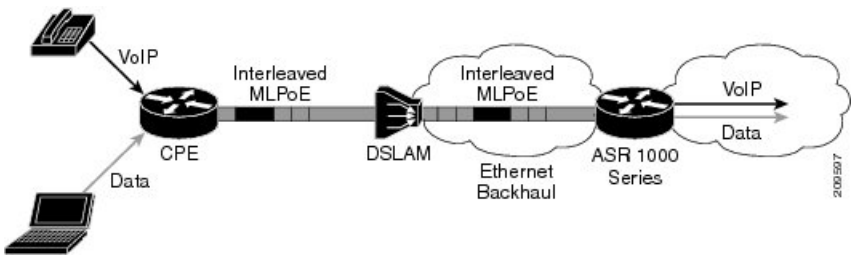
Information About MLPoE at PTA

MLPoE at PTA Overview

Single-link PPP over Ethernet and Multilink PPP over Ethernet (MLPoE) bundles support upstream and downstream link fragmentation and interleaving (LFI). Upstream refers to the traffic from the customer premises equipment (CPE) and downstream refers to the traffic to the CPE. The receiving device (CPE for downstream and PPP termination and aggregation [PTA] for upstream) reassembles fragmented, nonpriority packets. To reduce any delay in forwarding high-priority packets, the receiving device processes high-priority PPP packets as soon as they arrive.

The figure below shows a sample MLPoE network with LFI.

Figure 52: MLPoE DSL Network with LFI



PPP over Ethernet (PPPoE) sessions in MLPoE on a PTA device are handled as follows:

- All supported variations of PPPoE, such as PPP over Ethernet over ATM (PPPoEoA), PPP over Ethernet over Ethernet (PPPoEoE), PPP over Ethernet over Queue-in-Queue (PPPoEoQinQ), and PPP over Ethernet over VLAN (PPPoVLAN), can be used as member links for MLPoE bundles.

- Termination of an MLPoE bundle in a virtual routing and forwarding (VRF) block is similar to terminating a PPPoE session in a VRF instance.
- MLPoE bundles are distinguished by the username that was used to authenticate the PPPoE member link session.

How to Configure MLPoE at PTA

Configuring MLPoE at PTA

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **negotiation auto**
5. **pppoe enable group** *group-name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/1`	Specifies a Gigabit Ethernet interface for which Multilink PPP must be configured and enters interface configuration mode.
Step 4	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 5	**pppoe enable group** *group-name* **Example:** `Device(config-if)# pppoe enable group mlpoe-bba-group-10m`	Enables PPPoE sessions on an Ethernet interface or subinterface.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring MLPoE over VLAN

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **encapsulation dot1q** *vlan-id*
5. **pppoe enable group** *group-name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0.1`	Specifies the Gigabit Ethernet interface for which Multilink PPP must be configured and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 13`	Enables IEEE 802.1q encapsulation of traffic on the specified subinterface in VLANs. • *vlan-id* is the virtual LAN identifier. The range is from 1 to 1000.
Step 5	**pppoe enable group** *group-name* **Example:** `Device(config-subif)# pppoe enable group mlpoe-bba-group-10m`	Enables PPPoE sessions on the subinterface.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Configuring MLPoE over QinQ

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id* | *vlan-id-vlan-id* | [**,** *vlan-id-vlan-id*]}
5. **pppoe enable group** *group-name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/1.1`	Specifies the Gigabit Ethernet subinterface for which Multilink PPP must be configured and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **second-dot1q** {**any** | *vlan-id* | *vlan-id-vlan-id* | [**,** *vlan-id-vlan-id*]} **Example:** `Device(config-subif)# encapsulation dot1q 14 second-dot1q 140`	Enables IEEE 802.1q encapsulation of traffic on a specified subinterface in VLANs. • *vlan-id* is the Virtual LAN identifier. Enter a hyphen to separate the starting and ending VLAN ID values that are used to define a range of VLAN IDs. Optionally, enter a comma to separate each VLAN ID range from the next range. The range is from 1 to 4094.
Step 5	**pppoe enable group** *group-name* **Example:**	Enables PPPoE sessions on an Ethernet interface or subinterface.

	Command or Action	Purpose
	`Device(config-subif)# pppoe enable group mlpoe-bba-group-10m`	
Step 6	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Configuration Examples for MLPoE at PTA

Example: Configuring MLPoE at PTA

The following example shows how to configure the Multilink PPP over Ethernet (MLPoE) on the PTA device:

```
class-map match-all ip-prec-1
 match ip precedence 1
!
policy-map mlp-child-lfi-policy
 class ip-prec-1
  priority percent 10
policy-map mlp-parent-250K
 class class-default
  shape average 250000
  service-policy mlp-child-lfi-policy
policy-map mlp-parent-10M
 class class-default
  shape average 10000000
  service-policy mlp-child-lfi-policy
interface virtual-template 15
 description MLPoE/oEoVLAN/oEoQinQ (single-link bundle) Virtual Template
 ip address negotiated
 peer default ip address pool MLP-IPv4-Pool
 ppp max-failure 30
 ppp chap password 0 passowrd1
 ppp multilink
 ppp multilink interleave
 ppp multilink endpoint magic-number
 ppp timeout retry 4
 service-policy output mlp-parent-10M
bba-group pppoe mlpoe-bba-group-10M
 virtual-template 15
ip local pool MLP-IPv4-Pool 209.165.201.2 209.165.201.10
interface GigabitEthernet  0/0/0
 description MLPoE (Single-Link Bundles) Session (PTA Mode) to 7200-41 0/1
 no ip address
 negotiation auto
 pppoe enable group mlpoe-bba-group-10M
```

Example: Configuring MLPoE over VLAN

The following example shows how to configure Multilink PPP over Ethernet over VLAN (MLPoEoVLAN) on the PTA device:

```
class-map match-all ip-prec-1
 match ip precedence 1
```

```
policy-map mlp-child-lfi-policy
 class ip-prec-1
  priority percent 10
policy-map mlp-parent-250K
 class class-default
  shape average 250000
  service-policy mlp-child-lfi-policy
policy-map mlp-parent-10M
 class class-default
  shape average 10000000
  service-policy mlp-child-lfi-policy
interface virtual-template 15
 description MLPoE/oEoVLAN/oEoQinQ (single-link bundle) Virtual Template
 ip address negotiated
 peer default ip address pool MLP-IPv4-Pool
 ppp max-failure 30
 ppp chap password 0 password1
 ppp multilink
 ppp multilink interleave
 ppp multilink endpoint magic-number
 ppp timeout retry 4
 service-policy output mlp-parent-10M
bba-group pppoe mlpoe-bba-group-10M
 virtual-template 15
ip local pool MLP-IPv4-Pool 209.165.201.2 209.165.201.10
interface GigabitEthernet 0/0/0.13
 description MLPoEoVLAN Session (Single-Link Bundles) Session (PTA Mode)
 encapsulation dot1Q 13
 pppoe enable group mlpoe-bba-group-10M
```

Example: Configuring MLPoE over QinQ

The following example shows how to configure Multilink PPP over Ethernet over Queue-in-Queue (MLPoEoQinQ) on the PTA device:

```
class-map match-all ip-prec-1
 match ip precedence 1
policy-map mlp-child-lfi-policy
 class ip-prec-1
  priority percent 10
policy-map mlp-parent-250K
 class class-default
  shape average 250000
  service-policy mlp-child-lfi-policy
policy-map mlp-parent-10M
 class class-default
  shape average 10000000
  service-policy mlp-child-lfi-policy
interface virtual-template 15
 description MLPoE/oEoVLAN/oEoQinQ (single-link bundle) Virtual Template
 ip address negotiated
 peer default ip address pool MLP-IPv4-Pool
 ppp max-failure 30
 ppp chap password 0 password1
 ppp multilink
 ppp multilink interleave
 ppp multilink endpoint magic-number
 ppp timeout retry 4
 service-policy output mlp-parent-10M
bba-group pppoe mlpoe-bba-group-10M
 virtual-template 15
ip local pool MLP-IPv4-Pool 40.1.0.1 40.1.0.6
interface GigabitEthernet 0/0/0.14
```

```
description MLPoEoQinQ Sessions (Single-Link Bundles) Session (PTA Mode)
encapsulation dot1Q 14 second-dot1q 140
pppoe enable group mlpoe-bba-group-10M
```

Additional References for MLPoE at PTA

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
PPP commands	Dial Technologies Command Reference
Multilink PPP	*Multilink PPP Feature Functionality on the ASR 1000 Series Aggregation Services Router*

Standards and RFCs

Standard/RFC	Title
RFC 1990	*The PPP Multilink Protocol (MP)*
RFC 2686	*The Multi-Class Extension to Multi-Link PPP*

MIBs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

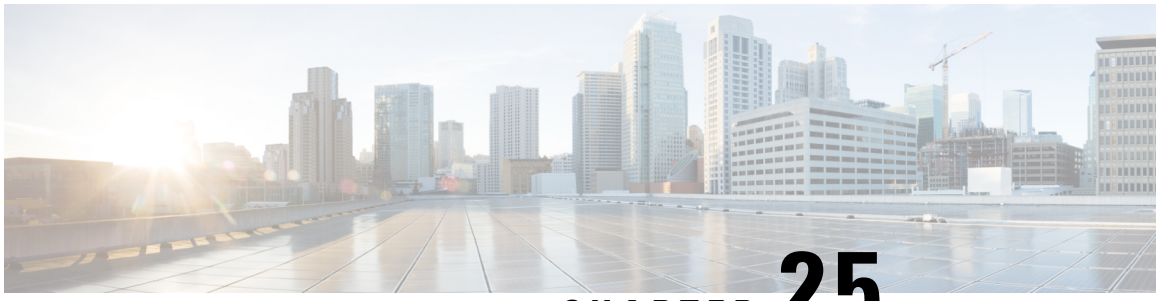
Feature Information for MLPoE at PTA

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 44: Feature Information for Multilink PPP Over Ethernet at PTA

Feature Name	Releases	Feature Information
MLPoE at PTA	12.2(33)XNE Cisco IOS XE Release 3.4S Cisco IOS XE Release 3.10S Cisco IOS XE Release 3.12S	Multilink PPP over Ethernet (MLPoE) at PPP Termination and Aggregation (PTA) feature allows the customer premises equipment (CPE) and PTA devices to interleave high-priority and low-latency packets (PPP encapsulated) between MLPPP fragments of lower-priority and higher-latency packets.



CHAPTER 25

Configurable CHAP Challenge Length

The Configurable Challenge Handshake Authentication Protocol (CHAP) Challenge Length feature allows you to configure the length of the CHAP challenge by specifying the minimum and maximum allowable challenge lengths in bytes.

- Prerequisites for Configurable CHAP Challenge Length, on page 447
- Information About Configurable CHAP Challenge Length, on page 447
- How to Configure Configurable CHAP Challenge Length, on page 448
- Configuration Examples for Configurable CHAP Challenge Length, on page 449
- Additional References for Configurable CHAP Challenge Length, on page 449
- Feature Information for Configurable CHAP Challenge Length, on page 450

Prerequisites for Configurable CHAP Challenge Length

The PPP encapsulation must be configured on the interface.

Information About Configurable CHAP Challenge Length

Configurable CHAP Challenge Length Overview

Challenge Handshake Authentication Protocol (CHAP) along with PPP is used to provide remote-device information to the central site. It verifies the identity of the peer by means of a three-way handshake.

When CHAP is enabled on any interface that supports PPP encapsulation, and a remote device attempts to connect to it, the local device or the access server sends a CHAP packet to the remote device. The CHAP packet requests or “challenges” the remote device to respond.

By default, the CHAP challenge is sent with a fixed 16-byte length to the peer. The Configurable CHAP Challenge Length feature allows the configuration of variable CHAP challenge lengths. A variable challenge length reduces the probability of an attacker predicting the challenge, thus optimizing the security.

Use the **ppp chap challenge-length** command to configure the CHAP challenge lengths.

How to Configure Configurable CHAP Challenge Length

Configuring Configurable CHAP Challenge Length

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface virtual-template** *number*
4. **ppp authentication chap**
5. **ppp chap challenge-length** *min-length max-length*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface virtual-template** *number* **Example:** `Device(config)# interface virtual-template 1`	Creates a virtual template interface and enters interface configuration mode. The range is from 1 to 4095.
Step 4	**ppp authentication chap** **Example:** `Device(config-if)# ppp authentication chap`	Enables CHAP authentication.
Step 5	**ppp chap challenge-length** *min-length max-length* **Example:** `Device(config-if)# ppp chap challenge-length 20 30`	Configures the minimum and maximum CHAP challenge lengths in bytes. The range is from 16 to 63.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for Configurable CHAP Challenge Length

Example: Configuring Configurable CHAP Challenge Length

The following example shows how to configure the Challenge Handshake Authentication Protocol (CHAP) challenge lengths:

```
Device> enable
Device# configure terminal
Device(config)# interface virtual-template 1
Device(config-if)# ppp authentication chap
Device(config-if)# ppp chap challenge-length 20 30
Device(config-if)# end
```

Additional References for Configurable CHAP Challenge Length

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
PPP commands	Dial Technologies Command Reference
Wide-area networking commands	Wide-Area Networking Command Reference

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configurable CHAP Challenge Length

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 45: Feature Information for Configurable CHAP Challenge Length

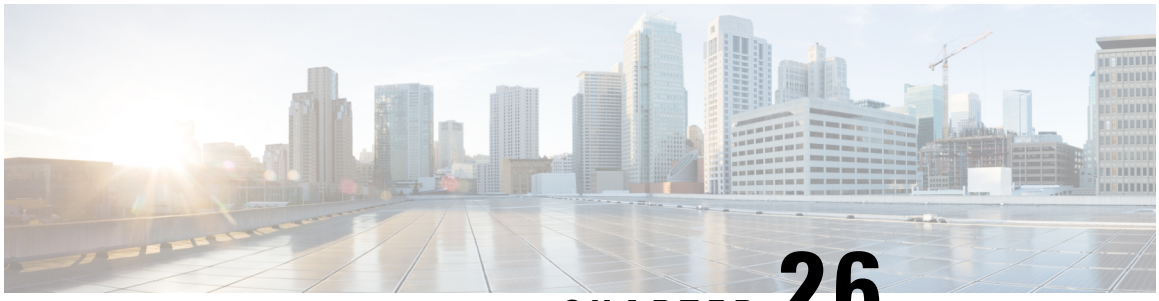
Feature Name	Releases	Feature Information
Configurable CHAP Challenge Length	Cisco IOS XE Release 3.12S	The Configurable Challenge Handshake Authentication Protocol (CHAP) feature allows you to configure the length of the CHAP challenge by specifying the minimum and maximum allowable challenge length in bytes. The following command was introduced: **ppp chap challenge-length**.



PART VI

Overlay Transport Virtualization

- Configuring Overlay Transport Virtualization, on page 453
- OTV Adjacency Server, on page 491



CHAPTER 26

Configuring Overlay Transport Virtualization

Overlay Transport Virtualization (OTV) is a MAC-in-IP method that extends Layer 2 connectivity across a transport network infrastructure. OTV provides Layer 2 connectivity between remote network sites by using MAC-address-based routing and IP-encapsulated forwarding across a transport network to provide support for applications that require Layer 2 adjacency.

The OTV application (also known as OTV) is one of the modules of the OTV architecture in Cisco software. OTV interacts with the following other modules of the OTV architecture in Cisco IOS software:

- Layer 2 Intermediate System-to-Intermediate System (IS-IS)
- Ethernet infrastructure
- IP tunnel infrastructure
- Layer 2 Forwarding Information Base (L2FIB)
- Multilayer Routing Information Base (MLRIB)
- Ethernet Operation, Administration, and Maintenance (OAM)
- Internet Group Management Protocol (IGMP)
- Address Resolution Protocol (ARP)

You deploy OTV on edge devices in each site. OTV requires no other changes to the sites or to the transport network.

- Prerequisites for OTV, on page 453
- Restrictions for OTV, on page 454
- Information About OTV, on page 454
- How to Configure OTV, on page 464
- Configuration Examples for OTV Features, on page 476
- Verifying the OTV Configuration, on page 484
- Additional References, on page 486
- Feature Information for OTV, on page 486

Prerequisites for OTV

You must have basic understanding of routing, switching, and multicast concepts.

Restrictions for OTV

- Configure the join interface and all Layer 3 interfaces that face the IP core between the OTV edge devices with the highest maximum transmission unit (MTU) size supported by the IP core. OTV sets the Don't Fragment (DF) bit in the IP header for all OTV control and data packets so that the core cannot fragment these packets.
- Ensure that PIM is not enabled on the join interface; enable only passive PIM on the join interface. Configure SSM for the OTV data group multicast address range by using the **ip pim passive** command.
- Ensure that a site identifier is configured and is the same for all edge devices in a same site. OTV brings down all overlays and generates a system message when it detects a mismatched site identifier from a neighbor edge device.
- Only one internal interface (site-facing interface) can be configured on an edge device for all the overlays, multiple internal interface's are not supported.
- Only physical interfaces/sub-interfaces/port-channel can be used as join-interfaces, GRE tunnels and loopback interfaces are not supported.
- OTV is compatible only with a transport network configured for IPv4. IPv6 is not supported.
- OTV cannot be configured on the same router on which Multiprotocol Label Switching (MPLS) is configured. If you try to configure an overlay interface on a router where MPLS is already configured, OTV creation will fail. Similarly, if you try to create an MPLS on a router where OTV is configured, MPLS creation will fail. You can remove a failed overlay interface configuration by using the **no interface overlay x** command.
- The transport network must support the Protocol Independent Multicast (PIM) sparse mode (Any Source Multicast [ASM]) for the provider multicast group and Source Specific Multicast (SSM) for the delivery group.
- If the device is not configured with OTV, the show bridge-domain command does not display any output.
- Overlay Transport Virtualization (OTV) and Cisco Unified Border Element (Cisco UBE) cannot interoperate with each other on Cisco IOS XE software.

Information About OTV

Functions of OTV

- Maintains a list of overlays
- Maintains a list of configured overlay parameters such as name, multicast address, encapsulation type, authentication, and OTV feature sets
- Maintains the state of the overlay interface
- Maintains the status of OTV VLAN membership from Ethernet infrastructure and the state of the authoritative edge device (AED) from IS-IS

- Maintains a database of overlay adjacencies as reported by IS-IS
- Maintains IP tunnel information and manages the encapsulation for data sent on the overlay network
- Manages delivery groups (DGs) for each overlay by snooping multicast traffic and monitoring traffic streams for active DGs
- Configures, starts, and stops the OTV IS-IS instance
- Interfaces with IP multicast to join provider multicast groups for each overlay

OTV Terms

Table 46: OTV Terms

Term	Description
Edge device	A device that performs typical Layer 2 learning and forwarding on it's internal interface (site-facing interface) and IP-based virtualization on transport-facing interfaces. OTV functionality occurs only in an edge device. You can configure multiple overlay interfaces on an edge device. You can also have multiple edge devices in the same site.
Authoritative edge device (AED)	An elected edge device that serves as the forwarder. OTV elects a forwarding device per site for each VLAN and designates the forwarding device as an AED. OTV provides loop-free multihoming by using this AED. The edge devices in a site communicate with each other through internal interfaces to elect an AED.
Transport network	A network that connects OTV sites. A transport network can be managed by customers or provided by a service provider or be a mix of both. OTV is compatible only with a transport network configured for IPv4. IPv6 is not supported.
Join interface	An uplink interface of an edge device. A join interface is a point-to-point routed interface. An edge device joins an overlay network through this interface. The IP address of this interface is used to advertise the reachability of a MAC address in a site. Both Ethernet and Packet over SONET (POS) interfaces are supported as join interfaces. The join interface connects the edge device to the transport network and it should be a Layer-3 interface.
Internal interface (site-facing interface)	A Layer 2 interface on an edge device that connects to the VLANs that are to be extended. These VLANs typically form a Layer 2 domain known as a site and can contain site-based switches or site-based routers. An internal interface is a Layer 2 access interface or a trunk interface regardless of whether the internal interface connects to a switch or a router.

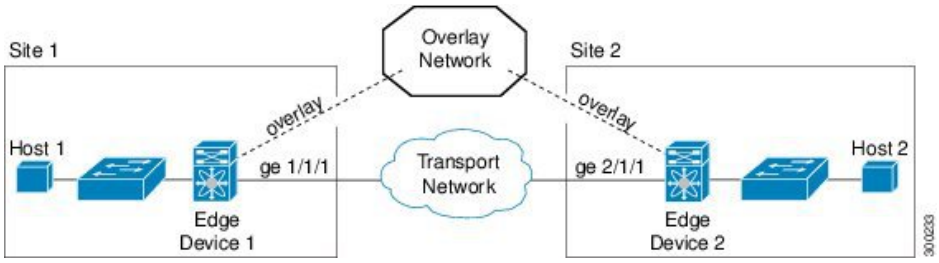
Term	Description
MAC-in-IP Routing	The type of routing that associates the destination MAC address of the Layer 2 traffic with the join interface IP address of an edge device. The OTV control-plane protocol advertises the MAC-in-IP association to edge devices. In MAC routing, MAC addresses are reachable in an overlay network through the IP address of a remote edge device. Layer 2 traffic that is destined to a MAC address is encapsulated in an IP packet based on the MAC-in-IP mapping stored in the MAC table.
Overlay interface	A logical, multiaccess, multicast-capable interface. An overlay interface encapsulates Layer 2 frames in IP unicast or multicast headers.
Overlay network	A logical network that interconnects remote sites for MAC routing of Layer 2 traffic. An overlay network comprises multiple edge devices.
Site	A Layer 2 network that may be single-homed or multihomed to the transport network and the OTV overlay network. Edge devices that operate in an overlay network provide Layer 2 connectivity between sites. Layer 2 sites are physically separated from each other by the transport network.
Site VLAN	A dedicated VLAN on which an internal adjacency is established. OTV sends local hello messages on the site VLAN to detect other OTV edge devices in the same site. OTV also uses the site VLAN to determine the AED within edge devices in the same site. We recommend that you use a dedicated VLAN as a site VLAN. You should also ensure the following: • Site VLAN should be active on the internal interface of the edge device. • Site VLAN is not extended across the overlay.

OTV Overlay Network

An OTV overlay network provides Layer 2 connectivity between remote sites over a transport network. An overlay network consists of one or more edge devices in each site. The sites are interconnected using a control-plane protocol across the transport network.

The figure below shows two sites connected through edge devices to a transport network to create a virtual overlay network.

Figure 53: OTV Overlay Network



An overlay network maps MAC addresses of the hosts at a site to their respective edge devices IP addresses. After OTV identifies the edge device to which a Layer 2 frame is to be sent, OTV encapsulates the frame and sends the resulting IP packet by using the transport network routing protocols.

OTV can support more than one overlay network running IPv4 unicast forwarding or multicast flooding. Each overlay network can support more than one unique VLAN.

Note OTV does not extend Spanning Tree Protocol (STP) across sites. Each site runs its own STP instead of all sites being included in a large STP domain. This per-site STP topology allows the use of different STP modes, such as Per-VLAN Spanning Tree (PVST), Rapid-PVST, or Multiple Spanning Tree (MST), in each site.

Edge Devices

Each site consists of one or more edge devices and other internal routers, switches, or servers. OTV is configured only on an edge device. The OTV configuration is completely transparent to the rest of the site. For example, information about MAC learning, STP root bridge placement, and STP mode is transparent. An edge device has an internal interface that is part of the Layer 2 network and an external interface that is reachable through IP in the transport network.

An edge device performs typical Layer 2 learning and forwarding on its internal interface and transmits and receives encapsulated Layer 2 traffic on it's join interface through the transport network. An edge device sends and receives control-plane traffic through the join interface. The control-plane traffic exchanges reachability information between remote sites to build up a table that maps MAC addresses to the join interface IP address of the edge device that is local to that site.

Site-to-Site Connectivity

OTV builds Layer 2 reachability information by communicating between edge devices with the overlay protocol. The overlay protocol forms adjacencies with all edge devices. After each edge device is adjacent with all its peers in an overlay network, the edge devices share MAC address reachability information with other edge devices that participate in the same overlay network.

OTV discovers edge devices through dynamic neighbor detection, which leverages the multicast support of the core.

Overlay Networks Mapping to Multicast Groups

For transport networks that support IP multicast, one multicast address (the control-group address) is used to encapsulate and exchange OTV control-plane protocol updates. Each edge device that participates in a particular overlay network shares the same control-group address with all other edge devices of the same overlay network.

As soon as a control-group, data-group address and a join interface are configured on an edge device, the edge device sends an IGMP report message to join the control group. Edge devices act as hosts in the multicast network and send multicast IGMP report messages to the assigned multicast group address.

As in traditional link-state routing protocols, edge devices exchange OTV control-plane hellos to build adjacencies with other edge devices in the overlay network. After adjacencies are established, OTV control-plane link-state packets (LSPs) communicate MAC-to-IP mappings to adjacent edge devices. These LSPs contain the IP address of the remote edge device, VLAN IDs, and the learned MAC addresses that are reachable through that edge device.

Edge devices participate in data-plane learning on internal interfaces to build up the list of MAC addresses that are reachable within their site. OTV sends these locally learned MAC addresses in the OTV control-plane updates.

OTV Packet Flow

When an edge device receives a Layer 2 frame on an internal interface, OTV performs the MAC table lookup based on the destination address of the Layer 2 frame. If the frame is destined to a MAC address that is reachable locally, the frame is internally forwarded to that device. OTV performs no other actions and the processing of the frame is complete.

If the frame is destined to a MAC address learned over an overlay network, OTV performs the following tasks:

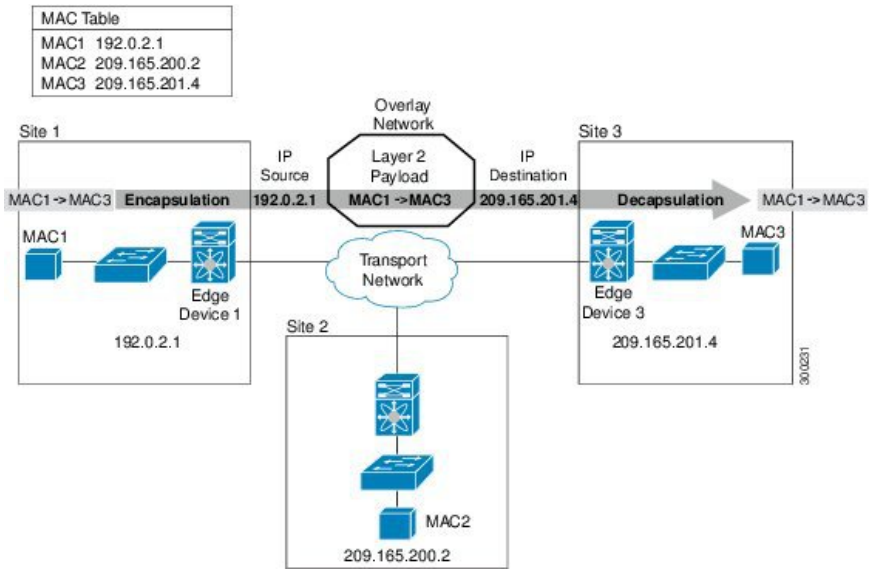
1. Strips the preamble and frame check sequence (FCS) from the Layer 2 frame.
2. Adds an OTV header to the Layer 2 frame and copies the 802.1Q information into the OTV header.
3. Adds the IP address to the packet based on the initial MAC address table lookup. This IP address is used as the destination address for the IP packet that is sent over the transport network.

OTV traffic appears as IP traffic to the transport network.

At the destination site, the edge device performs the reverse operation and presents the original Layer 2 frame to the local site. The edge device, based on the local MAC address table forwards the frame on its internal interface.

The figure below shows the encapsulation and decapsulation of a MAC-routed packet across an overlay network.

Figure 54: MAC Routing



In the figure above, Site 1 communicates with Site 3 over the overlay network. Edge Device 1 receives the Layer 2 frame from MAC1 and looks up the destination MAC address, MAC3, in the MAC table. The edge device encapsulates the Layer 2 frame in an IP packet with the IP destination address set for Edge Device 3 (209.165.201.4). When Edge Device 3 receives the IP packet, the edge device strips off the IP header and sends the original Layer 2 frame over it's internal interface to reach the host having MAC address MAC3.

Mobility

OTV uses a metric value to support seamless MAC mobility.

When an AED learns a new MAC address, the AED advertises the new address in OTV control-plane updates with a metric value of one if no other edge device has advertised that MAC address before.

In the case of a mobile MAC address, an AED advertises the newly learned, local MAC address with a metric value of zero. This metric value signals the remote edge device to stop advertising that MAC address. After the remote edge device stops advertising the moved MAC address, the AED that contains the new MAC address changes the metric value to one.

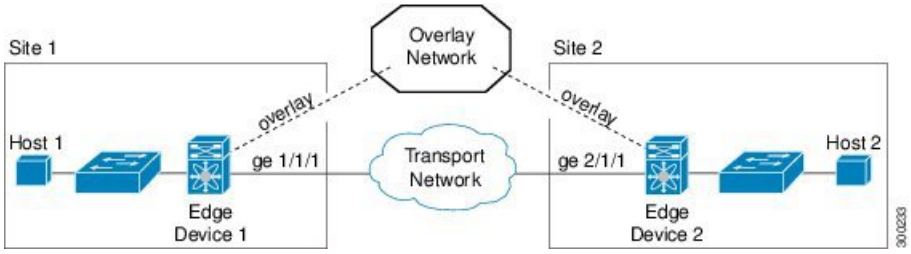
Virtual machine (VM) mobility is one common example of MAC mobility. VM mobility occurs when the virtual machine moves from one site to another. OTV detects this change based on the changed advertisement of the mobile MAC address.

Sample OTV Topologies

You can use OTV to connect remote sites in multiple topologies.

Single-Homed Network

Figure 55: Basic Two-Site OTV Topology

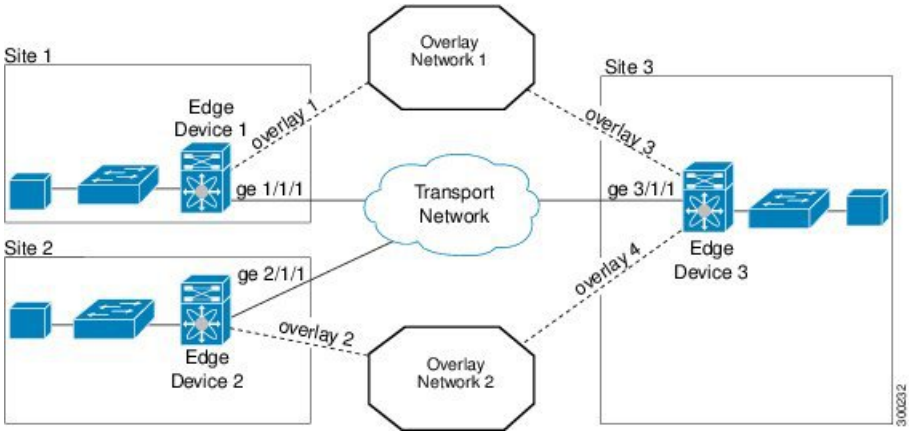


In this sample topology, both sites are connected over a common transport network. The edge devices in both the sites have an overlay interface configured (interface overlay 1 and interface overlay 2) with the same control-group address, which makes both the edge devices join a common overlay network.

Multiple Overlay Networks

You can configure an edge device in more than one overlay network. Each overlay network use different control and data group multicast addresses.

Figure 56: Two Overlay Networks



In the figure above, Site 3 connects to Site 1 over Overlay Network 1 through overlay interface 3 on Edge Device 3. Site 3 also connects to Site 2 over Overlay Network 2 through overlay interface 4 on Edge Device 3. Each overlay network has different control-group and data-group addresses.

Note The VLAN's extended across different overlay network's should be unique.

Site 3 uses Edge Device 3 to connect to both the overlay networks—Overlay Network 1 and Overlay Network 2. Edge Device 3 associates the same physical interface for both the overlay networks.

Multihomed Sites and Load Balancing

For resiliency and load balancing, a site can have multiple edge devices.

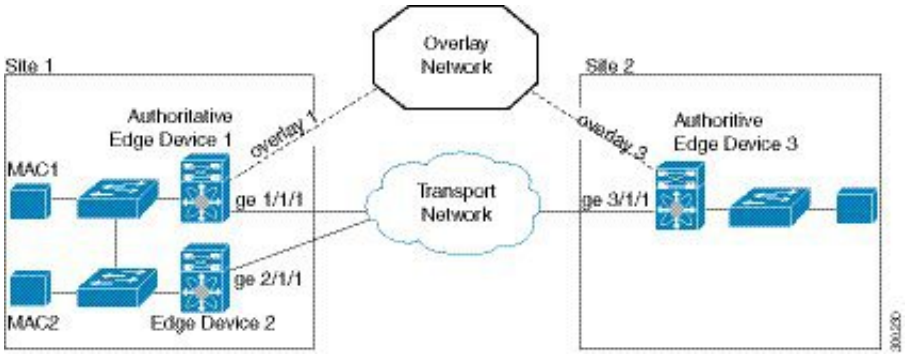
When more than one edge device exists in a site and both participate in the same overlay network, the site is considered multihomed. For the VLANs that are extended using OTV, one edge device is elected as an AED

on a per-VLAN basis. OTV leverages a local VLAN to establish an adjacency between edge devices on their internal interfaces. The local VLAN that is shared by the internal interfaces is the site VLAN. The adjacency establishment over the site VLAN determines which edge device is authoritative for what VLANs.

Load balancing is achieved because each edge device is authoritative for a subset of all VLANs that are transported over the overlay. Link utilization to and from the transport is optimized.

The figure below shows the AED that is selected for a multihomed site in an overlay network.

Figure 57: Multihomed Site



In the figure above, Site 1 is a multihomed site with two physical interfaces connected to the transport network.

Dual Site Adjacency

Dual site adjacency includes adjacency discovery over the overlay network and in the existing site VLAN. Dual site adjacency introduces additional resiliency and loop prevention. Loops may be caused by site VLAN partition or misconfiguration. Dual site adjacency also uses forwarding readiness notifications to detect when neighbor edge devices in the same site experience a change such as local failures (for example, the site VLAN or extended VLANs going down or the join interface going down). These forwarding readiness notifications trigger an immediate AED election for the site.

OTV sends forwarding readiness notifications to all neighbors of an edge device in the following isolation states:

- Site isolation: All extended VLANs on an edge device go down.
- Core isolation: All overlay adjacencies go down.

The dual site adjacency state results from the most recent adjacency state for either the overlay or the site VLAN adjacency. OTV determines AED election based on active dual site adjacencies only. An inactive dual site adjacency is ignored for AED election.

You must configure the same site identifier for all edge devices in a site. OTV advertises this site identifier in the IS-IS hello packets sent over the overlay network and on the local site VLAN. The combination of the IS-IS system ID and site identifier uniquely identifies the edge devices in a site.

OTV Features

The OTV control-plane creates adjacencies between remote sites to provide Layer 2 connectivity over a transport network. An OTV network performs the following functions:

- Discovers remote sites and builds a control-protocol adjacency

- Shares MAC routing information across an overlay network

An overlay network consists of one or more logical overlay interfaces that are configured on an edge device in each remote site that connects to the physical transport network. You associate the logical overlay interface with a physical interface (join-interface) that connects to the transport network. The OTV control plane is responsible for discovering edge devices in remote sites, creating control-protocol adjacencies to these sites, and establishing protocol adjacencies among the sites. The OTV control-plane protocol uses the IS-IS protocol to establish adjacencies and exchange MAC reachability across an overlay network.



Note You do not need to configure IS-IS to use OTV. IS-IS is enabled in the background after OTV is enabled.

The OTV control-plane protocol also sends and receives MAC routing updates between remote sites and updates the Routing Information Base (RIB) with these MAC-to-IP address pairs.

Overlay Interface

An overlay interface is a logical interface that connects to remote edge devices in an overlay network through an associated physical interface (join interface) on the transport network. From the perspective of MAC-based forwarding in a site, an overlay interface is simply another bridged interface. As a bridged interface, unicast MAC addresses are associated with an overlay interface. An overlay interface is eligible for inclusion in the Outbound Interface List (OIL) for different multicast groups. However, no STP packets are forwarded over an overlay interface. Unknown unicast packets are also not flooded on an overlay interface. From the perspective of IP transport, an overlay interface is not visible.

OTV encapsulates Layer 2 frames in IP packets and transmits them over the transport network via the join interface.

The following commands must be configured for an overlay interface to be in the up state:

- **no shutdown**
- **otv control-group**
- **otv data-group**
- **otv join-interface**

MAC Address Learning

OTV learns MAC-to-IP address pairs from the following:

- MAC address learning on internal interface
- OTV control-plane updates over an overlay network
- Multicast address learning through IGMP snooping

OTV edge devices snoop IGMP traffic and issue a Group Membership-link-state packet (GM-LSP) to advertise the presence of receivers to remote edge devices. The remote edge devices include the overlay interface in the Outbound Interface List (OIL) for the corresponding multicast group. OTV does not program multicast MAC addresses in the forwarding tables but rather updates the OIL state as necessary.

All learned MAC addresses are stored in the RIB with the associated remote IP addresses.

MAC Address Reachability Updates

The OTV control plane uses IS-IS link-state packets (LSPs) to propagate MAC address to IP address mappings to all edge devices in an overlay network. These address mappings contain the MAC address, VLAN ID, and the associated IP address of the edge device that the MAC address is reachable from.

An AED uses IGMP snooping to learn all multicast IP addresses in the local site. OTV includes these IP addresses in a special GM-LSP that is sent to remote edge devices in an overlay network.

Multicast Group Addresses and IGMP Snooping

OTV uses the control-group multicast address that is assigned from the transport network to create a unique multicast group between remote sites on an overlay network. Each edge device in an overlay network acts as a multicast host and sends an IGMP report message to join the control-group multicast address. OTV sends encapsulated OTV control-plane hello messages and MAC routing updates across this multicast group.

OTV uses IGMP snooping and group membership advertisements (GM-LSPs) to learn all multicast group members from remote sites. OTV also uses IGMP snooping to detect all multicast groups in a local site.

ARP Cache

OTV can suppress unnecessary ARP messages from being sent over an overlay network. OTV builds a local Layer 3-to-Layer 2 mapping for remote hosts. Any ARP requests from local hosts are served by this ARP cache.

High Availability

OTV supports stateless switchovers. A stateful switchover occurs when the active supervisor switches to the standby supervisor. There may be a few seconds of traffic loss while the OTV tunnel is recreated following a switchover.

OTV IS-IS

OTV uses the IS-IS protocol for control-plane learning of MAC entries. The OTV IS-IS component is responsible for transporting MAC information across all VPN sites. It carries unicast and multicast MAC information encoded in type, length, values (TLVs).

On the internal interface, OTV IS-IS is responsible for sending IS-IS hello (IIH) packets on the site VLAN by using a multicast MAC destination address. Using a multicast MAC address ensures that all Layer 2 switches in a site forward the packet and that the packet reaches all other OTV edge devices. Each site has a configured site ID. The site ID is advertised by each edge device in these IS-IS hello messages. The site ID is used to identify all edge devices belonging to the same site. IS-IS assigns an AED for each VLAN. The AED for a VLAN is the edge device responsible for announcing local MACs for a given VLAN to remote sites and accepting packets destined for that VLAN.

On the overlay interface, OTV IS-IS is responsible for sending out IIH packets with site ID TLV on the multicast control-group. Using the control-group multicast ensures that all remote sites participating in the overlay network are automatically discovered and an adjacency is formed among all edge devices belonging to the same overlay network. OTV IS-IS also informs OTV whenever a new neighbor is discovered.

OTV IS-IS also handles fast MAC moves between remote sites and the local site and guards against fast oscillations in the event of misconfigurations where the same MAC address is used in multiple sites.

OTV IS-IS Instances

The creation of an overlay interface triggers the creation of an OTV IS-IS instance. OTV IS-IS supports multiple overlays. There is a one-to-one relationship between an OTV IS-IS instance and an overlay interface. OTV IS-IS discovers neighbors, forms adjacencies, and exchanges unicast MAC and multicast group information per overlay. All IS-IS databases, such as the adjacency database and the LSP database, are maintained per overlay.

OTV IS-IS forms only level-1 adjacencies. It advertises the primary IP address of the join interface in its hellos and protocol data units (PDUs). This address along with the system ID of the neighbor is added to OTV, which stores this information in its overlay adjacency database.

OTV IS-IS MLRIB Interactions

OTV IS-IS is a client of Multilayer Routing Information Base (MLRIB) for Layer 2. OTV IS-IS registers with MLRIB to get notifications for all local Layer 2 unicast and multicast address additions or deletions. Unicast MAC address information is put in OTV IS-IS LSPs, while multicast address information is put in OTV IS-IS multicast group PDUs for flooding to all remote sites.

Based on neighbor LSP advertisements, OTV IS-IS adds MAC reachability information for remote unicast and multicast group addresses to MLRIB. When OTV is disabled on a VLAN (the VLAN is removed from the list of OTV-advertised VLANs), OTV IS-IS withdraws the remote reachability information from MLRIB.

How to Configure OTV

Creating an Overlay Interface

An overlay interface is a logical interface that connects to remote edge devices in an overlay network through an associated physical or port-channel interface (join-interface) on the transport network. After creating an overlay interface, you must associate the overlay interface with a join interface and configure control and data-group multicast addresses. For more information, see the “Associating an Overlay Interface with a Join Interface” and “Configuring a Multicast Group Address” sections.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **no shutdown**
5. **otv vpn-name** *name*
6. **description** *string*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**no shutdown** **Example:** `Device(config-if)# no shutdown`	Brings up the OTV overlay interface.
Step 5	Required: **otv vpn-name** *name* **Example:** `Device(config-if)# otv vpn-name overlay1`	(Optional) Creates an alias for the OTV overlay interface name. • The alias name is case-sensitive and must be no more than 20 alphanumeric characters in length.
Step 6	Required: **description** *string* **Example:** `Device(config-if)# description site4`	(Optional) Adds a description for the overlay network. • The description string can be up to 200 characters in length.
Step 7	Required: **end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Associating an Overlay Interface with a Physical Interface

Before you associate an overlay interface with a physical interface, ensure that IGMP Version 3 (IGMPv3) is configured on the physical Layer 3 interface that you configure as the join interface.

After creating an overlay interface, perform this task to associate the overlay interface with a join interface. Define a physical Layer 3 interface as the join interface for the overlay interface, and associate the same with the overlay interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **otv join-interface** *type number*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**otv join-interface** *type number* **Example:** `Device(config-if)# otv join-interface gigabitethernet 0/0/2`	Joins the OTV overlay interface with a physical Layer 3 interface. • You must configure an IP address on the physical interface. • You can specify only one join interface per overlay. • A single join interface on a edge device is shared across all overlays.
Step 5	Required: **end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

What to do next

To enable unicast and multicast IP forwarding on a join interface, perform the following tasks after creating the join interface:

- Configure the IP address and mask for the join interface by using the **ip address** command.
- Configure the join interface to operate in Protocol Independent Multicast (PIM) passive mode by using the **ip pim passive** command.
- Enable IP multicast routing by using the **ip multicast-routing distributed** command.
- Configure IGMPv3 on the join interface by using the **ip igmp version 3** command.
- The following commands are also necessary to be added globally to ensure multicast forwarding happens - these commands enable IGMP snooping on the internal interface:

  ip igmp snooping querier version 3

  ip igmp snooping querier

Configuring a Multicast Group Address

Perform this task to configure a unique multicast group address for each overlay network.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **otv control-group** *multicast-address*
5. **otv data-group** *multicast-address/mask*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**otv control-group** *multicast-address* **Example:** `Device(config-if)# otv control-group 239.1.1.1`	Configures a multicast group address used by the OTV control plane for this OTV overlay network. • The multicast group address is an IPv4 address in dotted decimal notation.
Step 5	**otv data-group** *multicast-address/mask* **Example:** `Device(config-if)# otv data-group 232.1.1.0/28`	Configures one or more ranges of local IPv4 multicast data-group prefixes used for multicast data traffic. • Use SSM multicast groups 232.0.0.0/8. • Enable SSM for the groups by using the **ip pim ssm** command in global configuration mode. • The multicast group address is an IPv4 address in dotted decimal notation. • A subnet mask is used to indicate ranges of addresses. • You can define up to 8 data-group ranges.

	Command or Action	Purpose
Step 6	Required: **end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring a VLAN over an Overlay Interface

Before configuring a VLAN over an overlay interface, ensure that there is connectivity for VLANs to be extended to the OTV edge device.

Ethernet service instances are configured with VLAN encapsulation on an overlay interface to define the VLANs that are part of an overlay network. MAC addresses learned on the service instances' bridge domains are advertised to other edge devices on the overlay along with the service instances' VLAN.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **service instance** *interface* **ethernet**
5. **encapsulation dot1q** *vlan-ID*
6. **bridge-domain** *bridge-domain-ID*
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**service instance** *interface* **ethernet** **Example:** `Device(config-if)# service instance 20 ethernet`	Configures an Ethernet service instance on the overlay interface being configured and enters service instance configuration mode. • The service instance identifier range is from 1 to 8000.

	Command or Action	Purpose
Step 5	**encapsulation dot1q** *vlan-ID* **Example:** `Device(config-if-srv)# encapsulation dot1q 20`	Defines the VLAN encapsulation format as IEEE 802.1Q and specifies the VLAN identifier.
Step 6	**bridge-domain** *bridge-domain-ID* **Example:** `Device(config-if-srv)# bridge-domain 20`	Binds the specified bridge domain to a service instance.
Step 7	**end** **Example:** `Device(config-if-srv)# end`	Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring the Site Bridge Domain and the Site Identifier

A site bridge domain is used by OTV to identify the service instance where local hello messages should be sent. There should be an Ethernet service instance configured with the site bridge domain on the internal interface. OTV uses the configured VLAN encapsulation (if any) from this service instance to encapsulate local hello messages before sending out a message from the local interface.

A site identifier is advertised by each edge device in an overlay network and is used to identify all edge devices belonging to the same site. All edge devices in the same site should be configured with the same site identifier.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **otv site bridge-domain** *bridge-domain-ID*
4. **exit**
5. **otv site-identifier** *site-ID*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**otv site bridge-domain** *bridge-domain-ID* **Example:** `Device(config)# otv site bridge-domain 10`	Configures the site bridge domain for sending IS-IS hellos over site interfaces and enters OTV site configuration mode. • The *bridge-domain-ID* range is from 1 to 4096. • One Ethernet service instance should be configured on the internal interface with the same bridge domain ID.
Step 4	**exit** **Example:** `Device(config-otv-site)# exit`	Exits OTV site configuration mode and returns to global configuration mode.
Step 5	**otv site-identifier** *site-ID* **Example:** `Device(config)# otv site-identifier 0000.0000.0001`	Configures the site identifier. • The same site identifier on all OTV edge devices should be configured belonging to the same site. • The site identifier should be unique across different sites. • The range is from 0x1 to 0xFFFFFFFF. The format is either hexadecimal or MAC address format.
Step 6	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Example

The following sample output shows the configuration of overlay interface 1:

```
Device#show running-config interface gigabitEthernet 0/0/2
!
interface GigabitEthernet0/0/2
 description "Join Interface"
 ip address 209.165.201.1 255.255.255.224
 ip pim passive
 ip igmp version 3
 negotiation auto
end

Device#

Device#show running-config interface GigabitEthernet0/0/0
!
interface GigabitEthernet0/0/0
 description "Internal Interface"
 no ip address
 negotiation auto
 service instance 10 ethernet
  encapsulation dot1q 10
  bridge-domain 10
 !
```

```
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 20
 !
Device#
```

```
Device#show otv overlay1
Overlay Interface Overlay1
 VPN name                 : overlay1
 VPN ID                   : 1
 State                    : UP
 Fwd-capable              : Yes
 Fwd-ready                : Yes
 AED-Server               : Yes
 Backup AED-Server        : No
 AED Capable              : Yes
 IPv4 control group       : 239.1.1.1
 Mcast data group range(s): 232.1.1.0/28
 Join interface(s)        : GigabitEthernet0/0/2
 Join IPv4 address        : 209.165.201.1
 Tunnel interface(s)      : Tunnel0
 Encapsulation format     : GRE/IPv4
 Site Bridge-Domain       : 10
 Capability               : Multicast-reachable
 Is Adjacency Server      : No
 Adj Server Configured    : No
 Prim/Sec Adj Svr(s)      : None

Device#
```

```
Device#show otv overlay 1 vlan

Overlay 1 VLAN Configuration Information
 Inst VLAN BD   Auth ED              State               Site If(s)
 0    20 20 *Device                  active              Gi0/0/0:SI20
 Total VLAN(s): 1

Device#
```

Configuring Authentication for OTV IS-IS Hellos

You can configure authentication for OTV IS-IS hello messages. OTV uses hello authentication to authenticate a remote site before OTV creates an adjacency to that remote site. Each overlay network uses a unique authentication key. An edge device creates an adjacency only with a remote site that shares the same authentication key and authentication method.

OTV supports the following authentication methods:

- Clear text
- Message digest algorithm 5 (MD5)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **otv isis authentication mode** {**md5** | **text**}
5. **otv isis authentication key-chain** *key-chain-name*

6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**otv isis authentication mode** {**md5** | **text**} **Example:** `Device(config-if)# otv isis authentication mode md5`	Configures the authentication method.
Step 5	**otv isis authentication key-chain** *key-chain-name* **Example:** `Device(config-if)# otv isis authentication key-chain OTVkey`	Configures an authentication key chain for edge device authentication. • The key-chain name is case-sensitive. **Note** The key-chain should be already configured on the edge device.
Step 6	Required: **end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Authentication for OTV IS-IS PDUs

Configure OTV to authenticate all incoming OTV IS-IS PDUs.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **otv isis overlay** *overlay-interface*

4. **authentication mode** {**md5** | **text**}
5. **authentication key-chain** *key-chain-name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**otv isis overlay** *overlay-interface* **Example:** `Device(config)# otv isis overlay 1`	Creates an OTV IS-IS instance and enters OTV IS-IS configuration mode. • The range is from 0 to 512.
Step 4	**authentication mode** {**md5**	**text**} **Example:** `Device(config-otv-isis)# authentication mode md5`	Configures the authentication method.
Step 5	**authentication key-chain** *key-chain-name* **Example:** `Device(config-otv-isis)# authentication key-chain OTVkey`	Configures the authentication key chain for PDU authentication. • The key-chain name is case-sensitive. **Note** The key-chain should be already configured on the edge device.
Step 6	Required: **end** **Example:** `Device(config-otv-isis)# end`	Exits OTV IS-IS configuration mode and returns to privileged EXEC mode.

Disabling ARP Caching

An ARP cache is maintained by every OTV edge device and is populated by snooping ARP replies. Initial ARP requests are broadcast to all sites, but subsequent ARP requests are suppressed at the edge device and answered locally. OTV edge devices respond to ARP requests on behalf of remote hosts. Perform this task to allow ARP requests over an overlay network and to disable ARP caching on OTV edge devices.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **no otv suppress arp-nd**
5. **end**
6. **show otv** [**overlay** *overlay-interface*] **arp-nd-cache**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**no otv suppress arp-nd** **Example:** `Device(config-if)# no otv suppress arp-nd`	Allows ARP requests over an overlay network and disables ARP caching on edge devices. • This command does not support IPv6.
Step 5	Required: **end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 6	Required: **show otv** [**overlay** *overlay-interface*] **arp-nd-cache** **Example:** `Device# show otv arp-nd-cache`	(Optional) Displays the OTV Layer 2 to Layer 3 address mapping cache used for ARP suppression.

Tuning OTV Parameters

You can tune parameters for the overlay routing protocol.

Note We recommend that only experienced users of OTV perform these configurations.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **otv isis csnp-interval** *seconds*
5. **otv isis hello-interval** [*seconds* | **minimal**]
6. **otv isis hello-multiplier** *multiplier*
7. **no otv isis hello padding**
8. **otv isis lsp-interval** *milliseconds*
9. **otv isis metric** {*metric* | **maximum**} [*delay-metric* | *expense-metric* | *error-metric*]
10. **otv isis priority** *value*
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode.
Step 4	Required: **otv isis csnp-interval** *seconds* **Example:** `Device(config-if)# otv isis csnp-interval 100`	(Optional) Specifies the interval between complete sequence number PDUs (CSNPs) sent on an interface. Default interval value is 10.
Step 5	Required: **otv isis hello-interval** [*seconds*	**minimal**] **Example:** `Device(config-if)# otv isis hello-interval 30`	(Optional) Specifies the interval between hello PDUs on an interface. Default interval value is 10.

	Command or Action	Purpose
Step 6	Required: **otv isis hello-multiplier** *multiplier* **Example:** `Device(config-if)# otv isis hello-multiplier 30`	(Optional) Specifies the multiplier that is used to calculate the interval within which hello PDUs must be received to keep the OTV adjacency up. Default multiplier value is 3.
Step 7	Required: **no otv isis hello padding** **Example:** `Device(config-if)# no otv isis hello padding`	(Optional) Pads OTV hello PDUs to the full MTU length. It is enabled by default.
Step 8	Required: **otv isis lsp-interval** *milliseconds* **Example:** `Device(config-if)# otv isis lsp-interval 30`	(Optional) Specifies the interval between LSP PDUs on an interface during flooding. Default interval value is 33.
Step 9	Required: **otv isis metric** {*metric* | **maximum**} [*delay-metric* | *expense-metric* | *error-metric*] **Example:** `Device(config-if)# otv isis metric 25`	(Optional) Configures the OTV metric on an interface. Default value is 10.
Step 10	Required: **otv isis priority** *value* **Example:** `Device(config-if)# otv isis priority 6`	(Optional) Configures the OTV priority for the designated router election. Default value is 64.
Step 11	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for OTV Features

Example: Configuring Overlay Interface and VLANs

Virtual Machine (VM1) should be reachable to the edge device 1. In this example, the MAC Address of VM1 is 000b.45b7.82c0.

The following example shows how to configure an edge device 1:

```
Device# 10:52 AM
ip multicast-routing distributed
!
ip igmp snooping querier version 3
ip igmp snooping querier
!
otv site bridge-domain 10
otv site-identifier 0000.0000.0001
!
interface overlay 1
 no shutdown
 otv vpn-name overlay1_site1
 otv control-group 239.1.1.1
```

```
 otv data-group 232.1.1.0/28
 otv join-interface GigabitEthernet 0/0/2
 !
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 20
 !

interface GigabitEthernet 0/0/2
 description "Join Interface"
 ip address 209.165.201.1 255.255.255.224
 ip pim passive
 ip igmp version 3
 !
interface GigabitEthernet 0/0/0
 description "Internal Interface"
 !
 service instance 10 ethernet
  encapsulation dot1q 10
  bridge-domain 10
 !
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 20
 !
ip pim ssm default
```

The following example shows how to configure a Switch 1:

```
SW1#show running-config interface GigabitEthernet 0/0
!
interface GigabitEthernet0/0
 description “Connected to Edge Device-1”
 switchport
 switchport trunk encapsulation dot1q
 switchport mode trunk
 mtu 9216
 no ip address
end

SW1#

SW1#show running-config interface GigabitEthernet 0/1
!
interface GigabitEthernet0/1
 description “Connected to VM1”
 switchport
 switchport access vlan 20
 switchport mode access
 mtu 9216
 no ip address
end

SW1#
```

Virtual Machine (VM2) should be reachable to the edge device 2. In this example, the MAC Address of VM2 is 0013.5f1c.6ec0.

The following example shows how to configure an edge device 2:

```
ip multicast-routing distributed
!
ip igmp snooping querier version 3
ip igmp snooping querier
!
```

```
otv site bridge-domain 11
otv site-identifier 0000.0000.0002
!
interface overlay 1
 no shutdown
 otv vpn-name overlay1_site2
 otv control-group 239.1.1.1
 otv data-group 232.1.1.0/28
 otv join-interface GigabitEthernet 0/0/2
 !
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 20
 !

interface GigabitEthernet 0/0/2
 description "Join Interface"
 ip address 209.165.201.2 255.255.255.224
 ip pim passive
 ip igmp version 3
 !
interface GigabitEthernet 0/0/0
 description "Internal Interface"
 !
 service instance 11 ethernet
  encapsulation dot1q 11
  bridge-domain 11
 !
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 20
 !
ip pim ssm default
```

The following example shows how to configure a Switch 2:

```
SW2#show running-config interface GigabitEthernet 0/0
!
interface GigabitEthernet0/0
 description “Connected to Edge Device-2”
 switchport
 switchport trunk encapsulation dot1q
 switchport mode trunk
 mtu 9216
 no ip address
end

SW2#

SW2#show running-config interface GigabitEthernet 0/1
!
interface GigabitEthernet0/1
 description “Connected to VM2”
 switchport
 switchport access vlan 20
 switchport mode access
 mtu 9216
 no ip address
end

SW2#
```

The following example shows how to configure OTV using multicast.

```
ip multicast-routing distributed

ip igmp snooping querier version 3
ip igmp snooping querier

otv site bridge-domain 11
otv site-identifier 0000.0000.0002

interface GigabitEthernet0/0/0
 description "ACCESS / INTERNAL INTERFACE"
 no shutdown
 negotiation auto
 service instance 11 ethernet
  encapsulation dot1q 11
  bridge-domain 11
 !
 service instance 20 ethernet
  encapsulation dot1q 20
  bridge-domain 20

interface GigabitEthernet0/0/1
 no ip address
 no shutdown
 negotiation auto

router ospf 14
 router-id 14.14.14.1

interface GigabitEthernet0/0/2
 description "JOIN INTERFACE"
 encapsulation dot1Q 11
 ip address 209.165.201.1 255.255.255.224
 ip pim passive
 ip igmp version 3
 ip ospf 14 area 14

interface Overlay1
 no ip address
 no shutdown
 otv control-group 239.1.1.1
 otv data-group 232.1.1.0/28
 otv join-interface GigabitEthernet0/0/2
 service instance 11 ethernet
  encapsulation dot1q 11
  bridge-domain 11

CORE:

ip multicast-routing distributed

router ospf 14
 router-id 14.14.14.2

interface Loopback14
 ip address 14.14.14.14 255.255.255.255
 ip ospf 14 area 14

ip pim rp-address 14.14.14.14

interface GigabitEthernet0/0/0
 no ip address
 no shutdown
 negotiation auto
```

```
interface GigabitEthernet0/0/2
 description "CORE INTERFACE CONNECTED TO ED1"
 encapsulation dot1Q 14
 ip address 209.165.201.1 255.255.255.224
 ip pim sparse-mode
 ip igmp version 3
 ip ospf 14 area 14

interface GigabitEthernet0/0/0
 no ip address
 no shutdown
 negotiation auto

interface GigabitEthernet0/0/1
 description "CORE INTERFACE CONNECTED TO ED2"
 encapsulation dot1Q 11
 ip address 209.165.201.1 255.255.255.224
 ip pim sparse-mode
 ip igmp version 3
 ip ospf 14 area 14

ED2:

ip multicast-routing distributed

ip igmp snooping querier version 3
ip igmp snooping querier

otv site bridge-domain 12
otv site-identifier 0000.0000.0003

interface GigabitEthernet0/0/0
 description "ACCESS / INTERNAL INTERFACE"
 no shutdown
 negotiation auto
 service instance 11 ethernet
  encapsulation dot1q 11
  bridge-domain 11
 !
 service instance 12 ethernet
  encapsulation dot1q 12
  bridge-domain 12

interface GigabitEthernet0/0/3
 no ip address
 no shutdown
 negotiation auto

router ospf 14
 router-id 14.14.14.3

interface GigabitEthernet0/0/4
 description "JOIN INTERFACE"
 encapsulation dot1Q 11
 ip address 209.165.201.1 255.255.255.224
 ip pim passive
 ip igmp version 3
 ip ospf 14 area 14

interface Overlay11
 no ip address
 no shutdown
 otv control-group 239.1.1.1
```

```
 otv data-group 232.1.1.0/28
 otv join-interface GigabitEthernet0/0/4
 service instance 11 ethernet
  encapsulation dot1q 11
  bridge-domain 11
```

The following is sample output from the **show otv** command:

```
Edge-Device-1#show otv overlay1
Overlay Interface Overlay1
 VPN name                 : overlay1_site1
 VPN ID                   : 1
 State                    : UP
 Fwd-capable              : Yes
 Fwd-ready                : Yes
 AED-Server               : Yes
 Backup AED-Server        : No
 AED Capable              : Yes
 IPv4 control group       : 239.1.1.1
 Mcast data group range(s): 232.1.1.0/28
 Join interface(s)        : GigabitEthernet0/0/2
 Join IPv4 address        : 209.165.201.1
 Tunnel interface(s)      : Tunnel0
 Encapsulation format     : GRE/IPv4
 Site Bridge-Domain       : 10
 Capability               : Multicast-reachable
 Is Adjacency Server      : No
 Adj Server Configured    : No
 Prim/Sec Adj Svr(s)      : None

Edge-Device-1#
```

The following sample output from the **show otv adjacency** command shows the OTV overlay adjacency status:

```
Edge-Device-1#show otv overlay 1 adjacency
Overlay Adjacency Database for overlay 1
Hostname                     System-ID      Dest Addr       Site-ID        Up Time   State
Edge-Device-2                 e4aa.5d0f.9b00 209.165.201.2   0000.0000.0002 01:15:13  UP


Edge-Device-1#
```

The following sample output from the **show otv vlan** command shows the OTV VLAN AED status:

```
Edge-Device-1#show otv overlay 1 vlan

Overlay 1 VLAN Configuration Information
 Inst VLAN BD   Auth ED              State               Site If(s)
 0    20 20 *Device                  active              Gi0/0/0:SI20
 Total VLAN(s): 1

Edge-Device-1#
```

The following sample output from the **show otv route** command shows the OTV unicast routing table:

```
Edge-Device-1#show otv overlay 1 route

Codes: BD - Bridge-Domain, AD - Admin-Distance,
       SI - Service Instance, * - Backup Route

OTV Unicast MAC Routing Table for Overlay1

 Inst VLAN BD     MAC Address    AD    Owner  Next Hops(s)
```

```
------------------------------------------------------------
 0    20  20   000b.45b7.82c0    40    BD Eng Gi0/0/0:SI20
 0    20  20   0013.5f1c.6ec0    50    ISIS   Edge-Device-2

2 unicast routes displayed in Overlay1

------------------------------------------------------------
2 Total Unicast Routes Displayed

Edge-Device-1#
```

The following sample output from the **show otv mroute** command shows the OTV multicast routing table:

```
Device# show otv mroute

OTV Multicast Routing Table for Overlay1

Bridge-Domain = 2, s = *, g = *
 Outgoing interface list:
  Default, NoRedist
 Incoming interface count = 0, Outgoing interface count = 1

Bridge-Domain = 3, s = *, g = *
 Outgoing interface list:
  Default, NoRedist
 Incoming interface count = 0, Outgoing interface count = 1

Bridge-Domain = 4, s = *, g = *
 Outgoing interface list:
  Default, NoRedist
 Incoming interface count = 0, Outgoing interface count = 1

Bridge-Domain = 10, s = *, g = 224.0.1.40
 Outgoing interface list:
  Overlay1, ED3
 Incoming interface count = 0, Outgoing interface count = 1

Bridge-Domain = 11, s = *, g = *
 Outgoing interface list:
  Default, NoRedist
 Incoming interface count = 0, Outgoing interface count = 1
5 multicast routes displayed in Overlay1

------------------------------------------------------------
5 Total Multicast Routes Displayed
```

The following sample output from the **show otv data-group** command shows the OTV data group multicast address mappings:

```
Device# show otv data-group

Flags:  D - Local active source dynamically detected
        S - Local active source statically configured
        J - Data group has been joined in the core
        U - Data group has not been joined in the core

Remote Active Sources for Overlay1
 BD    Active-Source   Active-Group    Delivery-Source       Delivery-Group  Flags
 1     10.0.1.1        232.0.0.1       209.165.201.10        232.5.0.0       U
 2     10.0.2.1        232.0.0.1       209.165.201.10        232.5.0.1       U
 3     10.0.3.1        232.0.0.1       209.165.201.10        232.5.0.2       U
 4     10.0.4.1        232.0.0.1       209.165.201.10        232.5.0.3       U
 5     10.0.5.1        232.0.0.1       209.165.201.10        232.5.0.4       J
 6     10.0.6.1        232.0.0.1       209.165.201.10        232.5.0.5       J
```

```
Displayed 6 remote data-group mappings

Local Active Sources for Overlay1
 BD    Active-Source   Active-Group    Delivery-Source       Delivery-Group  Flags
 1     10.0.1.1        232.0.0.1       209.165.201.10        232.5.0.0       D
 2     10.0.2.1        232.0.0.1       209.165.201.10        232.5.0.1       D
 3     10.0.3.1        232.0.0.1       209.165.201.10        232.5.0.2       D
 4     10.0.4.1        232.0.0.1       209.165.201.10        232.5.0.3       D
 5     10.0.5.1        232.0.0.1       209.165.201.10        232.5.0.4       D
 6     10.0.6.1        232.0.0.1       209.165.201.10        232.5.0.5       D
 7     10.0.7.1        232.0.0.1       209.165.201.10        232.5.0.6       D
 8     10.0.8.1        232.0.0.1       209.165.201.10        232.5.0.7       D
 9     10.0.9.1        232.0.0.1       209.165.201.10        232.5.0.8       D
Displayed 9 local data-group mappings
```

The following is sample output for configuring OTV using multicast.

```
ED2#show otv
Overlay Interface Overlay1
 VPN name                 : overlay1_site1
 VPN ID                   : 1
State                     : UP
AED Capable               : Yes
 IPv4 control group       : 239.1.1.1
 Mcast data group range(s): 232.1.1.0/28
 Join interface(s)        : GigabitEthernet0/0/2
 Join IPv4 address        : 209.165.201.1
 Tunnel interface(s)      : Tunnel0
 Encapsulation format     : GRE/IPv4
 Site Bridge-Domain       : 10
Capability                : Multicast-reachable
 Is Adjacency Server      : No
 Adj Server Configured    : No
 Prim/Sec Adj Svr(s)      : None

ED2#

MAC updates related to both VM1 and VM2:

ED2#show otv isis rib mac

Tag Overlay1:
 MAC  local rib for Overlay1 (Total 1)
  L2 Topology ID        Mac Address
  14                    000C.295E.EA91 --> MAC address VM1
    [50/1] via 209.165.201.1(Overlay1), LSP[3/2]
ED2#

The below MAC addresses is sent to the other ED's [these MAC addresses are sent from ED2
to ED1]:

ED2#show otv isis rib redistribution mac

Tag Overlay1:
 MAC redistribution local rib for Overlay1 (Total 3)
  L2 Topology ID        Mac Address
  14                    000C.297E.8CD5
    State: Up/Best/Advertised Metric: 1
  14                    000C.2980.1494 --> MAC address VM2
    State: Up/Best/Advertised Metric: 1
  14                    0050.56BF.4129
    State: Up/Best/Advertised Metric: 1
ED2#
```

```
The below command is the one using which OTV does ARP suppression:

ED2#show otv arp-nd-cache
Overlay150 ARP/ND L3->L2 Address Mapping Cache
BD     MAC            Layer-3 Address  Age (HH:MM:SS) Local/Remote
14     000c.295e.ea91 172.16.11.20     00:01:24       Remote

ED2#

Finally, the packet is routed out using the below table.

ED2#show otv route

Codes: BD - Bridge-Domain, AD - Admin-Distance,
       SI - Service Instance, * - Backup Route

OTV Unicast MAC Routing Table for Overlay150

 Inst VLAN BD     MAC Address    AD    Owner  Next Hops(s)
-------------------------------------------------------------
 0    14   14     000c.295e.ea91 50    ISIS   ED1
 0    14   14     000c.297e.8cd5 40    BD Eng Gi0/0/1:SI14
 0    14   14     000c.2980.1494 40    BD Eng Gi0/0/1:SI14
 0    14   14     0050.56bf.4129 40    BD Eng Gi0/0/1:SI14

4 unicast routes displayed in Overlay1

-------------------------------------------------------------
4 Total Unicast Routes Displayed

ED2#
```

Verifying the OTV Configuration

Use the following commands to display the required OTV configuration information. You can use one or more commands, as required, in any order.

SUMMARY STEPS

1. **show otv** [**overlay** *overlay-interface*]
2. **show otv** [**overlay** *overlay-interface*] **arp-nd-cache**
3. **show otv data-group** [**local** | **remote**] [**detail**]
4. **show otv log** {**event** | **error**}
5. **show otv** [**overlay** *overlay-interface*] **adjacency**
6. **show otv** [**overlay** *overlay-interface*] **vlan** [**authoritative**]
7. **show otv** [**overlay** *overlay-interface*] **site**
8. **show otv route**
9. **show otv mroute**

DETAILED STEPS

Procedure

Step 1 **show otv** [**overlay** *overlay-interface*]

Use this command to display the overlay status and parameters.

Example:

```
Device# show otv
```

Step 2 **show otv** [**overlay** *overlay-interface*] **arp-nd-cache**

Use this command to display the Layer 3 to Layer 2 address mapping cache that is used for ARP suppression.

Example:

```
Device# show otv arp-nd-cache
```

Step 3 **show otv data-group** [**local** | **remote**] [**detail**]

Use this command to display the advertised multicast groups.

Example:

```
Device# show otv data-group
```

Step 4 **show otv log** {**event** | **error**}

Use this command to display the OTV debug log of events or errors.

Example:

```
Device# show otv log event
```

Step 5 **show otv** [**overlay** *overlay-interface*] **adjacency**

Use this command to display information about neighbors in an overlay network.

Example:

```
Device# show otv adjacency
```

Step 6 **show otv** [**overlay** *overlay-interface*] **vlan** [**authoritative**]

Use this command to display information about the enabled OTV VLANs.

Example:

```
Device# show otv vlan
```

Step 7 **show otv** [**overlay** *overlay-interface*] **site**

Use this command to display OTV site information such as the site VLAN and neighbors within the site.

Example:

```
Device# show otv site
```

Step 8 **show otv route**

Use this command to display unicast OTV MAC routes from the MLRIB route database.

Example:

```
Device# show otv route
```

Step 9 **show otv mroute**

Use this command to display OTV multicast routes from the MLRIB route database.

Example:

```
Device# show otv mroute
```

Additional References

Related Documents

Related Topic	Document Title
Wide-area networking commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS Wide-Area Networking Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for OTV

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

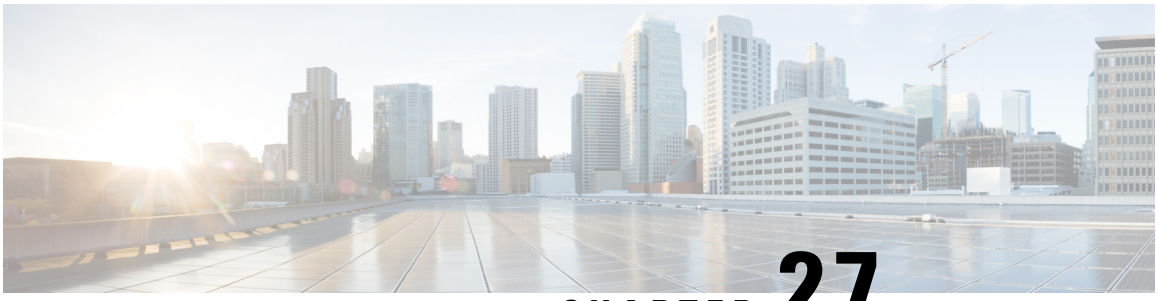
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 47: Feature Information for OTV

Feature Name	Releases	Feature Information
OTV—Overlay Transport Virtualization	Cisco IOS XE Release 3.5S	

Feature Name	Releases	Feature Information
		OTV provides Layer 2 connectivity between remote network sites by using MAC-address-based routing and IP-encapsulated forwarding across a transport network to provide support for applications that require Layer 2 adjacency. The following commands were introduced or modified: **authentication key-chain (OTV)**, **authentication mode (OTV)**, **authentication send-only (OTV)**, **clear otv arp-nd**, **clear otv isis**, **clear otv isis lspfull**, **clear otv isis neighbors**, **clear otv isis rib**, **debug l2fib**, **debug mlrib common**, **debug mlrib layer2**, **debug otv**, **debug otv isis**, **debug platform software l2fib**, **debug platform software otv**, **debug platform hardware qfp feature otv client**, **debug platform hardware qfp feature otv datapath**, **hostname dynamic (OTV)**, **interface overlay**, **log-adjacency-changes (OTV)**, **lsp-gen-intervaL (OTV)**, **lsp-mtu (OTV)**, **lsp-refresh-interval (OTV)**, **max-lsp-lifetime (OTV)**, **nsf (OTV)**, **otv active-source**, **otv control-group**, **otv data-group**, **otv filter-fhrp**, **otv fragmentation**, **otv isis authentication**, **otv isis csnp-interval**, **otv isis hello-interval**, **otv isis hello-multiplier**, **otv isis hello padding**, **otv isis lsp-interval**, **otv isis metric**, **otv isis overlay**, **otv isis priority**, **otv isis retransmit-interval**, **otv isis retransmit-throttle-interval**, **otv isis site otv join-interface**, **otv mac flood**, **otv site bridge-domain**, **otv site-identifier**, **otv suppress arp-nd**, **otv vpn-name**, **prc-interval (OTV)**, **show l2fib**, **show mlrib common log**, **show mlrib layer2 log**, **show otv**, **show otv adjacency**, **show otv**

Feature Name	Releases	Feature Information
		arp-nd-cache, **show otv data-group**, **show otv isis database**, **show otv isis hostname**, **show otv isis lsp-log**, **show otv isis neighbors**, **show otv isis nsf**, **show otv isis protocol**, **show otv isis rib**, **show otv isis spf-log**, **show otv isis vlan-database**, **show otv log**, **show otv mroute**, **show otv route**, **show otv site**, **show otv statistics**, **show otv summary**, **show otv vlan**, **show platform hardware qfp feature otv client interface**, **show platform software l2fib fp**, **show platform software l2fib rp**, **show platform software otv fp**, **skeptical interval (OTV)**, **spf-interval (OTV)**.



CHAPTER 27

OTV Adjacency Server

The OTV Adjacency Server feature enables you to provide unicast-only transport between edge devices when IP multicast is not available in the core network over which Overlay Transport Virtualization (OTV) runs. The OTV control plane over a unicast-only transport works exactly the same way as OTV with multicast core, except that in a unicast-core network, each OTV edge device needs to create multiple copies of each control plane packet and unicast them to each remote edge device in the same logical overlay. Because of this head-end replication behavior, the OTV Adjacency Server feature is not recommended for deployment when IP multicast is available in the core network. At the same time, the operational simplification of the unicast-only model makes this deployment option appealing in scenarios where LAN extension connectivity is required only between a few (two or three) data centers.

To use the OTV Adjacency Server feature, you configure one OTV edge device as a primary adjacency server and you can optionally configure another edge device as a secondary adjacency server as a backup. The remaining edge devices in the overlay network are configured to register to these adjacency servers.

This module describes how to configure an OTV adjacency server and edge devices in a unicast-core network.

- Restrictions for OTV Adjacency Server, on page 491
- Information About OTV Adjacency Server, on page 492
- How to Configure an OTV Adjacency Server, on page 494
- Configuration Examples for OTV Adjacency Server, on page 497
- Additional References for OTV Adjacency Server, on page 498
- Feature Information for OTV Adjacency Server, on page 498

Restrictions for OTV Adjacency Server

- Overlay Transport Virtualization (OTV) does not support a hybrid overlay networks where some edge devices are multicast-capable and others are not.
- OTV adjacency is formed between sites having the same overlay interface number only.
- In a unicast-only core network where an adjacency server has been configured, OTV does not perform the following tasks:
  - Map the configured internal site multicast group to a multicast group in the core.
  - Generate Internet Group Management Protocol (IGMP) and Protocol Independent Multicast (PIM) join interfaces to connect to the core network.

- Add the control group to the Layer 2 Forwarding Information Base (L2FIB) for forwarding default broadcast and link-local packets. Instead, in a unicast-core network, OTV adds active unicast replication list (URL) entries so that broadcast and link-local packets are unicast replicated to remote edge devices.

Information About OTV Adjacency Server

Overview of a Unicast-Core Network

Each Overlay Transport Virtualization (OTV) node provides multicast-send capability by replicating multicast packets at the head-end. In other words, each OTV node that sends a multicast packet on a nonmulticast-capable network, unicast replicates the packet. During unicast replication, a copy of a multicast packet, which is originated in the upper layers of the overlay network, is created and sent to each OTV neighbor that is interested in the multicast packet.

To be able to unicast replicate, each OTV node needs to know a list of neighbors to which to replicate multicast packets. Instead of configuring the list of neighbors on each OTV node, a more dynamic mechanism that supports unicast replication list (URL) is used. In this mechanism, no replication server is used for replicating the multicast packets. Therefore, there are no choke points or path delays because of the lack of multicast capability. The multicast data packets, even though they are sent as a unicast message, travel on the same path from the source OTV edge device to each interested party. The only difference from a multicast-core network is that in this case, multiple copies are sent from the OTV edge device source.

Adjacency Servers

Overview of an Adjacency Server

Overlay Transport Virtualization (OTV) provides support for nonmulticast-capable, unicast-only core networks through the OTV Adjacency Server feature. An edge device is configured as an adjacency server (primary or secondary). All other edge devices are configured with the IPv4 addresses of the primary and secondary adjacency servers, after which the edge devices communicate their reachability and capability information to the primary and secondary adjacency servers.

You can configure more than one adjacency server per overlay network. An adjacency server can serve multiple overlay networks.

Functions of an Adjacency Server

An adjacency server is responsible for informing all the other existing edge devices if there is any addition or loss of an edge device. Based on the reachability information, an edge device can further communicate directly with another edge device by using the unicast data path.

An adjacency server distributes the unicast replication list (URL) of all edge device addresses to the members of the overlay network. Each edge device then uses this list to encapsulate multicast packets in a unicast IP header destined for the unicast IP address of each remote edge device.

Unicast-Only Edge Devices

When an Overlay Transport Virtualization (OTV) edge device is configured with primary and secondary adjacency server addresses, the edge device sends Intermediate System-to-Intermediate System (IS-IS) hello (IIH) messages to both the adjacency servers. When these adjacency servers are configured, the edge devices are immediately added to the unicast replication list (URL). An edge device does not process an alternate server's type, length, value (TLV) until it detects that the primary adjacency server has timed out. Primary and secondary adjacency servers are configured on each edge device.

When a site is added to a unicast-core OTV network, you need to configure only that OTV edge device with the adjacency server addresses. No other site in the overlay network or other overlay networks need additional configuration.

If you configure an edge device as unicast-only, the following restrictions apply:

- The **otv control-group** command should not be configured.
- The **otv data-group** command should not be configured.
- Because no multicast **otv data-group** command is configured, the **show otv data-group** command does not display any data group mappings.

Unicast Replication List

Overlay Transport Virtualization (OTV) maintains the unicast IP address of each remote edge device in the overlay network in a unicast replication list (URL). One URL is maintained per overlay network. OTV marks each address in the URL as active or inactive depending on the local and remote edge devices' unicast-only status. The addresses that are marked as active are added to the forwarding path so that multicast traffic can be unicast replicated to those addresses. Inactive addresses are not added to the forwarding path.

How Adjacency Servers and Edge Devices Work

An adjacency server includes a new Intermediate System-to-Intermediate System (IS-IS) hello (IIH) type, length, value (TLV) in its IIH messages. At first, an adjacency server has no information about other edge devices because its unicast replication list (URL) is empty. After other Overlay Transport Virtualization (OTV) edge devices start sending IIH messages to the adjacency server, the adjacency server builds up its URL. The contents of the URL are the IP addresses of the edge devices that sent IIH messages. The contents of the URL are advertised in the new IIH TLV and sent to each member of the URL. IS-IS itself does not do the replication. The lower layers perform that task so that the overlay network appears as a multiaccess, multicast-capable logical LAN to the upper layers.

OTV edge devices that receive the new IIH TLV discover other OTV edge devices in that overlay network. The IPv4 addresses of other OTV edge devices are added to the local URL. Then, when subsequent IIH messages are sent by IS-IS, they are unicast replicated at the lower layers to each address in the URL. This allows all OTV edge devices in the network to find each other and bring up IS-IS adjacencies with each other. The rest of the IS-IS protocol runs on the overlay interface as if the overlay interface is a multiaccess, multicast-capable LAN.

Exclusivity Between Multicast- and Unicast-Core Networks

The configuration of multicast-core-specific commands and unicast-core-specific adjacency server commands is mutually exclusive. Therefore, if the **otv control-group** command or the **otv data-group** command is

configured, the adjacency server commands are not allowed until the previous commands are un-configured. Similarly, after an adjacency server command is configured, the **otv control-group** and **otv data-group** commands return errors until the adjacency server commands are configured.



Note An edge device can support a combination of unicast and multicast overlays, but an overlay network can be either unicast or multicast.

In multicast- and unicast-core networks, point-to-point or point-to-multipoint generic routing encapsulation (GRE) tunnels are established among the edge devices for forwarding both unicast and multicast traffic.

How to Configure an OTV Adjacency Server

Configuring an OTV Adjacency Server

Perform this task to configure an Overlay Transport Virtualization (OTV) edge device as an adjacency server in a network where the provider core does not support multicast capability.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **otv adjacency-server unicast-only**
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 1`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.

	Command or Action	Purpose
Step 4	**otv adjacency-server unicast-only** **Example:** `Device(config-if)# otv adjacency-server unicast-only`	Configures an OTV edge device as an adjacency server.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Example

The following is sample output from the **show otv** command in a unicast-core network when an OTV edge device is configured as a primary adjacency server:

```
Device# show otv overlay 3

Overlay Interface Overlay3
 VPN name                 : otv_3
 VPN ID                   : 1
 State                    : UP
 AED Capable              : Yes
 Join interface(s)        : GigabitEthernet0/1/1
 Join IPv4 address        : 10.0.2.8
 Tunnel interface(s)      : Tunnel0
 Encapsulation format     : GRE/IPv4
 Site Bridge-Domain       : 2
 Capability               : Unicast-only
 Is Adjacency Server      : Yes
 Adj Server Configured    : No
 Prim/Sec Adj Svr(s)      : None
```

The following is sample output from the **show otv** command in a unicast-core network when another OTV edge device is configured as a secondary adjacency server:

```
Device# show otv overlay 3

Overlay Interface Overlay3
 VPN name                 : otv_3
 VPN ID                   : 1
 State                    : UP
 AED Capable              : Yes
 Join interface(s)        : GigabitEthernet0/3/3
 Join IPv4 address        : 172.16.1.8
 Tunnel interface(s)      : Tunnel0
 Encapsulation format     : GRE/IPv4
 Site Bridge-Domain       : 2
 Capability               : Unicast-only
 Is Adjacency Server      : Yes
 Adj Server Configured    : Yes
 Prim/Sec Adj Svr(s)      : 10.0.2.8
```

Configuring an OTV Edge Device in a Unicast-Core Network

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface overlay** *interface*
4. **otv use-adjacency-server** *primary-address* [*secondary-address*] **unicast-only**
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface overlay** *interface* **Example:** `Device(config)# interface overlay 2`	Creates an OTV overlay interface and enters interface configuration mode. • The range is from 0 to 512.
Step 4	**otv use-adjacency-server** *primary-address* [*secondary-address*] **unicast-only** **Example:** `Device(config-if)# otv use-adjacency-server 10.10.2.2 unicast-only`	Configures an OTV edge device to register to an adjacency server.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Example

The following is sample output from the **show otv adjacency-server replication-list** command:

```
Device# show otv adjacency-server replication-list

OTV Overlay Replication List
 Overlay  Destination Address  Capability
 1        10.15.87.91          Unicast-only
```

```
1        10.99.60.99          Multicast-capable
5        10.66.32.49          Unicast-only
```

The following is sample output from the **show otv** command when an OTV edge device is configured to use primary and secondary adjacency servers:

```
Device# show otv overlay 3

Overlay Interface Overlay3
 VPN name                 : otv_3
 VPN ID                   : 1
 State                    : UP
 AED Capable              : Yes
 Join interface(s)        : GigabitEthernet0/1/1
 Join IPv4 address        : 192.168.1.5
 Tunnel interface(s)      : Tunnel1
 Encapsulation format     : GRE/IPv4
 Site Bridge-Domain       : 2
 Capability               : Unicast-only
 Is Adjacency Server      : No
 Adj Server Configured    : Yes
 Prim/Sec Adj Svr(s)      : 10.0.2.8/172.16.1.8
```

Configuration Examples for OTV Adjacency Server

Example: Configuring an OTV Adjacency Server

The following example shows how to configure an edge device as an adjacency server:

```
Device> enable
Device# configure terminal
Device(config)# interface overlay 1
Device(config-if)# otv adjacency-server unicast-only
Device(config-if)# end
```

Example: Configuring an OTV Edge Device in a Unicast-Core Network

The following example shows how to configure an edge device to register to an adjacency server in a unicast-core network:

```
Device> enable
Device# configure terminal
Device(config)# interface overlay 1
Device(config-if)# otv use-adjacency-server 10.10.1.2 unicast-only
Device(config-if)# end
```

Additional References for OTV Adjacency Server

Related Documents

Related Topic	Document Title
Wide-area networking commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS Wide-Area Networking Command Reference
Unicast-Only Transport Infrastructure	OTV Technology Introduction and Deployment Considerations
Configuring OTV Adjacency Servers	*Cisco Nexus 7000 Series NX-OS OTV Configuration Guide*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for OTV Adjacency Server

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 48: Feature Information for OTV Adjacency Server

Feature Name	Releases	Feature Information
OTV Adjacency Server	Cisco IOS XE Release 3.9S	The OTV Adjacency Server feature enables you to provide unicast-only transport between edge devices when IP multicast is not available in the core network over which Overlay Transport Virtualization (OTV) runs. To use the OTV Adjacency Server feature, you configure one OTV edge device as a primary adjacency server and you can optionally configure another edge device as a secondary adjacency server as a backup. The remaining edge devices in the overlay network are configured to register to these adjacency servers. In Cisco IOS XE Release 3.9S, support was added for the Cisco ASR 1000 Series Routers and Cisco CSR 1000V Series Routers. The following commands were introduced or modified: **otv adjacency-server unicast-only**, **otv use-adjacency-server unicast-only**, **show otv adjacency-server replication-list**.