Cisco IOS Wide-Area Networking Configuration Guide

Release 12.2SR

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

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.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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.

Cisco IOS Wide-Area Networking Configuration Guide
© 2009 Cisco Systems, Inc. All rights reserved.

About Cisco IOS Software Documentation

Last Updated: November 20, 2009

This document describes the objectives, audience, conventions, and organization used in Cisco IOS software documentation. Also included are resources for obtaining technical assistance, additional documentation, and other information from Cisco. This document is organized into the following sections:

- Documentation Objectives, page i
- Audience, page i
- Documentation Conventions, page i
- Documentation Organization, page iii
- Additional Resources and Documentation Feedback, page xi

Documentation Objectives

Cisco IOS documentation describes the tasks and commands available to configure and maintain Cisco networking devices.

Audience

The Cisco IOS documentation set is intended for users who configure and maintain Cisco networking devices (such as routers and switches) but who may not be familiar with the configuration and maintenance tasks, the relationship among tasks, or the Cisco IOS commands necessary to perform particular tasks. The Cisco IOS documentation set is also intended for those users experienced with Cisco IOS software who need to know about new features, new configuration options, and new software characteristics in the current Cisco IOS release.

Documentation Conventions

In Cisco IOS documentation, the term *router* may be used to refer to various Cisco products; for example, routers, access servers, and switches. These and other networking devices that support Cisco IOS software are shown interchangeably in examples and are used only for illustrative purposes. An example that shows one product does not necessarily mean that other products are not supported.

This section contains the following topics:

- Typographic Conventions, page ii
- Command Syntax Conventions, page ii
- Software Conventions, page iii
- Reader Alert Conventions, page iii

Typographic Conventions

Cisco IOS documentation uses the following typographic conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you 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 a Simple Network Management Protocol (SNMP) community string to *public*, do not use quotation marks around the string; otherwise, the string will include the quotation marks.

Command Syntax Conventions

Cisco IOS documentation uses the following command syntax conventions:

Convention	Description
bold	Bold text indicates commands and keywords that you enter as shown.
italic	Italic text indicates arguments for which you supply values.
[x]	Square brackets enclose an optional keyword or argument.
...	An ellipsis (three consecutive nonbolded periods without spaces) after a syntax element indicates that the element can be repeated.
|	A vertical line, called a pipe, that is enclosed within braces or square brackets indicates a choice within a set of keywords or arguments.
[x | y]	Square brackets enclosing keywords or arguments separated by a pipe indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a pipe indicate a required choice.
[x {y | z}]	Braces and a pipe within square brackets indicate a required choice within an optional element.

Software Conventions

Cisco IOS software uses the following program code conventions:

Convention	Description
`Courier font`	Courier font is used for information that is displayed on a PC or terminal screen.
`Bold Courier font`	Bold Courier font indicates text that the user must enter.
< >	Angle brackets enclose text that is not displayed, such as a password. Angle brackets also are used in contexts in which the italic font style is not supported; for example, ASCII text.
!	An exclamation point at the beginning of a line indicates that the text that follows is a comment, not a line of code. An exclamation point is also displayed by Cisco IOS software for certain processes.
[ ]	Square brackets enclose default responses to system prompts.

Reader Alert Conventions

Cisco IOS documentation uses the following conventions for reader alerts:

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 material not covered in the manual.

Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Documentation Organization

This section describes the Cisco IOS documentation set, how it is organized, and how to access it on Cisco.com. It also lists the configuration guides, command references, and supplementary references and resources that comprise the documentation set. It contains the following topics:

- Cisco IOS Documentation Set, page iv
- Cisco IOS Documentation on Cisco.com, page iv
- Configuration Guides, Command References, and Supplementary Resources, page v

Cisco IOS Documentation Set

The Cisco IOS documentation set consists of the following:

- Release notes and caveats provide information about platform, technology, and feature support for a release and describe severity 1 (catastrophic), severity 2 (severe), and select severity 3 (moderate) defects in released Cisco IOS software. Review release notes before other documents to learn whether updates have been made to a feature.
- Sets of configuration guides and command references organized by technology and published for each standard Cisco IOS release.
  - Configuration guides—Compilations of documents that provide conceptual and task-oriented descriptions of Cisco IOS features.
  - Command references—Compilations of command pages in alphabetical order that provide detailed information about the commands used in the Cisco IOS features and the processes that comprise the related configuration guides. For each technology, there is a single command reference that supports all Cisco IOS releases and that is updated at each standard release.
- Lists of all the commands in a specific release and all commands that are new, modified, removed, or replaced in the release.
- Command reference book for **debug** commands. Command pages are listed in alphabetical order.
- Reference book for system messages for all Cisco IOS releases.

Cisco IOS Documentation on Cisco.com

The following sections describe the organization of the Cisco IOS documentation set and how to access various document types.

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

New Features List

The New Features List for each release provides a list of all features in the release with hyperlinks to the feature guides in which they are documented.

Feature Guides

Cisco IOS features are documented in feature guides. Feature guides describe one feature or a group of related features that are supported on many different software releases and platforms. Your Cisco IOS software release or platform may not support all the features documented in a feature guide. See the Feature Information table at the end of the feature guide for information about which features in that guide are supported in your software release.

Configuration Guides

Configuration guides are provided by technology and release and comprise a set of individual feature guides relevant to the release and technology.

Command References

Command reference books contain descriptions of Cisco IOS commands that are supported in many different software releases and on many different platforms. The books are organized by technology. For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases*, at http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html.

Cisco IOS Supplementary Documents and Resources

Supplementary documents and resources are listed in Table 2 on page xi.

Configuration Guides, Command References, and Supplementary Resources

Table 1 lists, in alphabetical order, Cisco IOS software configuration guides and command references, including brief descriptions of the contents of the documents. The Cisco IOS command references contain commands for Cisco IOS software for all releases. The configuration guides and command references support many different software releases and platforms. Your Cisco IOS software release or platform may not support all these technologies.

Table 2 lists documents and resources that supplement the Cisco IOS software configuration guides and command references. These supplementary resources include release notes and caveats; master command lists; new, modified, removed, and replaced command lists; system messages; and the debug command reference.

For additional information about configuring and operating specific networking devices, and to access Cisco IOS documentation, go to the Product/Technologies Support area of Cisco.com at the following location:

http://www.cisco.com/go/techdocs

Table 1 Cisco IOS Configuration Guides and Command References

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS AppleTalk Configuration Guide* • *Cisco IOS AppleTalk Command Reference*	AppleTalk protocol.
• *Cisco IOS Asynchronous Transfer Mode Configuration Guide* • *Cisco IOS Asynchronous Transfer Mode Command Reference*	LAN ATM, multiprotocol over ATM (MPoA), and WAN ATM.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Bridging and IBM Networking Configuration Guide* • *Cisco IOS Bridging Command Reference* • *Cisco IOS IBM Networking Command Reference*	Transparent and source-route transparent (SRT) bridging, source-route bridging (SRB), Token Ring Inter-Switch Link (TRISL), and token ring route switch module (TRRSM). Data-link switching plus (DLSw+), serial tunnel (STUN), block serial tunnel (BSTUN); logical link control, type 2 (LLC2), synchronous data link control (SDLC); IBM Network Media Translation, including Synchronous Data Logical Link Control (SDLLC) and qualified LLC (QLLC); downstream physical unit (DSPU), Systems Network Architecture (SNA) service point, SNA frame relay access, advanced peer-to-peer networking (APPN), native client interface architecture (NCIA) client/server topologies, and IBM Channel Attach.
• *Cisco IOS Broadband Access Aggregation and DSL Configuration Guide* • *Cisco IOS Broadband Access Aggregation and DSL Command Reference*	PPP over ATM (PPPoA) and PPP over Ethernet (PPPoE).
• *Cisco IOS Carrier Ethernet Configuration Guide* • *Cisco IOS Carrier Ethernet Command Reference*	Connectivity fault management (CFM), Ethernet Local Management Interface (ELMI), IEEE 802.3ad link bundling, Link Layer Discovery Protocol (LLDP), media endpoint discovery (MED), and Operation, Administration, and Maintenance (OAM).
• *Cisco IOS Configuration Fundamentals Configuration Guide* • *Cisco IOS Configuration Fundamentals Command Reference*	Autoinstall, Setup, Cisco IOS command-line interface (CLI), Cisco IOS file system (IFS), Cisco IOS web browser user interface (UI), basic file transfer services, and file management.
• *Cisco IOS DECnet Configuration Guide* • *Cisco IOS DECnet Command Reference*	DECnet protocol.
• *Cisco IOS Dial Technologies Configuration Guide* • *Cisco IOS Dial Technologies Command Reference*	Asynchronous communications, dial backup, dialer technology, dial-in terminal services and AppleTalk remote access (ARA), dial-on-demand routing, dial-out, ISDN, large scale dial-out, modem and resource pooling, Multilink PPP (MLP), PPP, and virtual private dialup network (VPDN).
• *Cisco IOS Flexible NetFlow Configuration Guide* • *Cisco IOS Flexible NetFlow Command Reference*	Flexible NetFlow.
• *Cisco IOS High Availability Configuration Guide* • *Cisco IOS High Availability Command Reference*	A variety of high availability (HA) features and technologies that are available for different network segments (from enterprise access to service provider core) to facilitate creation of end-to-end highly available networks. Cisco IOS HA features and technologies can be categorized in three key areas: system-level resiliency, network-level resiliency, and embedded management for resiliency.
• *Cisco IOS Integrated Session Border Controller Command Reference*	A VoIP-enabled device that is deployed at the edge of networks. An SBC is a toolkit of functions, such as signaling interworking, network hiding, security, and quality of service (QoS).

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Intelligent Services Gateway Configuration Guide* • *Cisco IOS Intelligent Services Gateway Command Reference*	Subscriber identification, service and policy determination, session creation, session policy enforcement, session life-cycle management, accounting for access and service usage, and session state monitoring.
• *Cisco IOS Interface and Hardware Component Configuration Guide* • *Cisco IOS Interface and Hardware Component Command Reference*	LAN interfaces, logical interfaces, serial interfaces, virtual interfaces, and interface configuration.
• *Cisco IOS IP Addressing Services Configuration Guide* • *Cisco IOS IP Addressing Services Command Reference*	Address Resolution Protocol (ARP), Network Address Translation (NAT), Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and Next Hop Address Resolution Protocol (NHRP).
• *Cisco IOS IP Application Services Configuration Guide* • *Cisco IOS IP Application Services Command Reference*	Enhanced Object Tracking (EOT), Gateway Load Balancing Protocol (GLBP), Hot Standby Router Protocol (HSRP), IP Services, Server Load Balancing (SLB), Stream Control Transmission Protocol (SCTP), TCP, Web Cache Communication Protocol (WCCP), User Datagram Protocol (UDP), and Virtual Router Redundancy Protocol (VRRP).
• *Cisco IOS IP Mobility Configuration Guide* • *Cisco IOS IP Mobility Command Reference*	Mobile ad hoc networks (MANet) and Cisco mobile networks.
• *Cisco IOS IP Multicast Configuration Guide* • *Cisco IOS IP Multicast Command Reference*	Protocol Independent Multicast (PIM) sparse mode (PIM-SM), bidirectional PIM (bidir-PIM), Source Specific Multicast (SSM), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol (IGMP), and Multicast VPN (MVPN).
• *Cisco IOS IP Routing: BFD Configuration Guide*	Bidirectional forwarding detection (BFD).
• *Cisco IOS IP Routing: BGP Configuration Guide* • *Cisco IOS IP Routing: BGP Command Reference*	Border Gateway Protocol (BGP), multiprotocol BGP, multiprotocol BGP extensions for IP multicast.
• *Cisco IOS IP Routing: EIGRP Configuration Guide* • *Cisco IOS IP Routing: EIGRP Command Reference*	Enhanced Interior Gateway Routing Protocol (EIGRP).
• *Cisco IOS IP Routing: ISIS Configuration Guide* • *Cisco IOS IP Routing: ISIS Command Reference*	Intermediate System-to-Intermediate System (IS-IS).
• *Cisco IOS IP Routing: ODR Configuration Guide* • *Cisco IOS IP Routing: ODR Command Reference*	On-Demand Routing (ODR).
• *Cisco IOS IP Routing: OSPF Configuration Guide* • *Cisco IOS IP Routing: OSPF Command Reference*	Open Shortest Path First (OSPF).
• *Cisco IOS IP Routing: Protocol-Independent Configuration Guide* • *Cisco IOS IP Routing: Protocol-Independent Command Reference*	IP routing protocol-independent features and commands. Generic policy-based routing (PBR) features and commands are included.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS IP Routing: RIP Configuration Guide* • *Cisco IOS IP Routing: RIP Command Reference*	Routing Information Protocol (RIP).
• *Cisco IOS IP SLAs Configuration Guide* • *Cisco IOS IP SLAs Command Reference*	Cisco IOS IP Service Level Agreements (IP SLAs).
• *Cisco IOS IP Switching Configuration Guide* • *Cisco IOS IP Switching Command Reference*	Cisco Express Forwarding, fast switching, and Multicast Distributed Switching (MDS).
• *Cisco IOS IPv6 Configuration Guide* • *Cisco IOS IPv6 Command Reference*	For IPv6 features, protocols, and technologies, go to the IPv6 "Start Here" document.
• *Cisco IOS ISO CLNS Configuration Guide* • *Cisco IOS ISO CLNS Command Reference*	ISO Connectionless Network Service (CLNS).
• *Cisco IOS LAN Switching Configuration Guide* • *Cisco IOS LAN Switching Command Reference*	VLANs, Inter-Switch Link (ISL) encapsulation, IEEE 802.10 encapsulation, IEEE 802.1Q encapsulation, and multilayer switching (MLS).
• *Cisco IOS Mobile Wireless Gateway GPRS Support Node Configuration Guide* • *Cisco IOS Mobile Wireless Gateway GPRS Support Node Command Reference*	Cisco IOS Gateway GPRS Support Node (GGSN) in a 2.5-generation general packet radio service (GPRS) and 3-generation universal mobile telecommunication system (UMTS) network.
• *Cisco IOS Mobile Wireless Home Agent Configuration Guide* • *Cisco IOS Mobile Wireless Home Agent Command Reference*	Cisco Mobile Wireless Home Agent, an anchor point for mobile terminals for which mobile IP or proxy mobile IP services are provided.
• *Cisco IOS Mobile Wireless Packet Data Serving Node Configuration Guide* • *Cisco IOS Mobile Wireless Packet Data Serving Node Command Reference*	Cisco Packet Data Serving Node (PDSN), a wireless gateway that is between the mobile infrastructure and standard IP networks and that enables packet data services in a code division multiple access (CDMA) environment.
• *Cisco IOS Mobile Wireless Radio Access Networking Configuration Guide* • *Cisco IOS Mobile Wireless Radio Access Networking Command Reference*	Cisco IOS radio access network products.
• *Cisco IOS Multiprotocol Label Switching Configuration Guide* • *Cisco IOS Multiprotocol Label Switching Command Reference*	MPLS Label Distribution Protocol (LDP), MPLS Layer 2 VPNs, MPLS Layer 3 VPNs, MPLS traffic engineering (TE), and MPLS Embedded Management (EM) and MIBs.
• *Cisco IOS Multi-Topology Routing Configuration Guide* • *Cisco IOS Multi-Topology Routing Command Reference*	Unicast and multicast topology configurations, traffic classification, routing protocol support, and network management support.
• *Cisco IOS NetFlow Configuration Guide* • *Cisco IOS NetFlow Command Reference*	Network traffic data analysis, aggregation caches, and export features.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Network Management Configuration Guide* • *Cisco IOS Network Management Command Reference*	Basic system management; system monitoring and logging; troubleshooting, logging, and fault management; Cisco Discovery Protocol; Cisco IOS Scripting with Tool Control Language (Tcl); Cisco networking services (CNS); DistributedDirector; Embedded Event Manager (EEM); Embedded Resource Manager (ERM); Embedded Syslog Manager (ESM); HTTP; Remote Monitoring (RMON); SNMP; and VPN Device Manager Client for Cisco IOS software (XSM Configuration).
• *Cisco IOS Novell IPX Configuration Guide* • *Cisco IOS Novell IPX Command Reference*	Novell Internetwork Packet Exchange (IPX) protocol.
• *Cisco IOS Optimized Edge Routing Configuration Guide* • *Cisco IOS Optimized Edge Routing Command Reference*	Optimized edge routing (OER) monitoring; Performance Routing (PfR); and automatic route optimization and load distribution for multiple connections between networks.
• *Cisco IOS Quality of Service Solutions Configuration Guide* • *Cisco IOS Quality of Service Solutions Command Reference*	Traffic queueing, traffic policing, traffic shaping, Modular QoS CLI (MQC), Network-Based Application Recognition (NBAR), Multilink PPP (MLP) for QoS, header compression, AutoQoS, Resource Reservation Protocol (RSVP), and weighted random early detection (WRED).
• *Cisco IOS Security Command Reference*	Access control lists (ACLs); authentication, authorization, and accounting (AAA); firewalls; IP security and encryption; neighbor router authentication; network access security; network data encryption with router authentication; public key infrastructure (PKI); RADIUS; TACACS+; terminal access security; and traffic filters.
• *Cisco IOS Security Configuration Guide: Securing the Data Plane*	Access Control Lists (ACLs); Firewalls: Context-Based Access Control (CBAC) and Zone-Based Firewall; Cisco IOS Intrusion Prevention System (IPS); Flexible Packet Matching; Unicast Reverse Path Forwarding (uRPF); Threat Information Distribution Protocol (TIDP) and TMS.
• *Cisco IOS Security Configuration Guide: Securing the Control Plane*	Control Plane Policing, Neighborhood Router Authentication.
• *Cisco IOS Security Configuration Guide: Securing User Services*	AAA (includes 802.1x authentication and Network Admission Control [NAC]); Security Server Protocols (RADIUS and TACACS+); Secure Shell (SSH); Secure Access for Networking Devices (includes Autosecure and Role-Based CLI access); Lawful Intercept.
• *Cisco IOS Security Configuration Guide: Secure Connectivity*	Internet Key Exchange (IKE) for IPsec VPNs; IPsec Data Plane features; IPsec Management features; Public Key Infrastructure (PKI); Dynamic Multipoint VPN (DMVPN); Easy VPN; Cisco Group Encrypted Transport VPN (GETVPN); SSL VPN.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Service Advertisement Framework Configuration Guide* • *Cisco IOS Service Advertisement Framework Command Reference*	Cisco Service Advertisement Framework.
• *Cisco IOS Service Selection Gateway Configuration Guide* • *Cisco IOS Service Selection Gateway Command Reference*	Subscriber authentication, service access, and accounting.
• *Cisco IOS Software Activation Configuration Guide* • *Cisco IOS Software Activation Command Reference*	An orchestrated collection of processes and components to activate Cisco IOS software feature sets by obtaining and validating Cisco software licenses.
• *Cisco IOS Software Modularity Installation and Configuration Guide* • *Cisco IOS Software Modularity Command Reference*	Installation and basic configuration of software modularity images, including installations on single and dual route processors, installation rollbacks, software modularity binding, software modularity processes, and patches.
• *Cisco IOS Terminal Services Configuration Guide* • *Cisco IOS Terminal Services Command Reference*	DEC, local-area transport (LAT), and X.25 packet assembler/disassembler (PAD).
• *Cisco IOS Virtual Switch Command Reference*	Virtual switch redundancy, high availability, and packet handling; converting between standalone and virtual switch modes; virtual switch link (VSL); Virtual Switch Link Protocol (VSLP). **Note** For information about virtual switch configuration, see the product-specific software configuration information for the Cisco Catalyst 6500 series switch or for the Metro Ethernet 6500 series switch.
• *Cisco IOS Voice Configuration Library* • *Cisco IOS Voice Command Reference*	Cisco IOS support for voice call control protocols, interoperability, physical and virtual interface management, and troubleshooting. The library includes documentation for IP telephony applications.
• *Cisco IOS VPDN Configuration Guide* • *Cisco IOS VPDN Command Reference*	Layer 2 Tunneling Protocol (L2TP) dial-out load balancing and redundancy; L2TP extended failover; L2TP security VPDN; multihop by Dialed Number Identification Service (DNIS); timer and retry enhancements for L2TP and Layer 2 Forwarding (L2F); RADIUS Attribute 82 (tunnel assignment ID); shell-based authentication of VPDN users; tunnel authentication via RADIUS on tunnel terminator.
• *Cisco IOS Wide-Area Networking Configuration Guide* • *Cisco IOS Wide-Area Networking Command Reference*	Frame Relay; Layer 2 Tunnel Protocol Version 3 (L2TPv3); L2VPN Pseudowire Redundancy; L2VPN Interworking; Layer 2 Local Switching; Link Access Procedure, Balanced (LAPB); and X.25.
• *Cisco IOS Wireless LAN Configuration Guide* • *Cisco IOS Wireless LAN Command Reference*	Broadcast key rotation, IEEE 802.11x support, IEEE 802.1x authenticator, IEEE 802.1x local authentication service for Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling (EAP-FAST), Multiple Basic Service Set ID (BSSID), Wi-Fi Multimedia (WMM) required elements, and Wi-Fi Protected Access (WPA).

Table 2 lists documents and resources that supplement the Cisco IOS software configuration guides and command references.

Table 2 Cisco IOS Supplementary Documents and Resources

Document Title or Resource	Description
Cisco IOS Master Command List, All Releases	Alphabetical list of all the commands documented in all Cisco IOS releases.
Cisco IOS New, Modified, Removed, and Replaced Commands	List of all the new, modified, removed, and replaced commands for a Cisco IOS release.
Cisco IOS System Message Guide	List of Cisco IOS system messages and descriptions. System messages may indicate problems with your system, may be informational only, or may help diagnose problems with communications lines, internal hardware, or system software.
Cisco IOS Debug Command Reference	Alphabetical list of **debug** commands including brief descriptions of use, command syntax, and usage guidelines.
Release Notes and Caveats	Information about new and changed features, system requirements, and other useful information about specific software releases; information about defects in specific Cisco IOS software releases.
MIBs	Files used for network monitoring. To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator.
RFCs	Standards documents maintained by the Internet Engineering Task Force (IETF) that Cisco IOS documentation references where applicable. The full text of referenced RFCs may be obtained at the following URL: http://www.rfc-editor.org/

Additional Resources and Documentation Feedback

What's New in Cisco Product Documentation is released monthly and describes all new and revised Cisco technical documentation. The *What's New in Cisco Product Documentation* publication also provides information about obtaining the following resources:

- Technical documentation
- Cisco product security overview
- Product alerts and field notices
- Technical assistance

Cisco IOS technical documentation includes embedded feedback forms where you can rate documents and provide suggestions for improvement. Your feedback helps us improve our documentation.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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.

© 2008–2009 Cisco Systems, Inc. All rights reserved.

Using the Command-Line Interface in Cisco IOS Software

Last Updated: October 14, 2009

This document provides basic information about the command-line interface (CLI) in Cisco IOS software and how you can use some of the CLI features. This document contains the following sections:

- Initially Configuring a Device, page i
- Using the CLI, page ii
- Saving Changes to a Configuration, page xi
- Additional Information, page xii

For more information about using the CLI, see the "Using the Cisco IOS Command-Line Interface" section of the *Cisco IOS Configuration Fundamentals Configuration Guide*.

For information about the software documentation set, see the "About Cisco IOS Software Documentation" document.

Initially Configuring a Device

Initially configuring a device varies by platform. For information about performing an initial configuration, see the hardware installation documentation that is provided with the original packaging of the product or go to the Product/Technologies Support area of Cisco.com at http://www.cisco.com/go/techdocs.

After you have performed the initial configuration and connected the device to your network, you can configure the device by using the console port or a remote access method, such as Telnet or Secure Shell (SSH), to access the CLI or by using the configuration method provided on the device, such as Security Device Manager.

Changing the Default Settings for a Console or AUX Port

There are only two changes that you can make to a console port and an AUX port:

- Change the port speed with the **config-register 0x** command. Changing the port speed is not recommended. The well-known default speed is 9600.
- Change the behavior of the port; for example, by adding a password or changing the timeout value.

Note The AUX port on the Route Processor (RP) installed in a Cisco ASR 1000 series router does not serve any useful customer purpose and should be accessed only under the advisement of a customer support representative.

Using the CLI

This section describes the following topics:

- Understanding Command Modes, page ii
- Using the Interactive Help Feature, page v
- Understanding Command Syntax, page vi
- Understanding Enable and Enable Secret Passwords, page vii
- Using the Command History Feature, page viii
- Abbreviating Commands, page ix
- Using Aliases for CLI Commands, page ix
- Using the no and default Forms of Commands, page x
- Using the debug Command, page x
- Filtering Output Using Output Modifiers, page x
- Understanding CLI Error Messages, page xi

Understanding Command Modes

The CLI command mode structure is hierarchical, and each mode supports a set of specific commands. This section describes the most common of the many modes that exist.

Table 1 lists common command modes with associated CLI prompts, access and exit methods, and a brief description of how each mode is used.

Table 1 CLI Command Modes

Command Mode	Access Method	Prompt	Exit Method	Mode Usage
User EXEC	Log in.	`Router>`	Issue the **logout** or **exit** command.	• Change terminal settings. • Perform basic tests. • Display device status.
Privileged EXEC	From user EXEC mode, issue the **enable** command.	`Router#`	Issue the **disable** command or the **exit** command to return to user EXEC mode.	• Issue **show** and **debug** commands. • Copy images to the device. • Reload the device. • Manage device configuration files. • Manage device file systems.
Global configuration	From privileged EXEC mode, issue the **configure terminal** command.	`Router(config)#`	Issue the **exit** command or the **end** command to return to privileged EXEC mode.	Configure the device.
Interface configuration	From global configuration mode, issue the **interface** command.	`Router(config-if)#`	Issue the **exit** command to return to global configuration mode or the **end** command to return to privileged EXEC mode.	Configure individual interfaces.
Line configuration	From global configuration mode, issue the **line vty** or **line console** command.	`Router(config-line)#`	Issue the **exit** command to return to global configuration mode or the **end** command to return to privileged EXEC mode.	Configure individual terminal lines.

Table 1 CLI Command Modes (continued)

Command Mode	Access Method	Prompt	Exit Method	Mode Usage
ROM monitor	From privileged EXEC mode, issue the **reload** command. Press the **Break** key during the first 60 seconds while the system is booting.	`rommon # >` The # symbol represents the line number and increments at each prompt.	Issue the **continue** command.	• Run as the default operating mode when a valid image cannot be loaded. • Access the fall-back procedure for loading an image when the device lacks a valid image and cannot be booted. • Perform password recovery when a Ctrl-Break sequence is issued within 60 seconds of a power-on or reload event.
Diagnostic (available only on Cisco ASR 1000 series routers)	The router boots or enters diagnostic mode in the following scenarios. When a Cisco IOS process or processes fail, in most scenarios the router will reload. • A user-configured access policy was configured using the **transport-map** command, which directed the user into diagnostic mode. • The router was accessed using an RP auxiliary port. • A break signal (**Ctrl-C**, **Ctrl-Shift-6**, or the **send break** command) was entered, and the router was configured to enter diagnostic mode when the break signal was received.	`Router(diag)#`	If a Cisco IOS process failure is the reason for entering diagnostic mode, the failure must be resolved and the router must be rebooted to exit diagnostic mode. If the router is in diagnostic mode because of a transport-map configuration, access the router through another port or use a method that is configured to connect to the Cisco IOS CLI. If the RP auxiliary port was used to access the router, use another port for access. Accessing the router through the auxiliary port is not useful for customer purposes.	• Inspect various states on the router, including the Cisco IOS state. • Replace or roll back the configuration. • Provide methods of restarting the Cisco IOS software or other processes. • Reboot hardware (such as the entire router, an RP, an ESP, a SIP, a SPA) or other hardware components. • Transfer files into or off of the router using remote access methods such as FTP, TFTP, and SCP.

EXEC commands are not saved when the software reboots. Commands that you issue in a configuration mode can be saved to the startup configuration. If you save the running configuration to the startup configuration, these commands will execute when the software is rebooted. Global configuration mode is the highest level of configuration mode. From global configuration mode, you can enter a variety of other configuration modes, including protocol-specific modes.

ROM monitor mode is a separate mode that is used when the software cannot load properly. If a valid software image is not found when the software boots or if the configuration file is corrupted at startup, the software might enter ROM monitor mode. Use the question symbol (?) to view the commands that you can use while the device is in ROM monitor mode.

```
rommon 1 > ?
alias               set and display aliases command
boot                boot up an external process
confreg             configuration register utility
cont                continue executing a downloaded image
context             display the context of a loaded image
cookie              display contents of cookie PROM in hex
.
.
.
rommon 2 >
```

The following example shows how the command prompt changes to indicate a different command mode:

```
Router> enable
Router# configure terminal
Router(config)# interface ethernet 1/1
Router(config-if)# ethernet
Router(config-line)# exit
Router(config)# end
Router#
```

Note A keyboard alternative to the **end** command is Ctrl-Z.

Using the Interactive Help Feature

The CLI includes an interactive Help feature. Table 2 describes the purpose of the CLI interactive Help commands.

Table 2 CLI Interactive Help Commands

Command	Purpose
help	Provides a brief description of the Help feature in any command mode.
?	Lists all commands available for a particular command mode.
*partial command***?**	Provides a list of commands that begin with the character string (no space between the command and the question mark).
partial command<**Tab**>	Completes a partial command name (no space between the command and <Tab>).
command **?**	Lists the keywords, arguments, or both associated with the command (space between the command and the question mark).
command keyword **?**	Lists the arguments that are associated with the keyword (space between the keyword and the question mark).

The following examples show how to use the help commands:

help

```
Router> help

Help may be requested at any point in a command by entering a question mark '?'. If nothing
matches, the help list will be empty and you must backup until entering a '?' shows the
available options.

Two styles of help are provided:

1. Full help is available when you are ready to enter a command argument (e.g. 'show ?')
and describes each possible argument.

2. Partial help is provided when an abbreviated argument is entered and you want to know
what arguments match the input (e.g. 'show pr?'.)
```

?

```
Router# ?
Exec commands:
  access-enable        Create a temporary access-List entry
  access-profile       Apply user-profile to interface
  access-template      Create a temporary access-List entry
  alps                 ALPS exec commands
  archive              manage archive files
<snip>
```

partial command?

```
Router(config)# zo?
zone  zone-pair
```

partial command<Tab>

```
Router(config)# we<Tab> webvpn
```

command ?

```
Router(config-if)# pppoe ?
  enable        Enable pppoe
  max-sessions  Maximum PPPOE sessions
```

command keyword ?

```
Router(config-if)# pppoe enable ?
  group  attach a BBA group
  <cr>
```

Understanding Command Syntax

Command syntax is the format in which a command should be entered in the CLI. Commands include the name of the command, keywords, and arguments. Keywords are alphanumeric strings that are used literally. Arguments are placeholders for values that a user must supply. Keywords and arguments may be required or optional.

Specific conventions convey information about syntax and command elements. Table 3 describes these conventions.

Table 3 CLI Syntax Conventions

Symbol/Text	Function	Notes
< > (angle brackets)	Indicate that the option is an argument.	Sometimes arguments are displayed without angle brackets.
A.B.C.D.	Indicates that you must enter a dotted decimal IP address.	Angle brackets (< >) are not always used to indicate that an IP address is an argument.
WORD (all capital letters)	Indicates that you must enter one word.	Angle brackets (< >) are not always used to indicate that a WORD is an argument.
LINE (all capital letters)	Indicates that you must enter more than one word.	Angle brackets (< >) are not always used to indicate that a LINE is an argument.
<cr> (carriage return)	Indicates the end of the list of available keywords and arguments, and also indicates when keywords and arguments are optional. When <cr> is the only option, you have reached the end of the branch or the end of the command if the command has only one branch.	—

The following examples show syntax conventions:

```
Router(config)# ethernet cfm domain ?
  WORD  domain name
Router(config)# ethernet cfm domain dname ?
  level
Router(config)# ethernet cfm domain dname level ?
  <0-7>  maintenance level number
Router(config)# ethernet cfm domain dname level 7 ?
  <cr>

Router(config)# snmp-server file-transfer access-group 10 ?
  protocol  protocol options
  <cr>

Router(config)# logging host ?
  Hostname or A.B.C.D  IP address of the syslog server
  ipv6                 Configure IPv6 syslog server
```

Understanding Enable and Enable Secret Passwords

Some privileged EXEC commands are used for actions that impact the system, and it is recommended that you set a password for these commands to prevent unauthorized use. Two types of passwords, enable (not encrypted) and enable secret (encrypted), can be set. The following commands set these passwords and are issued in global configuration mode:

- **enable** *password*
- **enable secret** *password*

Using an enable secret password is recommended because it is encrypted and more secure than the enable password. When you use an enable secret password, text is encrypted (unreadable) before it is written to the config.text file. When you use an enable password, the text is written as entered (readable) to the config.text file.

Each type of password is case sensitive, can contain from 1 to 25 uppercase and lowercase alphanumeric characters, and can start with a numeral. Spaces are also valid password characters; for example, "two words" is a valid password. Leading spaces are ignored, but trailing spaces are recognized.

Note Both password commands have numeric keywords that are single integer values. If you choose a numeral for the first character of your password followed by a space, the system will read the number as if it were the numeric keyword and not as part of your password.

When both passwords are set, the enable secret password takes precedence over the enable password.

To remove a password, use the **no** form of the commands: **no enable** *password* or **no enable secret** *password*.

For more information about password recovery procedures for Cisco products, see http://www.cisco.com/en/US/products/sw/iosswrel/ps1831/products_tech_note09186a00801746e6.shtml.

Using the Command History Feature

The command history feature saves, in a command history buffer, the commands that you enter during a session. The default number of saved commands is 10, but the number is configurable within the range of 0 to 256. This command history feature is particularly useful for recalling long or complex commands.

To change the number of commands saved in the history buffer for a terminal session, issue the **terminal history size** command:

```
Router# terminal history size num
```

A command history buffer is also available in line configuration mode with the same default and configuration options. To set the command history buffer size for a terminal session in line configuration mode, issue the **history** command:

```
Router(config-line)# history [size num]
```

To recall commands from the history buffer, use the following methods:

- Press Ctrl-P or the Up Arrow key—Recalls commands beginning with the most recent command. Repeat the key sequence to recall successively older commands.
- Press Ctrl-N or the Down Arrow key—Recalls the most recent commands in the history buffer after they have been recalled using Ctrl-P or the Up Arrow key. Repeat the key sequence to recall successively more recent commands.

  Note The arrow keys function only on ANSI-compatible terminals such as the VT100.

- Issue the **show history** command in user EXEC or privileged EXEC mode—Lists the most recent commands that you entered. The number of commands that are displayed is determined by the setting of the **terminal history size** and **history** commands.

The command history feature is enabled by default. To disable this feature for a terminal session, issue the **terminal no history** command in user EXEC or privileged EXEC mode or the **no history** command in line configuration mode.

Abbreviating Commands

Typing a complete command name is not always required for the command to execute. The CLI recognizes an abbreviated command when the abbreviation contains enough characters to uniquely identify the command. For example, the **show version** command can be abbreviated as **sh ver**. It cannot be abbreviated as **s ver** because **s** could mean **show**, **set**, or **systat**. The **sh v** abbreviation also is not valid because the **show** command has **vrrp** as a keyword in addition to **version**. (Command and keyword examples are from Cisco IOS Release 12.4(13)T.)

Using Aliases for CLI Commands

To save time and the repetition of entering the same command multiple times, you can use a command alias. An alias can be configured to do anything that can be done at the command line, but an alias cannot move between modes, type in passwords, or perform any interactive functions.

Table 4 shows the default command aliases.

Table 4 Default Command Aliases

Command Alias	Original Command
h	help
lo	logout
p	ping
s	show
u or **un**	undebug
w	where

To create a command alias, issue the **alias** command in global configuration mode. The syntax of the command is **alias** *mode command-alias original-command*. Following are some examples:

- Router(config)# **alias exec prt partition**—privileged EXEC mode
- Router(config)# **alias configure sb source-bridge**—global configuration mode
- Router(config)# **alias interface rl rate-limit**—interface configuration mode

To view both default and user-created aliases, issue the **show alias** command.

For more information about the **alias** command, see http://www.cisco.com/en/US/docs/ios/fundamentals/command/reference/cf_a1.html.

Using the no and default Forms of Commands

Most configuration commands have a **no** form that is used to reset a command to its default value or disable a feature or function. For example, the **ip routing** command is enabled by default. To disable this command, you would issue the **no ip routing** command. To re-enable IP routing, you would issue the **ip routing** command.

Configuration commands may also have a **default** form, which returns the command settings to their default values. For commands that are disabled by default, using the **default** form has the same effect as using the **no** form of the command. For commands that are enabled by default and have default settings, the **default** form enables the command and returns the settings to their default values.

The **no** form is documented in the command pages of command references. The **default** form is generally documented in the command pages only when the **default** form performs a different function than the plain and **no** forms of the command. To see what **default** commands are available on your system, enter **default ?** in the appropriate command mode.

Using the debug Command

A **debug** command produces extensive output that helps you troubleshoot problems in your network. These commands are available for many features and functions within Cisco IOS software. Some **debug** commands are **debug all**, **debug aaa accounting**, and **debug mpls packets**. To use **debug** commands during a Telnet session with a device, you must first enter the **terminal monitor** command. To turn off debugging completely, you must enter the **undebug all** command.

For more information about **debug** commands, see the *Cisco IOS Debug Command Reference* at http://www.cisco.com/en/US/docs/ios/debug/command/reference/db_book.html.

Caution Debugging is a high priority and high CPU utilization process that can render your device unusable. Use **debug** commands only to troubleshoot specific problems. The best times to run debugging are during periods of low network traffic and when few users are interacting with the network. Debugging during these periods decreases the likelihood that the **debug** command processing overhead will affect network performance or user access or response times.

Filtering Output Using Output Modifiers

Many commands produce lengthy output that may use several screens to display. Using output modifiers, you can filter this output to show only the information that you want to see.

The following three output modifiers are available:

- **begin** *regular-expression*—Displays the first line in which a match of the regular expression is found and all lines that follow.
- **include** *regular-expression*—Displays all lines in which a match of the regular expression is found.
- **exclude** *regular-expression*—Displays all lines except those in which a match of the regular expression is found.

To use one of these output modifiers, type the command followed by the pipe symbol (|), the modifier, and the regular expression that you want to search for or filter. A regular expression is a case-sensitive alphanumeric pattern. It can be a single character or number, a phrase, or a more complex string.

The following example illustrates how to filter output of the **show interface** command to display only lines that include the expression "protocol."

```
Router# show interface | include protocol

FastEthernet0/0 is up, line protocol is up
Serial4/0 is up, line protocol is up
Serial4/1 is up, line protocol is up
Serial4/2 is administratively down, line protocol is down
Serial4/3 is administratively down, line protocol is down
```

Understanding CLI Error Messages

You may encounter some error messages while using the CLI. Table 5 shows the common CLI error messages.

Table 5 Common CLI Error Messages

Error Message	Meaning	How to Get Help
% Ambiguous command: "show con"	You did not enter enough characters for the command to be recognized.	Reenter the command followed by a space and a question mark (?). The keywords that you are allowed to enter for the command appear.
% Incomplete command.	You did not enter all the keywords or values required by the command.	Reenter the command followed by a space and a question mark (?). The keywords that you are allowed to enter for the command appear.
% Invalid input detected at "^" marker.	You entered the command incorrectly. The caret (^) marks the point of the error.	Enter a question mark (?) to display all the commands that are available in this command mode. The keywords that you are allowed to enter for the command appear.

For more system error messages, see the following document:

- *Cisco IOS Release 12.4T System Message Guide*

Saving Changes to a Configuration

To save changes that you made to the configuration of a device, you must issue the **copy running-config startup-config** command or the **copy system:running-config nvram:startup-config** command. When you issue these commands, the configuration changes that you made are saved to the startup configuration and saved when the software reloads or power to the device is turned off or interrupted. The following example shows the syntax of the **copy running-config startup-config** command:

```
Router# copy running-config startup-config
Destination filename [startup-config]?
```

You press Enter to accept the startup-config filename (the default), or type a new filename and then press Enter to accept that name. The following output is displayed indicating that the configuration was saved.

```
Building configuration...
[OK]
Router#
```

On most platforms, the configuration is saved to NVRAM. On platforms with a Class A flash file system, the configuration is saved to the location specified by the CONFIG_FILE environment variable. The CONFIG_FILE variable defaults to NVRAM.

Additional Information

- "Using the Cisco IOS Command-Line Interface" section of the *Cisco IOS Configuration Fundamentals Configuration Guide*

  http://www.cisco.com/en/US/docs/ios/fundamentals/configuration/guide/cf_cli-basics.html
- Cisco Product/Technology Support

  http://www.cisco.com/go/techdocs
- Support area on Cisco.com (also search for documentation by task or product)

  http://www.cisco.com/en/US/support/index.html
- Software Download Center (downloads; tools; licensing, registration, advisory, and general information) (requires Cisco.com user ID and password)

  http://www.cisco.com/kobayashi/sw-center/
- Error Message Decoder, a tool to help you research and resolve error messages for Cisco IOS software

  http://www.cisco.com/pcgi-bin/Support/Errordecoder/index.cgi
- Command Lookup Tool, a tool to help you find detailed descriptions of Cisco IOS commands (requires Cisco.com user ID and password)

  http://tools.cisco.com/Support/CLILookup
- Output Interpreter, a troubleshooting tool that analyzes command output of supported **show** commands

  https://www.cisco.com/pcgi-bin/Support/OutputInterpreter/home.pl

---

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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.

© 2008–2009 Cisco Systems, Inc. All rights reserved.

Frame Relay

Configuring Frame Relay

Feature History

Release	Modification
Cisco IOS	For information about feature support in Cisco IOS software, use Cisco Feature Navigator.

This chapter describes the tasks for configuring Frame Relay on a router or access server. For further general information about Frame Relay, see the chapter "Wide-Area Networking Overview" at the beginning of this book.

For a complete description of the Frame Relay commands mentioned in this chapter, refer to the chapter "Frame Relay Commands" in the *Cisco IOS Wide-Area Networking Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the section "Identifying Supported Platforms" in the chapter "Using Cisco IOS Software."

For information on the following related topics, see the corresponding chapters in other Cisco publications:

Task	Resource
Sending DDR traffic over Frame Relay	"Configuring Legacy DDR Spokes" and "Configuring Legacy DDR Hubs" chapters in the "Dial-on-Demand Routing Configuration" part in the *Cisco IOS Dial Technologies Configuration Guide*
Installing software on a new router or access server by downloading from a central server over an interface that supports Frame Relay	"Loading and Maintaining System Images" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*
Using AutoInstall over Frame Relay	"Using Autoinstall and Setup" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

Task	Resource
Configuring transparent bridging between devices over a Frame Relay network	"Configuring Transparent Bridging" chapter in the *Cisco IOS Bridging and IBM Networking Configuration Guide*
Configuring source-route bridging between SNA devices over a Frame Relay network	"Configuring Source-Route Bridging" chapter in the *Cisco IOS Bridging and IBM Networking Configuration Guide*
Configuring serial tunnel (STUN) and block serial tunnel encapsulation between devices over a Frame Relay network	"Configuring Serial Tunnel and Block Serial Tunnel" chapter in the *Cisco IOS Bridging and IBM Networking Configuration Guide*
Configuring access between SNA devices over a Frame Relay network	"Configuring SNA Frame Relay Access Support" chapter in the *Cisco IOS Bridging and IBM Networking Configuration Guide*
Configuring Voice over Frame Relay Using FRF.11 and FRF.12	"Configuring Voice over Frame Relay" chapter in the *Cisco IOS Voice, Video, and Fax Configuration Guide*
Configuring low latency queueing, PVC interface priority queueing, and link fragmentation and interleaving using multilink PPP for Frame Relay	*Cisco IOS Quality of Service Solutions Configuration Guide*

Cisco Frame Relay MIB

The Cisco Frame Relay MIB adds extensions to the standard Frame Relay MIB (RFC 1315). It provides additional link-level and virtual circuit (VC)-level information and statistics that are mostly specific to Cisco Frame Relay implementation. This MIB provides SNMP network management access to most of the information covered by the **show frame-relay** commands such as, **show frame-relay lmi**, **show frame-relay pvc**, **show frame-relay map**, and **show frame-relay svc**.

Frame Relay Hardware Configurations

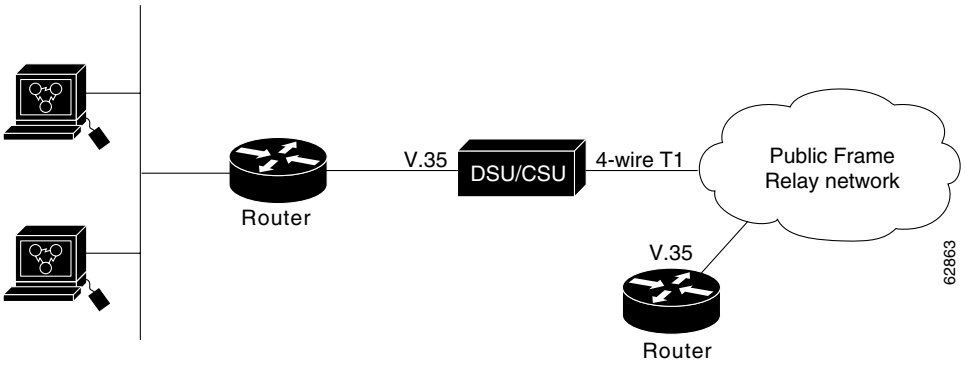
You can create Frame Relay connections using one of the following hardware configurations:

- Routers and access servers connected directly to the Frame Relay switch
- Routers 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 Routers can connect to Frame Relay networks either by direct connection to a Frame Relay switch or through CSU/DSUs. However, a single router 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. Figure 1 illustrates the connections among the components.

Figure 1 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 Configuration Task List

You must follow certain required, basic steps to enable Frame Relay for your network. In addition, you can customize Frame Relay for your particular network needs and monitor Frame Relay connections. The following sections outline these tasks:

- Enabling Frame Relay Encapsulation on an Interface (Required)
- Configuring Dynamic or Static Address Mapping (Required)

Note Frame Relay encapsulation is a prerequisite for any Frame Relay commands on an interface.

The tasks described in the following sections are used to enhance or customize your Frame Relay:

- Configuring the LMI (Optional)
- Configuring Frame Relay SVCs (Optional)
- Configuring Frame Relay Traffic Shaping (Optional)
- Configuring Frame Relay Switching (Optional)
- Customizing Frame Relay for Your Network (Optional)
- Monitoring and Maintaining the Frame Relay Connections (Optional)

See the section "Frame Relay Configuration Examples" at the end of this chapter for ideas about how to configure Frame Relay on your network. See the chapter "Frame Relay Commands" in the *Cisco IOS Wide-Area Networking Command Reference* for information about the Frame Relay commands listed in the following tasks. Use the index or search online for documentation of other commands.

Enabling Frame Relay Encapsulation on an Interface

To enable Frame Relay encapsulation on the interface level, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies the interface, and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation frame-relay [ietf]`	Enables and specifies the Frame Relay encapsulation method.

Frame Relay supports encapsulation of all supported protocols in conformance with RFC 1490, allowing interoperability among multiple vendors. Use the Internet Engineering Task Force (IETF) form of Frame Relay encapsulation if your router 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.

For an example of enabling Frame Relay encapsulation on an interface, see the section "IETF Encapsulation Examples" later in this chapter.

Configuring Dynamic or Static Address Mapping

Dynamic address mapping uses Frame Relay Inverse ARP to request the next-hop protocol address for a specific connection, given its known DLCI. Responses to Inverse ARP requests are entered in an address-to-DLCI mapping table on the router or access server; the 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, but 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. See the section "Disabling or Reenabling Frame Relay Inverse ARP" later in this chapter for more information.

See the following sections for further details on configuring dynamic or static address mapping:

- Configuring Dynamic Address Mapping
- Configuring Static Address Mapping

Configuring Dynamic Address Mapping

Inverse ARP is enabled by default for all protocols enabled on the physical interface. Packets are not sent out for protocols that are not enabled on the interface.

Because Inverse ARP is enabled by default, no additional command is required to configure dynamic mapping on an interface.

Configuring Static Address Mapping

A static map links a specified next-hop protocol address to a specified DLCI. Static mapping removes the need for Inverse 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 if the router at the other end either does not support Inverse ARP at all or does not support Inverse ARP for a specific protocol that you want to use over Frame Relay.

To establish static mapping according to your network needs, use one of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)#` **`frame-relay map`** *`protocol protocol-address dlci`* [**`broadcast`**] [**`ietf`**] [**`cisco`**]	Maps between a next-hop protocol address and DLCI destination address.
`Router(config-if)#` **`frame-relay map clns`** *`dlci`* [**`broadcast`**]	Defines a DLCI used to send ISO CLNS frames.
`Router(config-if)#` **`frame-relay map bridge`** *`dlci`* [**`broadcast`**] [**`ietf`**]	Defines a DLCI destination bridge.

The supported protocols and the corresponding keywords to enable them are as follows:

- IP—**ip**
- DECnet—**decnet**
- AppleTalk—**appletalk**
- XNS—**xns**
- Novell IPX—**ipx**
- VINES—**vines**
- ISO CLNS—**clns**

You can greatly 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.

For examples of establishing static address mapping, refer to the section "Static Address Mapping Examples" later in this chapter.

Configuring the LMI

Beginning with Cisco IOS Release 11.2, the 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 are no longer required to configure the LMI explicitly.

See the following sections for further details on configuring the LMI:

- Activating LMI Autosense
- Explicitly Configuring the LMI

For information on using Enhanced Local Management Interface with traffic shaping, see the section "Configuring Frame Relay Traffic Shaping" later in this chapter.

For an example of configuring the LMI, see the section "Pure Frame Relay DCE Example" later in this chapter.

Activating LMI Autosense

LMI autosense is active in the following situations:

- The router 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 DTE.
- The LMI type is not explicitly configured.

See the following sections for additional information concerning activating LMI autosense:

- Status Request
- Status Messages
- LMI Autosense
- Configuration Options

Status Request

When LMI autosense is active, it sends out a full status request, in all three LMI types, to the switch. The order is ANSI, ITU, cisco, but it is done in rapid succession. Cisco IOS 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 elicit a reply (status message) from the switch. The router will decode the format of the reply and configure itself automatically. If more than one reply is received, the router will configure itself with the type of the last received reply. This is to accommodate intelligent switches that can handle multiple formats simultaneously.

LMI Autosense

If 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 will attempt 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 under way is that **debug frame lmi** is turned on. At every N391 interval, the user will now see three rapid status inquiries coming out of the serial interface: one in ANSI, one in ITU, and one in cisco LMI-type.

Configuration Options

No configuration options are provided; LMI autosense is transparent to the user. You can turn off LMI autosense by explicitly configuring an LMI type. The LMI type must be written into NVRAM so that next time the router 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.

Explicitly Configuring the LMI

Frame Relay software supports the industry-accepted standards for addressing the LMI, including the Cisco specification. If you want to configure the LMI and thus deactivate LMI autosense, perform the tasks in the following sections:

- Setting the LMI Type (Required)
- Setting the LMI Keepalive Interval (Required)
- Setting the LMI Polling and Timer Intervals (Optional)

Setting the LMI Type

If the router 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 needs 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, and ITU-T Q.933 Annex A. To do so, use the following commands beginning in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# frame-relay lmi-type {ansi | cisco | q933a}`	Sets the LMI type.
Step 2	`Router# copy nvram:startup-config destination`	Writes the LMI type to NVRAM.

For an example of setting the LMI type, see the section "Pure Frame Relay DCE Example" later in this chapter.

Setting the LMI Keepalive Interval

A keepalive interval must be set to configure the LMI. By default, this interval is 10 seconds and, according to the LMI protocol, must be less than the corresponding interval on the switch. To set the keepalive interval, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# keepalive number`	Sets the LMI keepalive interval.

To disable keepalives on networks that do not utilize LMI, use the **no keepalive** interface configuration command. For an example of how to specify an LMI keepalive interval, see the section "Two Routers in Static Mode Example" later in this chapter.

Setting the LMI Polling and Timer Intervals

You can set various optional counters, intervals, and thresholds to fine-tune the operation of your LMI DTE and DCE devices. Set these attributes by using one or more of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay lmi-n392dce` *threshold*	Sets the DCE and Network-to-Network Interface (NNI) error threshold.
`Router(config-if)# frame-relay lmi-n393dce` *events*	Sets the DCE and NNI monitored events count.
`Router(config-if)# frame-relay lmi-t392dce` *seconds*	Sets the polling verification timer on a DCE or NNI interface.
`Router(config-if)# frame-relay lmi-n391dte` *keep-exchanges*	Sets a full status polling interval on a DTE or NNI interface.
`Router(config-if)# frame-relay lmi-n392dte` *threshold*	Sets the DTE or NNI error threshold.
`Router(config-if)# frame-relay lmi-n393dte` *events*	Sets the DTE and NNI monitored events count.

See the chapter "Frame Relay Commands" in the *Cisco IOS Wide-Area Networking Command Reference* for polling and timing interval commands.

Configuring Frame Relay SVCs

Access to Frame Relay networks is made through private leased lines at speeds ranging from 56 kbps to 45 Mbps. Frame Relay is a connection-oriented packet-transfer mechanism that establishes VCs between endpoints.

Switched virtual circuits (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.

SVCs can coexist with PVCs in the same sites and routers. For example, routers at remote branch offices might set up PVCs to the central headquarters for frequent communication, but set up SVCs with each other as needed for intermittent communication. As a result, any-to-any communication can be set up without any-to-any PVCs.

On SVCs, quality of service (QoS) elements can be specified on a call-by-call basis to request network resources.

SVC support is offered in the Enterprise image on Cisco platforms that include a serial or HSSI interface.

You must have the following services before Frame Relay SVCs can operate:

- Frame Relay SVC support by the service provider—The service provider's switch must be capable of supporting SVC operation.
- Physical loop connection—A leased line or dedicated line must exist between the router (DTE) and the local Frame Relay switch.

For examples of configuring Frame Relay SVCs, see the section "SVC Configuration Examples" later in this chapter.

Operating SVCs

SVC operation requires that the Data Link layer (Layer 2) be set up, running ITU-T Q.922 Link Access Procedures to Frame mode bearer services (LAPF), prior to signalling for an SVC. Layer 2 sets itself up as soon as SVC support is enabled on the interface, if both the line and the line protocol are up. When the SVCs are configured and demand for a path occurs, the Q.933 signalling sequence is initiated. Once the SVC is set up, data transfer begins.

Q.922 provides a reliable link layer for Q.933 operation. All Q.933 call control information is transmitted over DLCI 0; this DLCI is also used for the management protocols specified in ANSI T1.617 Annex D or Q.933 Annex A.

You must enable SVC operation at the interface level. Once it is enabled at the interface level, it is enabled on any subinterfaces on that interface. One signalling channel, DLCI 0, is set up for the interface, and all SVCs are controlled from the physical interface.

Enabling Frame Relay SVC Service

To enable Frame Relay SVC service and set up SVCs, perform the tasks in the following sections. The subinterface tasks are not required, but offer additional flexibility for SVC configuration and operation. The LAPF tasks are not required and not recommended unless you understand thoroughly the impacts on your network.

- Configuring SVCs on a Physical Interface (Required)
- Configuring SVCs on a Subinterface (Optional)
- Configuring a Map Class (Required)
- Configuring a Map Group with E.164 or X.121 Addresses (Required)
- Associating the Map Class with Static Protocol Address Maps (Required)
- Configuring LAPF Parameters (Optional)

For examples of configuring Frame Relay SVCs, see the section "SVC Configuration Examples" later in this chapter.

Configuring SVCs on a Physical Interface

To enable SVC operation on a Frame Relay interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies the physical interface.
Step 2	`Router(config-if)# ip address ip-address mask`	Specifies the interface IP address, if needed.
Step 3	`Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation on the interface.
Step 4	`Router(config-if)# map-group group-name`	Assigns a map group to the interface.
Step 5	`Router(config-if)# frame-relay svc`	Enables Frame Relay SVC support on the interface.

Map group details are specified with the **map-list** command.

Configuring SVCs on a Subinterface

To configure Frame Relay SVCs on a subinterface, complete all the commands in the preceding section, except assigning the map group. After the physical interface is configured, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number.subinterface-number {multipoint | point-to-point}`	Specifies a subinterface configured for SVC operation.
Step 2	`Router(config-subif)# ip address ip-address mask`	Specifies the subinterface IP address, if needed.
Step 3	`Router(config-subif)# map-group group-name`	Assigns a map group to the subinterface.

Configuring a Map Class

Perform the following tasks to configure a map class:

- Specify the map class name. (Required)
- Specify a custom queue list for the map class. (Optional)
- Specify a priority queue list for the map class. (Optional)
- Enable BECN feedback to throttle the output rate on the SVC for the map class. (Optional)
- Set nondefault QoS values for the map class (no need to set the QoS values; default values are provided). (Optional)

To configure a map class, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# map-class frame-relay map-class-name`	Specifies Frame Relay map class name and enters map class configuration mode.
Step 2	`Router(config-map-class)# frame-relay custom-queue-list list-number`	Specifies a custom queue list to be used for the map class.
Step 3	`Router(config-map-class)# frame-relay priority-group list-number`	Assigns a priority queue to VCs associated with the map class.
Step 4	`Router(config-map-class)# frame-relay adaptive-shaping [becn | foresight]`[1]	Enables the type of BECN feedback to throttle the frame-transmission rate.
Step 5	`Router(config-map-class)# frame-relay cir in bps`	Specifies the inbound committed information rate (CIR), in bits per second.
Step 6	`Router(config-map-class)# frame-relay cir out bps`	Specifies the outbound CIR, in bits per second.
Step 7	`Router(config-map-class)# frame-relay mincir in bps`[2]	Sets the minimum acceptable incoming CIR, in bits per second.
Step 8	`Router(config-map-class)# frame-relay mincir out bps`[2]	Sets the minimum acceptable outgoing CIR, in bits per second.
Step 9	`Router(config-map-class)# frame-relay bc in bits`[2]	Sets the incoming committed burst size (Bc), in bits.
Step 10	`Router(config-map-class)# frame-relay bc out bits`[2]	Sets the outgoing Bc, in bits.
Step 11	`Router(config-map-class)# frame-relay be in bits`[2]	Sets the incoming excess burst size (Be), in bits.
Step 12	`Router(config-map-class)# frame-relay be out bits`[2]	Sets the outgoing Be, in bits.
Step 13	`Router(config-map-class)# frame-relay idle-timer seconds`[2]	Sets the idle timeout interval, in seconds.

1. This command replaces the **frame-relay becn-response-enable** command, which will be removed in a future Cisco IOS release. If you use the **frame-relay becn-response-enable** command in scripts, you should replace it with the **frame-relay adaptive-shaping becn** command.

2. The **in** and **out** keywords are optional. Configuring the command without the **in** and **out** keywords will apply that value to both the incoming and the outgoing traffic values for the SVC setup. For example, **frame-relay cir 56000** applies 56000 to both incoming and outgoing traffic values for setting up the SVC.

You can define multiple map classes. A map class is associated with a static map, not with the interface or subinterface itself. Because of the flexibility this association allows, you can define different map classes for different destinations.

Configuring a Map Group with E.164 or X.121 Addresses

After you have defined a map group for an interface, you can associate the map group with a specific source and destination address to be used. You can specify E.164 addresses or X.121 addresses for the source and destination. To specify the map group to be associated with a specific interface, use the following command in global configuration mode:

Command	Purpose
`Router(config)# map-list` *map-group-name* `source-addr` {`e164` | `x121`} *source-address* `dest-addr` {`e164` | `x121`} *destination-address*	Specifies the map group associated with specific source and destination addresses for the SVC.

Associating the Map Class with Static Protocol Address Maps

To define the protocol addresses under a **map-list** command and associate each protocol address with a specified map class, use the **class** command. Use this command for each protocol address to be associated with a map class. To associate a map class with a protocol address, use the following command in map list configuration mode:

Command	Purpose
`Router(config-map-list)#` *protocol protocol-address* `class` *class-name* [`ietf`] [`broadcast` [`trigger`]]	Specifies a destination protocol address and a Frame Relay map class name from which to derive QoS information.

The **ietf** keyword specifies RFC 1490 encapsulation; the **broadcast** keyword specifies that broadcasts must be carried. The **trigger** keyword, which can be configured only if **broadcast** is also configured, enables a broadcast packet to trigger an SVC. If an SVC already exists that uses this map class, the SVC will carry the broadcast.

Configuring LAPF Parameters

Frame Relay Link Access Procedure for Frame Relay (LAPF) commands are used to tune Layer 2 system parameters to work well with the Frame Relay switch. Normally, you do not need to change the default settings. However, if the Frame Relay network indicates that it does not support the Frame Reject frame (FRMR) at the LAPF Frame Reject procedure, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# no frame-relay lapf frmr`	Selects not to send FRMR frames at the LAPF Frame Reject procedure.

By default, the Frame Reject frame is sent at the LAPF Frame Reject procedure.

Note Manipulation of Layer 2 parameters is not recommended if you do not know well the resulting functional change. For more information, refer to the ITU-T Q.922 specification for LAPF.

If you must change Layer 2 parameters for your network environment and you understand the resulting functional change, use the following commands as needed:

Command	Purpose
`Router(config-if)# frame-relay lapf k` *number*	Sets the LAPF window size k.
`Router(config-if)# frame-relay lapf n200` *retries*	Sets the LAPF maximum retransmission count N200.
`Router(config-if)# frame-relay lapf n201` *bytes*	Sets maximum length of the Information field of the LAPF I frame N201, in bytes.
`Router(config-if)# frame-relay lapf t200` *tenths-of-a-second*	Sets the LAPF retransmission timer value T200, in tenths of a second.
`Router(config-if)# frame-relay lapf t203` *seconds*	Sets the LAPF link idle timer value T203 of DLCI 0, in seconds.

Configuring Frame Relay Traffic Shaping

Traffic shaping applies to both PVCs and SVCs. For information about creating and configuring SVCs, see the section "Configuring Frame Relay SVCs" earlier in this chapter.

To configure Frame Relay traffic shaping, perform the tasks in the following sections:

- Enabling Frame Relay Encapsulation on an Interface (earlier in this chapter)
- Defining VCs for Different Types of Traffic
- Enabling Frame Relay Traffic Shaping on the Interface
- Configuring Enhanced Local Management Interface
- Specifying a Traffic-Shaping Map Class for the Interface
- Defining a Map Class with Queueing and Traffic-Shaping Parameters
- Defining Access Lists
- Defining Priority Queue Lists for the Map Class
- Defining Custom Queue Lists for the Map Class

Note Frame Relay traffic shaping is not effective for Layer 2 PVC switching using the **frame-relay route** command.

For examples of configuring Frame Relay traffic shaping, see the section "Frame Relay Traffic Shaping Examples" later in this chapter.

Defining VCs for Different Types of Traffic

By defining separate VCs for different types of traffic and specifying queueing and an outbound traffic rate for each VC, you can provide guaranteed bandwidth for each type of traffic. By specifying different traffic rates for different VCs over the same line, you can perform virtual time division multiplexing. By throttling outbound traffic from high-speed lines in central offices to lower-speed lines in remote locations, you can ease congestion and data loss in the network; enhanced queueing also prevents congestion-caused data loss.

Enabling Frame Relay Traffic Shaping on the Interface

Enabling Frame Relay traffic shaping on an interface enables both traffic shaping and per-VC queueing on all the PVCs and SVCs on the interface. Traffic shaping enables the router to control the circuit's output rate and react to congestion notification information if also configured.

To enable Frame Relay traffic shaping on the specified interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay traffic-shaping`	Enables Frame Relay traffic shaping and per-VC queueing.

Note The default committed information rate (CIR) of 56K will apply in the following situations:
—When traffic shaping is enabled (by using the **frame-relay traffic-shaping** command), but a map class is not assigned to the VC
—When traffic shaping is enabled (by using the **frame-relay traffic-shaping** command) and a map class is assigned to the VC, but traffic-shaping parameters have not been defined in the map class

To configure a map class with traffic-shaping and per-VC queueing parameters, see the sections "Specifying a Traffic-Shaping Map Class for the Interface" and "Defining a Map Class with Queueing and Traffic-Shaping Parameters".

Frame Relay ForeSight

ForeSight is the network traffic control software used in some Cisco switches. The Cisco Frame Relay switch can extend ForeSight messages over a User-to-Network Interface (UNI), passing the backward congestion notification for VCs.

ForeSight allows Cisco Frame Relay routers to process and react to ForeSight messages and adjust VC level traffic shaping in a timely manner.

ForeSight must be configured explicitly on both the Cisco router and the Cisco switch. ForeSight is enabled on the Cisco router when Frame Relay traffic shaping is configured. However, the router's response to ForeSight is not applied to any VC until the **frame-relay adaptive-shaping foresight**

command is added to the VCs map-class. When ForeSight is enabled on the switch, the switch will periodically send out a ForeSight message based on the time value configured. The time interval can range from 40 to 5000 milliseconds.

When a Cisco router receives a ForeSight message indicating that certain DLCIs are experiencing congestion, the Cisco router reacts by activating its traffic-shaping function to slow down the output rate. The router reacts as it would if it were to detect the congestion by receiving a packet with the backward explicit congestion notification (BECN) bit set.

When ForeSight is enabled, Frame Relay traffic shaping will adapt to ForeSight messages and BECN messages.

For an example of configuring Foresight, see the section "Traffic Shaping with ForeSight Example" later in this chapter.

Frame Relay ForeSight Prerequisites

For router ForeSight to work, the following conditions must exist on the Cisco router:

- Frame Relay traffic shaping must be enabled on the interface.
- The traffic shaping for a circuit is adapted to ForeSight.

The following additional condition must exist on the Cisco switch:

- The UNI connecting to the router is Consolidated Link Layer Management (CLLM) enabled, with the proper time interval specified.

Frame Relay router ForeSight is enabled automatically when you use the **frame-relay traffic-shaping** command. However, you must issue the **map-class frame-relay** command and the **frame-relay adaptive-shaping foresight** command before the router will respond to ForeSight and apply the traffic-shaping effect on a specific interface, subinterface, or VC.

Frame Relay Congestion Notification Methods

The difference between the BECN and ForeSight congestion notification methods is that BECN requires a user packet to be sent in the direction of the congested DLCI to convey the signal. The sending of user packets is not predictable and, therefore, not reliable as a notification mechanism. Rather than waiting for user packets to provide the congestion notification, timed ForeSight messages guarantee that the router receives notification before congestion becomes a problem. Traffic can be slowed down in the direction of the congested DLCI.

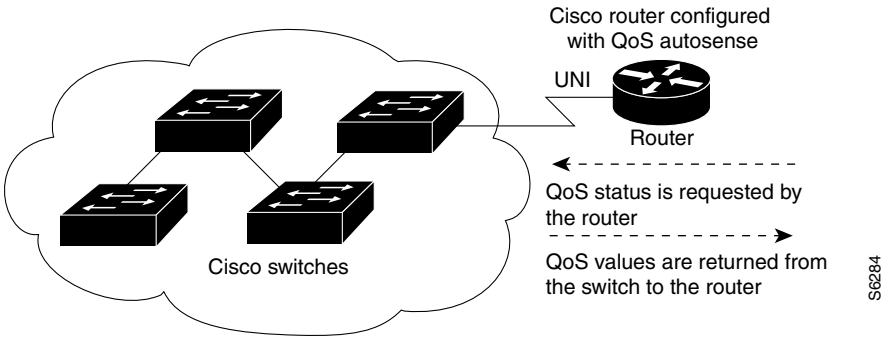
Configuring Enhanced Local Management Interface

Enhanced Local Management Interface (ELMI) allows the router to learn QoS parameters and connectivity information from the Cisco switch and to use this information for traffic shaping, configuration, or management purposes. ELMI simplifies the process of configuring traffic shaping on the router and reduces chances of specifying inconsistent or incorrect values when configuring the router. ELMI works between Cisco routers and Cisco switches (BPX and IGX platforms).

ELMI QoS Autosense

When used in conjunction with traffic shaping, ELMI enables the router to respond to changes in the network dynamically. ELMI enables automated exchange of Frame Relay QoS parameter information between the Cisco router and the Cisco switch. Figure 2 illustrates a Cisco switch and a Cisco router, both configured with ELMI enabled. The switch sends QoS information to the router, which uses it for traffic rate enforcement.

Figure 2 Enhanced Local Management Interface—Sent Between the Cisco Switch and the Cisco Router



Routers can base congestion management and prioritization decisions on known QoS values, such as the Committed Information Rate (CIR), Committed Burst Size (Bc), and Excess Burst Size (Be). The router senses QoS values from the switch and can be configured to use those values in traffic shaping.

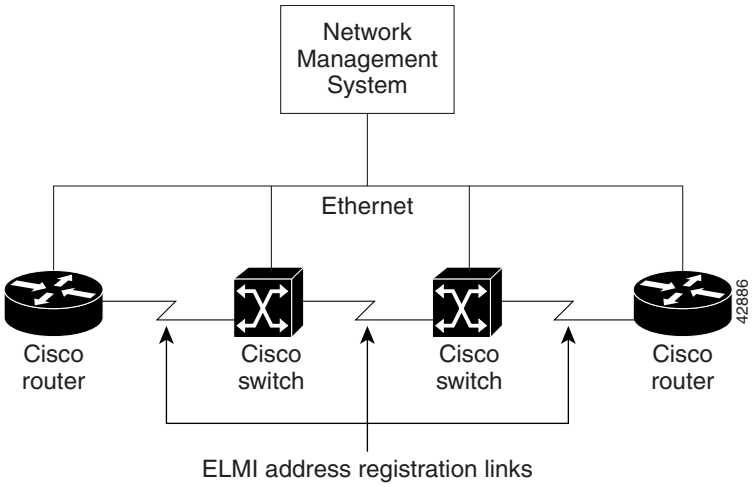
It is not necessary to configure traffic shaping on the interface to enable ELMI, but you may want to do so in order to know the values being used by the switch. If you want the router to respond to the QoS information received from the switch by adjusting the output rate, you must configure traffic shaping on the interface. To configure traffic shaping, use the **frame-relay traffic-shaping** command in interface configuration mode.

ELMI Address Registration

ELMI address registration enables a network management system (NMS) to detect connectivity among Cisco switches and routers in a network using the ELMI protocol. During ELMI version negotiation, neighboring devices exchange their management IP addresses and ifIndex. The NMS polls the devices and uses the Cisco Frame Relay MIB to collect this connectivity information. ELMI address registration allows for autodetection of the complete network topology.

Figure 3 shows a typical network in which ELMI address registration is in use.

Figure 3 Connectivity Detection Using ELMI Address Registration



ELMI address registration takes place on all interfaces on which ELMI is enabled, even if all the interfaces are connected to the same router or switch. The router periodically sends a version inquiry message with version information, the management IP address, and ifIndex to the switch. The switch sends its management IP address and ifIndex using the version status message. When the management IP address of the switch changes, an asynchronous ELMI version status message is immediately sent to the neighboring device.

Note The ELMI address registration mechanism does not check for duplicate or illegal addresses.

When ELMI is enabled, the router automatically chooses the IP address of one of the interfaces to use for ELMI address registration purposes. The router will choose the IP address of an Ethernet interface first, and then serial and other interfaces. You have the option to use the IP address chosen by the router or to disable the autoaddress mechanism and configure the management IP address yourself. You can also choose to disable ELMI address registration on a specific interface or on all interfaces.

To configure ELMI, complete the tasks in the following sections:

- Enabling ELMI (Required)
- Disabling Automatic IP Address Selection (Optional)
- Configuring the IP Address to Be Used for ELMI Address Registration (Optional)
- Enabling ELMI Address Registration on an Interface (Optional)
- Verifying ELMI Address Registration (Optional)

For examples of the configurations in this section, see the section "ELMI Configuration Examples" at the end of this chapter.

Enabling ELMI

To enable ELMI, use the following commands beginning in interface configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies the physical interface.
Step 2	`Router(config-if)# encapsulation frame-relay [cisco | ietf]`	Enables Frame Relay encapsulation on the interface.
Step 3	`Router(config-if)# frame-relay QoS-autosense`	Enables ELMI.

Disabling Automatic IP Address Selection

Automatic IP address selection is enabled by default when ELMI is enabled.

To disable the automatic selection of the IP address to be used for ELMI address registration, use the following global configuration command:

Command	Purpose
`Router(config)# no frame-relay address registration auto-address`	Disables the automatic selection of the IP address to be used for ELMI address registration.

Note When automatic IP address selection is disabled and an IP address has not been configured using the **frame-relay address registration ip** global configuration command, the IP address for ELMI address registration will be set to 0.0.0.0.

Configuring the IP Address to Be Used for ELMI Address Registration

To configure the IP address for ELMI address registration, use the following global configuration command:

Command	Purpose
`Router(config)# frame-relay address registration ip address`	Configures the IP address to be used for ELMI address registration.

Note Automatic IP address selection is disabled when you configure the management IP address using the **frame-relay address registration ip** global configuration command.

Enabling ELMI Address Registration on an Interface

To enable ELMI address registration on an interface, use the following interface configuration command:

Command	Purpose
`Router(config-if)# frame-relay address-reg enable`	Enables ELMI address registration on an interface. To disable ELMI address registration on an interface, use the **no** form of the command.

Verifying ELMI Address Registration

To verify that ELMI address registration is configured correctly, use the following privileged EXEC configuration command:

Command	Purpose
`Router# show frame-relay qos-autosense [interface interface]`	Displays the QoS values and ELMI address registration information sensed from the switch.

Specifying a Traffic-Shaping Map Class for the Interface

If you specify a Frame Relay map class for a main interface, all the VCs on its subinterfaces inherit all the traffic-shaping parameters defined for the class.

To specify a map class for the specified interface, use the following command beginning in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay class map-class-name`	Specifies a Frame Relay map class for the interface.

You can override the default for a specific DLCI on a specific subinterface by using the **class** VC configuration command to assign the DLCI explicitly to a different class. See the section "Configuring Frame Relay Subinterfaces" for information about setting up subinterfaces.

For an example of assigning some subinterface DLCIs to the default class and assigning others explicitly to a different class, see the section "Frame Relay Traffic Shaping Examples" later in this chapter.

Defining a 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 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.

To define a map class, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# map-class frame-relay map-class-name`	Specifies a map class to define.
Step 2	`Router(config-map-class)# frame-relay traffic-rate average [peak]`	Defines the traffic rate for the map class.
Step 3	`Router(config-map-class)# frame-relay custom-queue-list list-number`	Specifies a custom queue list.
Step 4	`Router(config-map-class)# frame-relay priority-group list-number`	Specifies a priority queue list.
Step 5	`Router(config-map-class)# frame-relay adaptive-shaping {becn | foresight}`[1]	Selects BECN or ForeSight as congestion backward-notification mechanism to which traffic shaping adapts.

1. This command replaces the **frame-relay becn-response-enable** command, which will be removed in a future Cisco IOS release. If you use the **frame-relay becn-response-enable** command in scripts, you should replace it with the **frame-relay adaptive-shaping** software command.

For an example of map class backward compatibility and interoperability, see the section "Backward Compatibility Example" later in this section.

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.

Defining Priority Queue Lists for the Map Class

You can define a priority list for a protocol and you can also define a default priority list. The number used for a specific priority list ties the list to the Frame Relay priority group defined for a specified map class.

For example, if you enter the **frame relay priority-group 2** command for the map class "fast_vcs" and then you enter the **priority-list 2 protocol decnet high** command, that priority list is used for the "fast_vcs" map class. The average and peak traffic rates defined for the "fast_vcs" map class are used for DECnet traffic.

Defining Custom Queue Lists for the Map Class

You can define a queue list for a protocol and a default queue list. You can also specify the maximum number of bytes to be transmitted in any cycle. The number used for a specific queue list ties the list to the Frame Relay custom queue list defined for a specified map class.

For example, if you enter the **frame relay custom-queue-list 1** command for the map class "slow_vcs" and then you enter the **queue-list 1 protocol ip list 100** command, that queue list is used for the "slow_vcs" map class; **access-list 100** definition is also used for that map class and queue. The average and peak traffic rates defined for the "slow_vcs" map class are used for IP traffic that meets the **access list 100** criteria.

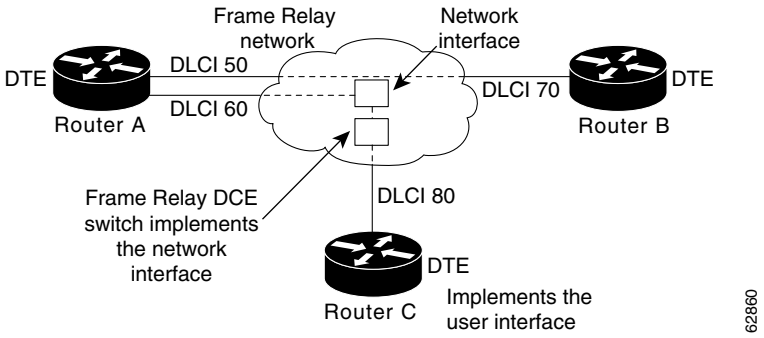
Configuring Frame Relay Switching

Frame Relay switching is a means of switching packets based on the DLCI, which can be considered the Frame Relay equivalent of a MAC address. You perform switching by configuring your Cisco router or access server into a Frame Relay network. There are two parts to a Frame Relay network:

- Frame Relay DTE (the router or access server)
- Frame Relay DCE switch

Figure 4 illustrates Frame Relay switched networks. Routers A, B, and C are Frame Relay DTEs connected to each other via a Frame Relay network.

Figure 4 Frame Relay Switched Network



Frame Relay switching is supported on the following interface types:

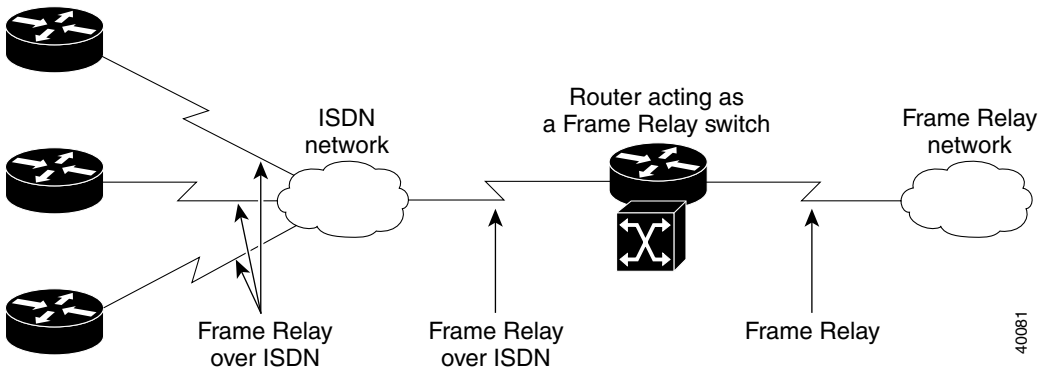
- Serial interfaces
- ISDN interfaces

Note Frame Relay switching is not supported on subinterfaces.

Frame Relay Switching over ISDN B Channels

Frame Relay switching over ISDN B channels enables you to transport Frame Relay data over ISDN. This feature allows small offices to be hubbed out of larger offices rather than being connected directly to the core network. The hub router acts as a Frame Relay switch, switching between ISDN and serial interfaces, as shown in Figure 5.

Figure 5 Router Used As a Frame Relay Switch over ISDN



Frame Relay switching over ISDN provides the following functionality:

- LMI is supported on ISDN Frame Relay DCE interfaces.
- A single BRI/PRI interface can use a combination of switched PVCs and terminated Frame Relay PVCs.
- Frame Relay switching supports both leased-line ISDN, on which a B channel is permanently connected, and switched ISDN, on which B channels may be dynamically set up and torn down.

Note the following restrictions for Frame Relay switching over ISDN:

- Frame Relay traffic shaping is not supported on ISDN interfaces.
- The router configured for Frame Relay switching over ISDN cannot initiate the ISDN call.
- PVC-level congestion management is not supported over ISDN. Interface-level congestion management is supported.
- When Frame Relay switching is performed by using a dialer profile, encapsulation of the underlying physical (BRI) interface must be configured as high-level data link control (HDLC).

Frame Relay Switching Configuration Task List

To configure Frame Relay switching, perform the tasks in the following sections. Each task is identified as required or optional.

- Enabling Frame Relay Switching (Required)
- Enabling Frame Relay Encapsulation on an Interface (earlier in this chapter) (Required)
- Configuring a Frame Relay DTE Device, DCE Switch, or NNI Support (Required)
- Creating Switched PVCs (Required)
- Identifying a PVC As Switched (Optional)
- Configuring Frame Relay Traffic Shaping on Switched PVCs (Optional)
- Configuring Traffic Policing on UNI DCE Devices (Optional)
- Configuring Congestion Management on Switched PVCs (Optional)
- Configuring FRF.12 Fragmentation on Switched PVCs (Optional)
- Verifying Frame Relay Switching (Optional)
- Troubleshooting Frame Relay Switching (Optional)

For configuration examples of Frame Relay switching, see the section "Frame Relay Switching Examples" later in this chapter.

Enabling Frame Relay Switching

You must enable packet switching before you can configure it on a Frame Relay DTE or DCE, or with Network-to-Network Interface (NNI) support. Do so by using the following command in global configuration mode before configuring the switch type:

Command	Purpose
`Router(config)# frame-relay switching`	Enables Frame Relay switching.

Configuring a Frame Relay DTE Device, DCE Switch, or NNI Support

You can configure an interface as a DTE device or a DCE switch, or as a switch connected to a switch to support NNI connections. (DTE is the default.) To do so, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay intf-type [dce | dte | nni]`	Configures a Frame Relay DTE device or DCE switch.

Creating Switched PVCs

To create a switched PVC over ISDN, or to create a switched PVC on which traffic shaping, traffic policing, and congestion management can be configured, use the following command in global configuration mode:

Command	Purpose
`Router(config)# connect connection-name interface dlci interface dlci`	Defines connections between Frame Relay PVCs.

To create a switched PVC with a static route, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay route in-dlci interface out-interface-type out-interface-number out-dlci`	Specifies a static route for PVC switching.

Note Static routes cannot be configured over tunnel interfaces on the Cisco 800 series, 1600 series, and 1700 series platforms. Static routes can only be configured over tunnel interfaces on platforms that have the Enterprise feature set.

Identifying a PVC As Switched

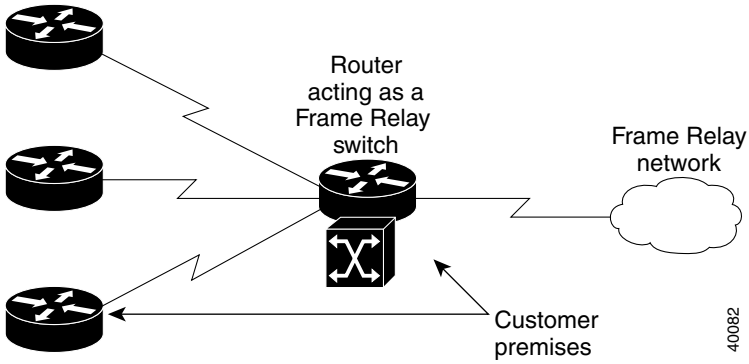
Before you can associate a map class with a switched PVC, you must identify the PVC as being switched. To identify a PVC as switched, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay interface-dlci dlci switched`	Identifies a PVC as switched.

Configuring Frame Relay Traffic Shaping on Switched PVCs

Applying Frame Relay traffic shaping to switched PVCs enables a router to be used as a Frame Relay port concentrator in front of a Frame Relay switch. The Frame Relay switch will shape the concentrated traffic before sending it into the network. Figure 6 shows the network configuration.

Figure 6 Router Used As a Frame Relay Port Concentrator



When you configure traffic shaping, you will define the traffic-shaping parameters in a Frame Relay map class and then attach the map class to the interface or a single switched PVC. All the traffic-shaping map-class parameters are applicable to switched PVCs: namely, Bc, Be, CIR, minimum CIR, average rate, peak rate, and adaptive shaping.

Frame Relay traffic shaping must be enabled on the interface before traffic-shaping map-class parameters will be effective. Note that when you enable Frame Relay traffic shaping, all PVCs, switched and terminated, will be shaped on that interface. Switched PVCs that are not associated with a map class will inherit shaping parameters from the interface or use default values.

For the specific configuration tasks for Frame Relay traffic shaping, see the section "Configuring Frame Relay Traffic Shaping" earlier in this chapter.

Configuring Traffic Policing on UNI DCE Devices

Traffic policing prevents congestion on incoming PVCs by discarding or setting the DE bit on packets that exceed specified traffic parameters.

To configure traffic policing on UNI DCE devices, perform the following tasks:

- Enabling Frame Relay Policing
- Configuring Frame Relay Policing Parameters

Enabling Frame Relay Policing

To enable Frame Relay policing on a interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay policing`	Enables Frame Relay policing on all switched PVCs on the interface.

Configuring Frame Relay Policing Parameters

To configure policing parameters in a Frame Relay map class, use one or more of the following commands in map-class configuration mode:

Command	Purpose
`Router(config-map-class)# frame-relay cir {in | out} bps`	Sets the CIR for a Frame Relay PVC, in bits per second.
`Router(config-map-class)# frame-relay bc {in | out} bits`	Sets the committed burst size for a Frame Relay PVC, in bits.
`Router(config-map-class)# frame-relay be {in | out} bits`	Sets the excess burst size for a Frame Relay PVC, in bits.
`Router(config-map-class)# frame-relay tc milliseconds`	Sets the measurement interval for policing incoming traffic on a PVC when the CIR is zero, in milliseconds.

You can associate the map class with the interface or individual switched PVCs. Switched PVCs that are not associated with a map class will inherit policing parameters from the interface.

If you use a map class to configure both traffic policing and shaping, use the **in** keyword to specify incoming traffic for policing and the **out** keyword to specify outgoing traffic for shaping. If you configure shaping on one segment of a switched PVC and policing on the other, the shaping parameters will be derived from the policing parameters unless you specifically define shaping parameters in the map class.

Configuring Congestion Management on Switched PVCs

Frame Relay congestion management can be used to manage outgoing traffic congestion on switched PVCs. When Frame Relay congestion management is enabled, one way that the router manages congestion is by setting backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN) bits on packets. When a switched PVC or interface is congested, packets experiencing congestion are marked with the FECN bit, and packets traveling in the reverse direction are marked with the BECN bit. When these bits reach a user device at the end of the network, the user device can react to the ECN bits and adjust the flow of traffic.

When the output interface queue reaches or exceeds the ECN excess threshold, all Frame Relay DE bit packets on all PVCs crossing that interface will be marked with FECN or BECN, depending on their direction of travel. When the queue reaches or exceeds the ECN committed threshold, all Frame Relay packets will be marked with FECN or BECN.

A second way the router manages congestion is by discarding Frame Relay packets that are marked with the discard eligible (DE) bit and that exceed a specified level of congestion.

When the queue reaches or exceeds the DE threshold, Frame Relay packets with the DE bit will be discarded rather than queued.

You can define two levels of congestion. The first level applies to individual PVCs transmitting traffic in excess of the committed information rate (CIR). The second level applies to all PVCs at an interface. This scheme allows you to adjust the congestion on PVCs transmitting above the CIR before applying congestion management measures to all PVCs.

Congestion management parameters can be configured on the output interface queue and on traffic-shaping queues.

To configure congestion management on switched PVCs, perform the tasks in the following sections:

- Configuring Frame Relay Congestion Management on the Interface
- Configuring Frame Relay Congestion Management on Traffic-Shaping Queues

Configuring Frame Relay Congestion Management on the Interface

To configure Frame Relay congestion management on all switched PVCs on an interface, use the following commands beginning in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# frame-relay congestion management`	Enables Frame Relay congestion management on all switched PVCs on an interface and enters Frame Relay congestion management configuration mode.
Step 2	`Router(config-fr-congest)# threshold de` *percentage*	Configures the threshold at which DE-marked packets will be discarded from switched PVCs on the output interface.
Step 3	`Router(config-fr-congest)# threshold ecn {bc | be}` *percentage*	Configures the threshold at which ECN bits will be set on packets in switched PVCs on the output interface.

Configuring Frame Relay Congestion Management on Traffic-Shaping Queues

To configure Frame Relay congestion management on the traffic-shaping queues of switched PVCs, use one or more of the following commands in map-class configuration mode:

Command	Purpose
`Router(config-map-class)# frame-relay congestion threshold de` *percentage*	Configures the threshold at which DE-marked packets will be discarded from the traffic-shaping queue of a switched PVC.
`Router(config-map-class)# frame-relay congestion threshold ecn` *percentage*	Configures the threshold at which ECN bits will be set on packets in the traffic-shaping queue of a switched PVC.
`Router(config-map-class)# frame-relay holdq` *queue-size*	Configures the maximum size of a traffic-shaping queue on a switched PVC.

Configuring FRF.12 Fragmentation on Switched PVCs

The FRF.12 Implementation Agreement allows long data frames to be fragmented into smaller pieces. This process allows real-time traffic and non-real-time traffic to be carried together on lower-speed links without causing excessive delay to the real-time traffic. For further information about FRF.12 fragmentation, see the section "End-to-End FRF.12 Fragmentation" later in this chapter.

Some Frame Relay access devices do not support the FRF.12 standard for end-to-end fragmentation. Large packets sourced from these devices can cause significant serialization delay across low-speed trunks in switched networks. Using FRF.12 fragmentation can help prevent this delay. An edge router that receives large packets from a Frame Relay access device will fragment those packets before transmitting them across the switched network. The edge router that receives the fragmented packets will

reassemble those packets before sending them to a Frame Relay access device that does not support FRF.12. If the receiving Frame Relay access device does support FRF.12, the router will transmit the fragmented packets without reassembling them.

Note the following conditions and restrictions on FRF.12 fragmentation on switched PVCs:

- Frame Relay traffic shaping must be enabled.
- Interface queueing must be dual FIFO queueing or PVC interface priority queueing.
- Switched PVCs must be configured using the **connect** command.
- If the Frame Relay access device does not support FRF.12 fragmentation, the FRF.12 Support on Switched Frame Relay PVCs 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 switched 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 switched PVCs.

To configure FRF.12 on switched PVCs, use the following map-class configuration command:

Command	Purpose
`Router(config-map-class)# frame-relay fragment` *fragment_size* **switched**	Enables FRF.12 fragmentation on switched Frame Relay PVCs for a Frame Relay map class.

The map class can be associated with one or more switched PVCs.

Verifying Frame Relay Switching

To verify the correct configuration of Frame Relay switching, use one or more of the following commands:

Command	Purpose
`Router#` **show frame-relay fragment** [**interface** *interface*] [*dlci*]	Displays statistics about Frame Relay fragmentation.
`Router#` **show frame-relay pvc** [**interface** *interface*] [*dlci*]	Displays statistics about Frame Relay PVCs including detailed reasons for packet drops on switched PVCs and complete status information for switched NNI PVCs.
`Router#` **show interfaces** [*type number*]	Displays information about the configuration and queue at the interface.

Troubleshooting Frame Relay Switching

To diagnose problems in switched Frame Relay networks, use the following EXEC commands:

Command	Purpose
`Router# debug frame-relay switching [interface interface] [dlci] [interval seconds]`	Displays debug messages for switched Frame Relay PVCs. The **interval** keyword and *seconds* argument sets the interval at which the debug messages will be displayed.
`Router# show frame-relay pvc [interface interface] [dlci]`	Displays statistics about Frame Relay PVCs, including detailed reasons for packet drops on switched PVCs and complete status information for switched NNI PVCs.

Customizing Frame Relay for Your Network

Perform the tasks in the following sections to customize Frame Relay:

- Configuring Frame Relay End-to-End Keepalives
- Configuring PPP over Frame Relay
- Configuring Frame Relay Subinterfaces
- Disabling or Reenabling Frame Relay Inverse ARP
- Creating a Broadcast Queue for an Interface
- Configuring Frame Relay Fragmentation
- Configuring Payload Compression
- Configuring TCP/IP Header Compression
- Configuring Real-Time Header Compression with Frame Relay Encapsulation
- Configuring Discard Eligibility
- Configuring DLCI Priority Levels

Configuring Frame Relay End-to-End Keepalives

Frame Relay end-to-end keepalives enable monitoring of PVC status for network monitoring or backup applications and are configurable on a per-PVC basis with configurable timers. The Frame Relay switch within the local PVC segment deduces the status of the remote PVC segment through a Network-to-Network Interface (NNI) and reports the status to the local router. If LMI support within the switch is not end-to-end, end-to-end keepalives are the only source of information about the remote router. End-to-end keepalives verify that data is getting through to a remote device via end-to-end communication.

Each PVC connecting two end devices needs two separate keepalive systems, because the upstream path may not be the same as the downstream path. One system sends out requests and handles responses to those requests—the send side—while the other system handles and replies to requests from the device at the other end of the PVC—the receive side. The send side on one device communicates with the receive side on the other device, and vice versa.

The send side sends out a keepalive request and waits for a reply to its request. If a reply is received before the timer expires, a send-side Frame Relay end-to-end keepalive is recorded. If no reply is received before the timer expires, an error event is recorded. A number of the most recently recorded events are examined. If enough error events are accumulated, the keepalive status of the VC is changed from up to down, or if enough consecutive successful replies are received, the keepalive status of the VC is changed from down to up. The number of events that will be examined is called the *event window.*

The receive side is similar to the send side. The receive side waits for requests and sends out replies to those requests. If a request is received before the timer expires, a success event is recorded. If a request is not received, an error event is recorded. If enough error events occur in the event window, the PVC state will be changed from up to down. If enough consecutive success events occur, the state will be changed from down to up.

End-to-end keepalives can be configured in one of four modes: bidirectional, request, reply, or passive-reply.

- In bidirectional mode, both the send side and the receive side are enabled. The send side of the device sends out and waits for replies to keepalive requests from the receive side of the other PVC device. The receive side of the device waits for and replies to keepalive requests from the send side of the other PVC device.
- In request mode, only the send side is enabled, and the device sends out and waits for replies to its keepalive requests.
- In reply mode, only the receive side is enabled, and the device waits for and replies to keepalive requests.
- In passive-reply mode, the device only responds to keepalive requests, but does not set any timers or keep track of any events.

Because end-to-end keepalives allow traffic flow in both directions, they can be used to carry control and configuration information from end to end. Consistency of information between end hosts is critical in applications such as those relating to prioritized traffic and Voice over Frame Relay. Whereas SVCs can convey such information within end-to-end signalling messages, PVCs will benefit from a bidirectional communication mechanism.

End-to-end keepalives are derived from the Frame Relay LMI protocol and work between peer Cisco communications devices. The key difference is that rather than running over the signalling channel, as is the case with LMI, end-to-end keepalives run over individual data channels.

Encapsulation of keepalive packets is proprietary; therefore, the feature is available only on Cisco devices running a software release that supports the Frame Relay End-to-End Keepalive feature.

You must configure both ends of a VC to send keepalives. If one end is configured as bidirectional, the other end must also be configured as bidirectional. If one end is configured as request, the other end must be configured as reply or passive-reply. If one end is configured as reply or passive-reply, the other end must be configured as request

To configure Frame Relay end-to-end keepalives, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# map-class frame-relay map-class-name`	Specifies a map class for the VC.
Step 2	`Router(config-map-class)# frame-relay end-to-end keepalive mode {bidirectional | request | reply | passive-reply}`	Specifies Frame Relay end-to-end keepalive mode.

The four modes determine the type of keepalive traffic each device sends and responds to:

- In bidirectional mode, the device will send keepalive requests to the other end of the VC and will respond to keepalive requests from the other end of the VC.
- In request mode, the device will send keepalive requests to the other end of the VC.
- In reply mode, the device will respond to keepalive requests from the other end of the VC.
- In passive-reply mode, the device will respond to keepalive requests from the other end of the VC, but will not track errors or successes.

For an example of configuring bidirectional or request modes with default values, see the section "End-to-End Keepalive Bidirectional Mode with Default Configuration Example" or "End-to-End Keepalive Request Mode with Default Configuration Example," and for an example of configuring request mode with modified values, see the section "End-to-End Keepalive Request Mode with Modified Configuration Example" later in this chapter.

You can modify the end-to-end keepalives default parameter values by using any of the following map-class configuration commands:

Command	Purpose
`Router(config-map-class)# frame-relay end-to-end keepalive error-threshold {send | receive} count`	Modifies the number of errors needed to change the keepalive state from up to down.
`Router(config-map-class)# frame-relay end-to-end keepalive event-window {send | receive} count`	Modifies the number of recent events to be checked for errors.
`Router(config-map-class)# frame-relay end-to-end keepalive success-events {send | receive} count`	Modifies the number of consecutive success events required to change the keepalive state from down to up.
`Router(config-map-class)# frame-relay end-to-end keepalive timer {send | receive} interval`	Modifies the timer interval.

Verifying Frame Relay End-to-End Keepalives

To monitor the status of Frame Relay end-to-end keepalives, use the following command in EXEC configuration mode:

Command	Purpose
`Router# show frame-relay end-to-end keepalive interface`	Shows the status of Frame Relay end-to-end keepalives.

Configuring PPP over Frame Relay

Point-to-point protocol (PPP) over Frame Relay allows a router to establish end-to-end PPP sessions over Frame Relay. This is done over a PVC, which is the only circuit currently supported. The PPP session does not occur unless the associated Frame Relay PVC is in an "active" state. The Frame Relay PVC can coexist with other circuits using different Frame Relay encapsulation methods, such as RFC 1490 and the Cisco proprietary method, over the same Frame Relay link. There can be multiple PPP over Frame Relay circuits on one Frame Relay link.

One PPP connection resides on one virtual access interface. This is internally created from a virtual template interface, which contains all necessary PPP and network protocol information and is shared by multiple virtual access interfaces. The virtual access interface is coexistent with the creation of the

Frame Relay circuit when the corresponding DLCI is configured. Hardware compression and fancy queueing algorithms, such as weighted fair queueing, custom queueing, and priority queueing, are not applied to virtual access interfaces.

PPP over Frame Relay is only supported on IP. IP datagrams are transported over the PPP link using RFC 1973 compliant Frame Relay framing. The frame format is shown in Figure 7.

Figure 7 PPP over Frame Relay Frame Format

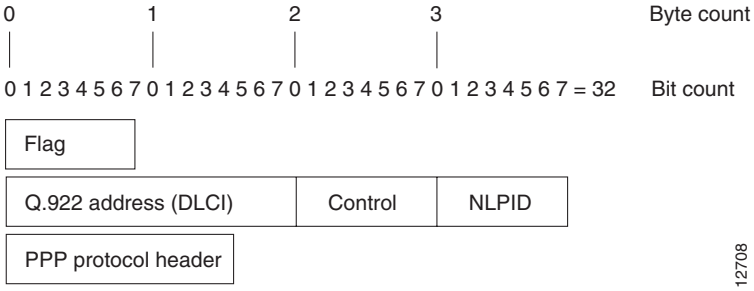


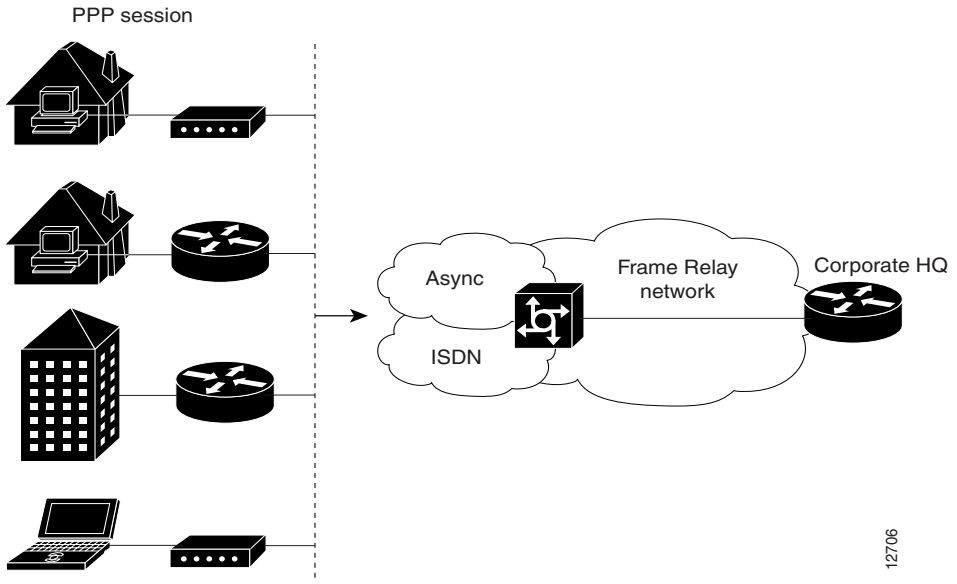
Table 1 lists the Frame Relay frame format components illustrated in Figure 7.

Table 1 PPP Frame Relay Frame Format Descriptions

Field	Description
Flag	A single byte that indicates the beginning or end of a frame.
Address	A two-byte field that indicates the logical connection that maps to the physical channel; the DLCI.
Control	A single byte that calls for transmission of user data. PPP over Frame Relay uses a value of 0X03, which indicates that the frame is an unnumbered information (UI) frame.
NLPID	Network layer protocol ID—a single byte that uniquely identifies a PPP packet to Frame Relay.
PPP protocol	PPP packet type.

Figure 8 shows remote users running PPP to access their Frame Relay corporate networks.

Figure 8 PPP over Frame Relay Scenario



Enabling PPP over Frame Relay

Before PPP over Frame Relay is configured, Frame Relay must be enabled on the router using the **encapsulation frame-relay** command. The only task required in order to implement PPP over Frame Relay is to configure the interface with the locally terminated PVC and the associated virtual template for PPP and IP, as described in the following section.

After configuring Frame Relay encapsulation on the Cisco router or access server, you must configure the physical interface with the PVC and apply a virtual template with PPP encapsulation to the DLCI.

To configure the physical interface that will carry the PPP session and link it to the appropriate virtual template interface, perform the following task in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay interface-dlci dlci [ppp virtual-template-name]`	Defines the PVC and maps it to the virtual template.

For an example of configuring PPP over Frame Relay, see the section "PPP over Frame Relay Examples" or "PPP over Frame Relay DCE Example" later in this chapter.

Configuring Frame Relay Subinterfaces

For a general explanation of Frame Relay subinterfaces, read the following section, "Understanding Frame Relay Subinterfaces."

To configure the Frame Relay subinterface and define subinterface addressing, perform the tasks in the following sections:

- Defining Subinterface Addressing (Required)
- Configuring Transparent Bridging for Frame Relay (Optional)

- Configuring a Backup Interface for a Subinterface (Optional)

For a selection of subinterface configuration examples, see the section "Subinterface Examples" later in this chapter.

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 talk to station B, and station B can talk to station C, then station A should be able to talk to station C directly. Transitivity is true on LANs, but not on Frame Relay networks unless A is directly connected to C.

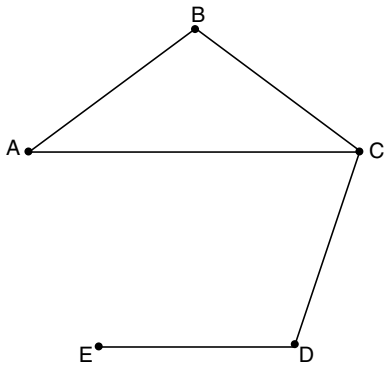
Additionally, certain protocols such as AppleTalk and transparent bridging cannot be supported on partially meshed networks because they require *split horizon*. Split horizon is a routing technique in which a packet received on an interface cannot be sent from the same interface even if received and transmitted on different VCs.

Configuring Frame Relay subinterfaces ensures that a single physical interface is treated as multiple virtual interfaces. This treatment allows you to overcome split horizon rules. Packets received on one virtual interface can be forwarded to another virtual interface even if they are configured on the same physical interface.

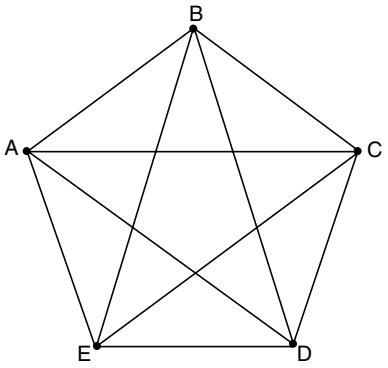
Subinterfaces address the limitations of Frame Relay networks by providing a way to subdivide a partially meshed Frame Relay network into a number of smaller, fully meshed (or point-to-point) subnetworks. Each subnetwork is assigned its own network number and appears to the protocols as if it were reachable through a separate interface. (Note that point-to-point subinterfaces can be unnumbered for use with IP, reducing the addressing burden that might otherwise result.)

Figure 9 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 be made to work with certain protocols (for example, IP), but will not work at all with other protocols (for example, AppleTalk) because nodes C and D will not relay the packet out the same interface on which it was received. One way to make this network work fully is to create a fully meshed network (network B), but doing so requires a large number of PVCs, which may not be economically feasible.

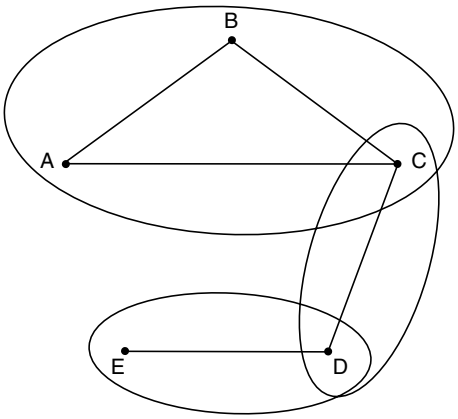
Figure 9 Using Subinterfaces to Provide Full Connectivity on a Partially Meshed Frame Relay Network



Network A: Partially meshed Frame Relay network without full connectivity



Network B: Fully meshed Frame Relay network with full connectivity



Network C: Partially meshed Frame Relay network with full connectivity (configuring subinterfaces)

62873

Using subinterfaces, you can subdivide the Frame Relay network into three 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 two 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.

Subinterfaces can be configured for multipoint or point-to-point communication. (There is no default.) To configure subinterfaces on a Frame Relay network, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number.subinterface-number {multipoint | point-to-point}`	Creates a point-to-point or multipoint subinterface.
Step 2	`Router(config-subif)# encapsulation frame-relay`	Configures Frame Relay encapsulation on the serial interface.

For an example of configuring Frame Relay subinterfaces, see the section "Subinterface Examples" later in this chapter.

Defining 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. For multipoint subinterfaces, the destinations can be dynamically resolved through the use of Frame Relay Inverse ARP or can be statically mapped through the use of the **frame-relay map** command.

See the following sections for further information about subinterface addressing:

- Addressing on Point-to-Point Subinterfaces
- Addressing on Multipoint Subinterfaces
- Accepting Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces
- Configuring Static Address Mapping on Multipoint Subinterfaces

For subinterface addressing examples, see the section "Static Address Mapping Examples" later in this chapter.

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:

Command	Purpose
`Router(config-subif)# frame-relay interface-dlci dlci`	Associates the selected point-to-point subinterface with a DLCI.

Note This command is typically used on subinterfaces; however, it can also be applied to main interfaces. The **frame-relay interface-dlci** 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 PVC on a multipoint subinterface.

For an explanation of the many available options for this command, refer to the *Cisco IOS Wide-Area Networking Command Reference*.

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 router or access server. Instead, you can simply avoid using that subinterface number and use a different subinterface number.

Addressing on Multipoint Subinterfaces

If you specified a multipoint subinterface in the preceding procedure, perform the configuration tasks in the following sections:

- Accepting Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces
- Configuring Static Address Mapping on Multipoint Subinterfaces

You can configure some protocols for dynamic address mapping and others for static address mapping.

Accepting Inverse ARP for Dynamic Address Mapping on Multipoint Subinterfaces

Dynamic address mapping uses Frame Relay Inverse ARP to request the next-hop protocol address for a specific connection, given a DLCI. Responses to Inverse ARP requests are entered in an address-to-DLCI mapping table on the router or access server; the table is then used to supply the next-hop protocol address or the DLCI for outgoing traffic.

Since the physical interface is now configured as multiple subinterfaces, you must provide information that distinguishes a subinterface from the physical interface and associates a specific subinterface with a specific DLCI.

To associate a specific multipoint subinterface with a specific DLCI, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay interface-dlci dlci`	Associates a specified multipoint subinterface with a DLCI.

Inverse ARP is enabled by default for all protocols it supports, but 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 the protocol is not supported on the other end of the connection. See the section "Disabling or Reenabling Frame Relay Inverse ARP" later in this chapter for more information.

Because Inverse ARP is enabled by default for all protocols that it supports, no additional command is required to configure dynamic address mapping on a subinterface.

For an example of configuring Frame Relay multipoint subinterfaces with dynamic address mapping, see the section "Frame Relay Multipoint Subinterface with Dynamic Addressing Example" later in this chapter.

Configuring Static Address Mapping on Multipoint Subinterfaces

A static map links a specified next-hop protocol address to a specified DLCI. Static mapping removes the need for Inverse 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 if the router at the other end either does not support Inverse ARP at all or does not support Inverse ARP for a specific protocol that you want to use over Frame Relay.

To establish static mapping according to your network needs, use one of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay map protocol protocol-address dlci [broadcast] [ietf] [cisco]`	Maps between a next-hop protocol address and DLCI destination address.
`Router(config-if)# frame-relay map clns dlci [broadcast]`	Defines a DLCI used to send ISO CLNS frames.
`Router(config-if)# frame-relay map bridge dlci [broadcast] [ietf]`	Defines a DLCI destination bridge.

The supported protocols and the corresponding keywords to enable them are as follows:

- IP—**ip**
- DECnet—**decnet**
- AppleTalk—**appletalk**
- XNS—**xns**
- Novell IPX—**ipx**
- VINES—**vines**
- ISO CLNS—**clns**

The **broadcast** keyword is required for routing protocols such as OSI protocols and the Open Shortest Path First (OSPF) protocol. See 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.

For an example of establishing static address mapping on multipoint subinterfaces, see the sections "Two Routers in Static Mode Example," "AppleTalk Routing Example," "DECnet Routing Example," and "IPX Routing Example" later in this chapter.

Configuring Transparent Bridging for Frame Relay

You can configure transparent bridging for point-to-point or point-to-multipoint subinterfaces on Frame Relay encapsulated serial and HSSI interfaces. See the following sections for further information:

- Point-to-Point Subinterfaces
- Point-to-Multipoint Interfaces

For an example of Frame Relay transparent bridging, see the section "Transparent Bridging Using Subinterfaces Example" later in this chapter.

Note All PVCs configured on a subinterface belong to the same bridge group.

Point-to-Point Subinterfaces

To configure transparent bridging for point-to-point subinterfaces, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies an interface.
Step 2	`Router(config-if)# encapsulation frame-relay`	Configures Frame Relay encapsulation on the interface.
Step 3	`Router(config)# interface type number:subinterface-number point-to-point`	Specifies a subinterface.
Step 4	`Router(config-subif)# frame-relay interface-dlci dlci`	Associates a DLCI with the subinterface.
Step 5	`Router(config-subif)# bridge-group bridge-group`	Associates the subinterface with a bridge group.

Point-to-Multipoint Interfaces

To configure transparent bridging for point-to-multipoint subinterfaces, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies an interface.
Step 2	`Router(config-if)# encapsulation frame-relay`	Configures Frame Relay encapsulation.
Step 3	`Router(config)# interface type number:subinterface-number multipoint`	Specifies a subinterface.
Step 4	`Router(config-subif)# frame-relay map bridge dlci [broadcast] [ietf]`	Defines a DLCI destination bridge.
Step 5	`Router(config-subif)# bridge-group bridge-group`	Associates the subinterface with a bridge group.

Configuring a 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 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 will have precedence over the subinterface's backup interface in the case of complete loss of connectivity with the Frame Relay network. As a result, a subinterface backup is activated only if the main interface is up, or 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.

To configure a backup interface for a Frame Relay subinterface, use the following commands beginning in global configuration mode:

	Command	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.
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

Frame Relay Inverse ARP is a method of building dynamic address mappings in Frame Relay networks running AppleTalk, Banyan VINES, DECnet, IP, Novell IPX, and XNS. Inverse ARP allows the router or access server to discover the protocol address of a device associated with the 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 DLCI (see the section "Configuring Dynamic or Static Address Mapping" earlier in this chapter for further information).

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, because there is only a single destination and discovery is not required.

To select Inverse ARP or disable it, use one of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay inverse-arp protocol dlci`	Enables Frame Relay Inverse ARP for a specific protocol and DLCI pair, only if it was previously disabled.
`Router(config-if)# no frame relay inverse-arp protocol dlci`	Disables Frame Relay Inverse ARP for a specific protocol and DLCI pair.

Creating a Broadcast Queue for an Interface

Very large Frame Relay networks may have performance problems when many DLCIs terminate in a single router or access server that must replicate routing updates and service advertising updates on each DLCI. The updates can consume access-link bandwidth and cause significant latency variations in user traffic; the updates can also consume interface buffers and lead to higher packet rate loss for both user data and routing updates.

To avoid such problems, you can create a special broadcast queue for an interface. The broadcast queue is managed independently of the normal interface queue, has its own buffers, and has a configurable size and service rate.

A broadcast queue is given a maximum transmission rate (throughput) limit measured in both bytes per second and packets per second. The queue is serviced to ensure that no more than this maximum is provided. The broadcast queue has priority when transmitting at a rate below the configured maximum, and hence has a guaranteed minimum bandwidth allocation. The two transmission rate limits are intended to avoid flooding the interface with broadcasts. The actual transmission rate limit in any second is the first of the two rate limits that is reached.

To create a broadcast queue, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)#` **`frame-relay broadcast-queue`** *`size byte-rate packet-rate`*	Creates a broadcast queue for an interface.

Configuring Frame Relay Fragmentation

Cisco has developed three types of Frame Relay fragmentation, which are described in the following sections:

- End-to-End FRF.12 Fragmentation
- Frame Relay Fragmentation Using FRF.11 Annex C
- Cisco-Proprietary Fragmentation

For further information about Frame Relay fragmentation, see the following sections:

- Frame Relay Fragmentation and Hardware Compression Interoperability
- Frame Relay Fragmentation Conditions and Restrictions

End-to-End FRF.12 Fragmentation

The purpose of end-to-end FRF.12 fragmentation is to support 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. 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 in order for fragmentation to work.

To configure end-to-end FRF.12 fragmentation, perform the tasks in the following sections. Each task is identified as required or optional.

- Configuring End-to-End FRF.12 Fragmentation (Required)
- Verifying the Configuration of End-to-End FRF.12 Fragmentation (Optional)

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:

Command	Purpose
`Router(config)# map-class frame-relay map-class-name`	Specifies a map class to define QoS values for a Frame Relay SVC or PVC.
`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.

The map class can be applied to one or many PVCs.

Note When Frame Relay fragmentation is configured, WFQ or LLQ is mandatory. If a map class is configured for Frame Relay fragmentation and the queueing 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 Quality of Service Solutions Configuration Guide*, Release 12.2.

For an example of configuring FRF.12 fragmentation, see the section "FRF.12 Fragmentation Example" later in this chapter.

For information about configuring FRF.12 fragmentation on switched Frame Relay PVCs, see the section "Configuring FRF.12 Fragmentation on Switched PVCs" earlier in this chapter.

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 kb/s).

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 kb/s port speeds, the fragment size needs to be set for the 64 kb/s 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 kb/s, the recommended fragment size (for 10 ms serialization delay) is 80 bytes. If the lowest link speed is 128 kb/s, 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.

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
`Router# show frame-relay fragment [interface interface] [dlci]`	Displays Frame Relay fragmentation information.
`Router# show frame-relay pvc [interface interface] [dlci]`	Displays statistics about PVCs for Frame Relay interfaces.

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.

With FRF.11, all data packets contain fragmentation headers, regardless of size. This form of fragmentation is not recommended for use with Voice over IP (VoIP).

See the chapter "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. When the **vofr cisco** command is configured on a DLCI and fragmentation is enabled on a map class, the Cisco 2600 series, 3600 series, and 7200 series routers can interoperate as tandem nodes (but cannot perform call termination) with Cisco MC3810 concentrators running Cisco IOS releases prior to 12.0(3)XG or 12.0(4)T.

To configure Cisco-proprietary voice encapsulation, use the **vofr cisco** command. You must then configure a map class to enable voice traffic on the PVCs.

See the chapter "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 Fragmentation and Hardware Compression Interoperability

FRF.12, FRF.11 Annex C, and Cisco-proprietary fragmentation can be used with FRF.9 or data-stream hardware compression on interfaces and virtual circuits (VCs) using Cisco-proprietary or Internet Engineering Task Force (IETF) encapsulation types.

When payload compression and Frame Relay fragmentation are used at the same time, payload compression is always performed before fragmentation.

Frame Relay fragmentation can be used with the following hardware compression modules:

- Cisco 2600 AIM-COMPR2

- Cisco 3620 and 3640 NM-COMPR
- Cisco 3660 AIM-COMPR4
- Cisco 7200 SA-COMPR

Voice over Frame Relay and Voice over IP packets will not be payload-compressed when Frame Relay fragmentation is configured.

Note On VCs using IETF encapsulation, FRF.9 hardware and software compression will work with Frame Relay fragmentation but will not work with header compression.

For more information about FRF.9 or data-stream compression, see the section "Configuring Payload Compression" later in this chapter.

For an example of Frame Relay fragmentation and hardware compression configured on the same interface, see the "Frame Relay Fragmentation with Hardware Compression Example" later in this chapter.

Frame Relay Fragmentation Conditions and Restrictions

When Frame Relay fragmentation is configured, the following conditions and restrictions apply:

- WFQ and LLQ at the PVC level are the only queueing strategies that can be used.
- Frame Relay traffic shaping (FRTS) must be configured to enable Frame Relay fragmentation (except on the Cisco 7500 series routers on which Versatile Interface Processor-Based Distributed FRF.11 and FRF.12 is enabled).
- VoFR frames are never fragmented, regardless of size.
- When end-to-end FRF.12 fragmentation is used, the VoIP packets will not include the FRF.12 header, provided the size of the VoIP packet is smaller than the fragment size configured. However, when FRF.11 Annex C or Cisco-proprietary fragmentations are used, VoIP packets will include the fragmentation header.
- If fragments arrive out of sequence, packets are dropped.

Note Fragmentation is performed after frames are removed from the WFQ.

Configuring Payload Compression

There are three types of payload compression:

- Packet-by-packet payload compression
- Standard-based FRF.9 payload compression
- Cisco-proprietary data-stream payload compression

To configure payload compression in your Frame Relay network, perform the tasks in the following sections:

- Configuring Packet-by-Packet Payload Compression
- Configuring Standard-Based FRF.9 Compression
- Configuring Data-Stream Compression

- Verifying Payload Compression

Configuring Packet-by-Packet Payload Compression

You can configure payload compression on point-to-point or multipoint interfaces or subinterfaces. Payload compression uses the Stacker method to predict what the next character in the frame will be. Because the prediction is done packet by packet, the dictionary is not conserved across packet boundaries.

Payload compression on each VC consumes approximately 40 kilobytes for dictionary memory.

To configure payload compression on a specified multipoint interface or subinterface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay map` *protocol protocol-address dlci* `payload-compression packet-by-packet`	Enables payload compression on a multipoint interface.

To configure payload compression on a specified point-to-point interface or subinterface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay payload-compression packet-by-packet`	Enables payload compression on a point-to-point interface.

Configuring Standard-Based FRF.9 Compression

Frame Relay compression can now occur on the VIP board, on the Compression Service Adapter (CSA), or on the main CPU of the router. FRF.9 is standard-based and therefore provides multivendor compatibility. FRF.9 compression uses relatively higher compression ratios, allowing more data to be compressed for faster transmission. FRF.9 compression provides the ability to maintain multiple decompression/compression histories on a per-DLCI basis.

The CSA hardware has been in use on the Cisco 7200 series and Cisco 7500 series platforms, but it has had no support for Frame Relay compression. The CSA can be used in the Cisco 7200 series or in the second-generation Versatile Interface Processor (VIP2) in all Cisco 7500 series routers. The specific VIP2 model required for the CSA is VIP2-40, which has 2 MB of SRAM and 32 MB of DRAM.

See the following sections for further information on FRF.9 compression:

- Selecting FRF.9 Compression Method
- Configuring FRF.9 Compression Using Map Statements
- Configuring FRF.9 Compression on the Subinterface

Selecting FRF.9 Compression Method

The router enables compression in the following order:

1. If the router contains a compression service adapter, compression is performed in the CSA hardware (hardware compression).

2. If the CSA is not available, compression is performed in the software installed on the VIP2 card (distributed compression).
3. If the VIP2 card is not available, compression is performed in the main processor of the router (software compression).

Configuring FRF.9 Compression Using Map Statements

You can control where you want compression to occur by specifying an interface. To enable FRF.9 compression on a specific CSA, VIP CPU, or host CPU, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies the interface.
Step 2	`Router(config-if)# encapsulation frame-relay`	Specifies Frame Relay as encapsulation type.
Step 3	`Router(config-if)# frame-relay map payload-compression frf9 stac [hardware-options]`	Enables FRF.9 compression.

Configuring FRF.9 Compression on the Subinterface

To configure FRF.9 compression on the subinterface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies the subinterface type and number.
Step 2	`Router(config-subif)# encapsulation frame-relay`	Specifies Frame Relay as encapsulation type.
Step 3	`Router(config-subif)# frame-relay payload-compression frf9 stac [hardware-options]`	Enables FRF.9 compression.

Configuring Data-Stream Compression

Data-stream compression is a proprietary hardware and software compression protocol that can be used on the same VC or interface and IP header compression. Data-stream compression is functionally equivalent to FRF.9 compression and must be used with Cisco-proprietary encapsulation. Frame Relay fragmentation can also be used with data-stream compression.

To configure data-stream compression with IP header compression, perform the tasks in the following sections:

- Configuring Data-Stream Hardware Compression and IP Header Compression on a Point-to-Point Subinterface
- Configuring Data-Stream Hardware Compression and IP Header Compression on a Multipoint Subinterface

Configuring Data-Stream Hardware Compression and IP Header Compression on a Point-to-Point Subinterface

To configure data-stream hardware compression and TCP or Real-Time Transport Protocol (RTP) header compression on a point-to-point subinterface, use the following commands beginning in global configuration mode. Note that when you specify data-stream hardware compression, Cisco-proprietary encapsulation is automatically enabled.

	Command	Purpose
Step 1	`Router(config)# interface type number point-to-point`	Configures a subinterface type and enters subinterface configuration mode.
Step 2	`Router(config-subif)# ip address address mask`	Sets the IP address for an interface.
Step 3	`Router(config-subif)# frame-relay interface-dlci dlci`	Assigns a DLCI to a specified Frame Relay subinterface on the router or access server.
Step 4	`Router(config-subif)# frame-relay payload-compression data-stream stac [hardware-options]`	Enables hardware compression on an interface or subinterface that uses Cisco-proprietary encapsulation.

Step 5	`Router(config-subif)# frame-relay ip tcp header-compression [passive]`
	
	or
	
	`Router(config-subif)# frame-relay ip rtp header-compression [passive]`
		Configures an interface to ensure that the associated PVCs carry outgoing TCP headers in compressed form.
		
		Enables RTP header compression on the physical interface.

For an example of data-stream compression and IP header compression configured on a point-to-point subinterface, see the section "Data-Stream Hardware Compression with TCP/IP Header Compression on a Point-to-Point Subinterface Example" later in this chapter.

Configuring Data-Stream Hardware Compression and IP Header Compression on a Multipoint Subinterface

To configure data-stream hardware compression and TCP or RTP header compression on a multipoint subinterface, use the following commands beginning in global configuration mode. Note that when you specify data-stream hardware compression, Cisco-proprietary encapsulation is automatically enabled.

	Command	Purpose
Step 1	`Router(config)# interface type number multipoint`	Configures a subinterface type and enters subinterface configuration mode.
Step 2	`Router(config-subif)# frame-relay interface-dlci dlci`	Assigns a DLCI to a specified Frame Relay subinterface on the router or access server.

	Command	Purpose
Step 3	`Router(config-subif)# frame-relay map` *protocol protocol-address dlci* [**payload-compression data-stream stac** [*hardware-options*]]	Defines the mapping between a destination protocol address and the DLCI used to connect to the destination address on an interface that uses Cisco-proprietary encapsulation.
Step 4	`Router(config-subif)# frame-relay ip tcp header-compression` [**passive**] or `Router(config-subif)# frame-relay ip rtp header-compression` [**passive**]	Configures an interface to ensure that the associated PVCs carry outgoing TCP headers in compressed form. Enables RTP header compression on the physical interface.

For an example of data-stream compression and IP header compression configured on a multipoint subinterface, see the section "Data-Stream Hardware Compression with TCP/IP Header Compression on a Multipoint Subinterface Example" later in this chapter.

For an example of data-stream compression and IP header compression configured with FRF.12 fragmentation, see the section "Data-Stream Hardware Compression with RTP Header Compression and Frame Relay Fragmentation Example" later in this chapter.

Verifying Payload Compression

To verify that payload compression is working correctly, use the following privileged EXEC commands:

Command	Purpose
`Router#` **show compress**	Displays compression statistics.
`Router#` **show frame-relay pvc** *dlci*	Displays statistics about PVCs for Frame Relay interfaces, including the number of packets in the post-hardware-compression queue.
`Router#` **show traffic-shape queue**	Displays information about the elements queued at a particular time at the DLCI level, including the number of packets in the post-hardware-compression queue.

Configuring TCP/IP Header Compression

TCP/IP header compression, as described by RFC 1144, 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 and 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.

See the following sections for information about configuring or disabling TCP/IP header compression:

- Configuring an Individual IP Map for TCP/IP Header Compression
- Configuring an Interface for TCP/IP Header Compression
- Disabling TCP/IP Header Compression

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.

Configuring an Individual IP Map for TCP/IP Header Compression

Note An interface configured to support TCP/IP header compression cannot also support priority queueing or custom queueing.

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, even 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.

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
`Router(config-if)# 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.

For an example of how to configure TCP header compression on an IP map, see the section "Using an IP Map to Override TCP/IP Header Compression Example" later in this chapter.

Configuring an Interface for TCP/IP Header Compression

You can configure the interface with 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.

To apply TCP/IP header compression to an interface, you must use the following commands in interface configuration mode:

	Command	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.

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 section "Configuring an Individual IP Map for TCP/IP Header Compression."

For an example of how to configure TCP header compression on an interface, see the section "Using an IP Map to Override TCP/IP Header Compression Example" later in this chapter.

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
`Router(config-if)# no frame-relay ip tcp header-compression`	DisablesTCP/IP header compression on all Frame Relay IP maps that are not explicitly configured for TCP header compression.
or	
`Router(config-if)# frame-relay map ip` *`ip-address dlci`* `nocompress`	Disables RTP and TCP/IP header compression on a specified Frame Relay IP map.

For examples of turning off TCP/IP header compression, see the sections "Disabling Inherited TCP/IP Header Compression Example" and "Disabling Explicit TCP/IP Header Compression Example" later in this chapter.

Configuring 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, providing 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. RTP is not intended for data traffic, which uses TCP or UDP.

For configuration tasks for and examples of RTP header compression using Frame Relay encapsulation, see the chapter "Configuring IP Multicast Routing" in the *Cisco IOS IP Configuration Guide*.

The commands for configuring this feature appear in the *Cisco IOS IP Command Reference, Volume 3 of 3: Multicast.*

Configuring Discard Eligibility

Some Frame Relay packets can be set with low priority or low time sensitivity. These 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 DLCI that is affected.

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:

Command	Purpose
`Router(config)# frame-relay de-list` *list-number* {**protocol** *protocol* | **interface** *type number*} *characteristic*	Defines a DE list.

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 User Datagram Protocol (UDP) port, an access list number, or a packet size. See the **frame-relay de-list** command in the *Cisco IOS Wide-Area Networking Command Reference* for further information.

To define a DE group specifying the DE list and DLCI affected, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# frame-relay de-group` *group-number dlci*	Defines a DE group.

Configuring DLCI Priority Levels

DLCI priority levels allow you to separate different types of traffic and provides a traffic management tool for congestion problems caused by following situations:

- Mixing batch and interactive traffic over the same DLCI
- Queueing traffic from sites with high-speed access at destination sites with lower-speed access

Before you configure the DLCI priority levels, perform the following tasks:

- Define a global priority list.
- Enable Frame Relay encapsulation, as described in the section "Enabling Frame Relay Encapsulation on an Interface" earlier in this chapter.
- Define dynamic or static address mapping, as described in the section "Configuring Dynamic or Static Address Mapping" earlier in this chapter.

- Make sure 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, as described in the section "Configuring the LMI" earlier in this chapter.

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 router or access server. In fact, they are independent of the device's priority queues. However, if you enable queueing and use the same DLCIs for queueing, then high-priority DLCIs can be put into high-priority queues.

To configure DLCI priority levels, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# 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
`Router# clear frame-relay-inarp`	Clears dynamically created Frame Relay maps, which are created by the use of Inverse ARP.
`Router# show interfaces serial type number`	Displays information about Frame Relay DLCIs and the LMI.
`Router# show frame-relay lmi [type number]`	Displays LMI statistics.
`Router# show frame-relay map`	Displays the current Frame Relay map entries.
`Router# show frame-relay pvc [type number [dlci]]`	Displays PVC statistics.
`Router# show frame-relay route`	Displays configured static routes.
`Router# show frame-relay traffic`	Displays Frame Relay traffic statistics.
`Router# show frame-relay lapf`	Displays information about the status of LAPF.
`Router# show frame-relay svc maplist`	Displays all the SVCs under a specified map list.

Frame Relay Configuration Examples

The following sections provide examples of Frame Relay configurations:

- IETF Encapsulation Examples

- Static Address Mapping Examples
- Subinterface Examples
- SVC Configuration Examples
- Frame Relay Traffic Shaping Examples
- Backward Compatibility Example
- Booting from a Network Server over Frame Relay Example
- Frame Relay Switching Examples
- Frame Relay End-to-End Keepalive Examples
- PPP over Frame Relay Examples
- Frame Relay Fragmentation Configuration Examples
- Payload Compression Configuration Examples
- TCP/IP Header Compression Examples

IETF Encapsulation Examples

The following sections provide examples of IETF encapsulation on the interface level and on a per-DLCI basis:

- IETF Encapsulation on the Interface Example
- IETF Encapsulation on a Per-DLCI Basis Example

IETF Encapsulation on the Interface Example

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
```

IETF Encapsulation on a Per-DLCI Basis Example

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
```

Static Address Mapping Examples

The following sections provide examples of static address mapping for two routers in static mode and specific examples for IP, AppleTalk, DECnet, and IPX protocols:

- Two Routers in Static Mode Example
- AppleTalk Routing Example
- DECnet Routing Example

- IPX Routing Example

Two Routers in Static Mode Example

The following example shows how to configure two routers for static mode:

Configuration for Router 1

```
interface serial 0
 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 serial 0
 ip address 131.108.64.1 255.255.255.0
 encapsulation frame-relay
 keepalive 10
 frame-relay map ip 131.108.64.2 43
```

AppleTalk Routing Example

The following example shows how to configure two routers to communicate with each other using AppleTalk over a Frame Relay network. Each router has a Frame Relay static address map for the other router. The use of the **appletalk cable-range** command indicates that this is extended AppleTalk (Phase II).

Configuration for Router 1

```
interface serial0
 ip address 172.21.59.24 255.255.255.0
 encapsulation frame-relay
 appletalk cable-range 10-20 18.47
 appletalk zone eng
 frame-relay map appletalk 18.225 100 broadcast
```

Configuration for Router 2

```
interface serial2/3
 ip address 172.21.177.18 255.255.255.0
 encapsulation frame-relay
 appletalk cable-range 10-20 18.225
 appletalk zone eng
 clockrate 2000000
 frame-relay map appletalk 18.47 100 broadcast
```

DECnet Routing Example

The following example sends all DECnet packets destined for address 56.4 out on DLCI 101. In addition, any DECnet broadcasts for interface serial 1 will be sent on that DLCI.

```
decnet routing 32.6
!
interface serial 1
 encapsulation frame-relay
 frame-relay map decnet 56.4 101 broadcast
```

IPX Routing Example

The following example shows how to send packets destined for IPX address 200.0000.0c00.7b21 out on DLCI 102:

```
ipx routing 000.0c00.7b3b
!
interface ethernet 0
 ipx network 2abc
!
interface serial 0
 ipx network 200
 encapsulation frame-relay
 frame-relay map ipx 200.0000.0c00.7b21 102 broadcast
```

Subinterface Examples

The following sections provide Frame Relay subinterface examples and variations appropriate for different routed protocols and bridging:

- Basic Subinterface Example
- Frame Relay Multipoint Subinterface with Dynamic Addressing Example
- IPX Routes over Frame Relay Subinterfaces Example
- Unnumbered IP over a Point-to-Point Subinterface Example
- Transparent Bridging Using Subinterfaces Example

Basic Subinterface Example

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
```

Frame Relay Multipoint Subinterface with Dynamic Addressing Example

The following example configures two multipoint subinterfaces for dynamic address resolution. Each subinterface is provided with an individual protocol address and subnet mask, and the **frame-relay interface-dlci** command associates the subinterface with a specified DLCI. Addresses of remote destinations for each multipoint subinterface will be resolved dynamically.

```
interface serial0
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface serial0.103 multipoint
 ip address 172.21.177.18 255.255.255.0
 frame-relay interface-dlci 300
```

```
!
interface serial0.104 multipoint
 ip address 172.21.178.18 255.255.255.0
 frame-relay interface-dlci 400
```

IPX Routes over Frame Relay Subinterfaces Example

The following example configures a serial interface for Frame Relay encapsulation and sets up multiple IPX virtual networks corresponding to Frame Relay subinterfaces:

```
ipx routing 0000.0c02.5f4f
!
interface serial 0
 encapsulation frame-relay
 interface serial 0.1 multipoint
 ipx network 1
 frame-relay map ipx 1.000.0c07.d530 200 broadcast
 interface serial 0.2 multipoint
 ipx network 2
 frame-relay map ipx 2.000.0c07.d530 300 broadcast
```

For subinterface serial 0.1, the router at the other end might be configured as follows:

```
ipx routing
interface serial 2 multipoint
 ipx network 1
 frame-relay map ipx 1.000.0c02.5f4f 200 broadcast
```

Unnumbered IP over a Point-to-Point Subinterface Example

The following example sets up unnumbered IP over subinterfaces at both ends of a point-to-point connection. In this example, router A functions as the DTE, and router B functions as the DCE. Routers A and B are both attached to Token Ring networks.

Configuration for Router A

```
interface token-ring 0
 ip address 131.108.177.1 255.255.255.0
!
interface serial 0
 no ip address
 encapsulation frame-relay IETF
!
interface serial0.2 point-to-point
 ip unnumbered TokenRing0
 ip pim sparse-mode
 frame-relay interface-dlci 20
```

Configuration for Router B

```
frame-relay switching
!
interface token-ring 0
 ip address 131.108.178.1 255.255.255.0
!
interface serial 0
 no ip address
 encapsulation frame-relay IETF
 bandwidth 384
 clockrate 4000000
 frame-relay intf-type dce
```

```
!
interface serial 0.2 point-to-point
 ip unnumbered TokenRing1
 ip pim sparse-mode
!
 bandwidth 384
 frame-relay interface-dlci 20
```

Transparent Bridging Using Subinterfaces Example

The following example shows Frame Relay DLCIs 42, 64, and 73 as separate point-to-point links with transparent bridging running over them. The bridging spanning tree views each PVC as a separate bridge port, and a frame arriving on the PVC can be relayed back out on a separate PVC.

```
interface serial 0
 encapsulation frame-relay
interface serial 0.1 point-to-point
 bridge-group 1
 frame-relay interface-dlci 42
interface serial 0.2 point-to-point
 bridge-group 1
 frame-relay interface-dlci 64
interface serial 0.3 point-to-point
 bridge-group 1
 frame-relay interface-dlci 73
```

SVC Configuration Examples

The following sections provide examples of Frame Relay SVC configuration for interfaces and subinterfaces:

- SVC Interface Example
- SVC Subinterface Example

SVC Interface Example

The following example configures a physical interface, applies a map group to the physical interface, and then defines the map group:

```
interface serial 0
 ip address 172.10.8.6
 encapsulation frame-relay
 map-group bermuda
 frame-relay lmi-type q933a
 frame-relay svc
!
map-list bermuda source-addr E164 123456 dest-addr E164 654321
 ip 131.108.177.100 class hawaii
 appletalk 1000.2 class rainbow
!
map-class frame-relay rainbow
 frame-relay idle-timer 60
!
map-class frame-relay hawaii
 frame-relay cir in 64000
 frame-relay cir out 64000
```

SVC Subinterface Example

The following example configures a point-to-point interface for SVC operation. It assumes that the main serial 0 interface has been configured for signalling and that SVC operation has been enabled on the main interface:

```
int s 0.1 point-point
! Define the map-group; details are specified under the map-list holiday command.
map-group holiday
!
! Associate the map-group with a specific source and destination.
map-list holiday local-addr X121 <X121-addr> dest-addr E164 <E164-addr>
! Specify destination protocol addresses for a map-class.
 ip 131.108.177.100 class hawaii IETF
 appletalk 1000.2 class rainbow IETF broadcast
!
! Define a map class and its QoS settings.
map-class hawaii
 frame-relay cir in 2000000
 frame-relay cir out 56000
 frame-relay be 9000
!
! Define another map class and its QoS settings.
map-class rainbow
 frame-relay cir in 64000
 frame-relay idle-timer 2000
```

Frame Relay Traffic Shaping Examples

The following sections provide examples of Frame Relay traffic shaping:

- Traffic Shaping with Three Point-to-Point Subinterfaces Example
- Traffic Shaping with ForeSight Example
- ELMI Configuration Examples

Traffic Shaping with Three Point-to-Point Subinterfaces Example

In the following example, VCs on subinterfaces Serial0.1 and Serial0.2 inherit class parameters from the main interface—namely, those defined in the map class “slow_vcs”—but the VC defined on subinterface Serial0.2 (DLCI 102) is specifically configured to use map class “fast_vcs”.

Map class “slow_vcs” uses a peak rate of 9600 and average rate of 4800 bps. Because BECN feedback is enabled, the output rate will be cut back to as low as 2400 bps in response to received BECNs. This map class is configured to use custom queueing using queue-list 1. In this example, queue-list 1 has 3 queues, with the first two being controlled by access lists 100 and 115.

Map class “fast_vcs” uses a peak rate of 64000 and average rate of 16000 bps. Because BECN feedback is enabled, the output rate will be cut back to as low as 8000 bps in response to received BECNs. This map class is configured to use priority-queueing using priority-group 2.

```
interface serial0
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay traffic-shaping
 frame-relay class slow_vcs
!
interface serial0.1 point-to-point
```

```
 ip address 10.128.30.1 255.255.255.248
 ip ospf cost 200
 bandwidth 10
 frame-relay interface-dlci 101
!
interface serial0.2 point-to-point
 ip address 10.128.30.9 255.255.255.248
 ip ospf cost 400
 bandwidth 10
 frame-relay interface-dlci 102
  class fast_vcs
!
interface serial0.3 point-to-point
 ip address 10.128.30.17 255.255.255.248
 ip ospf cost 200
 bandwidth 10
 frame-relay interface-dlci 103
!
map-class frame-relay slow_vcs
 frame-relay traffic-rate 4800 9600
 frame-relay custom-queue-list 1
 frame-relay adaptive-shaping becn
!
map-class frame-relay fast_vcs
 frame-relay traffic-rate 16000 64000
 frame-relay priority-group 2
 frame-relay adaptive-shaping becn
!
access-list 100 permit tcp any any eq 2065
access-list 115 permit tcp any any eq 256
!
priority-list 2 protocol decnet high
priority-list 2 ip normal
priority-list 2 default medium
!
queue-list 1 protocol ip 1 list 100
queue-list 1 protocol ip 2 list 115
queue-list 1 default 3
queue-list 1 queue 1 byte-count 1600 limit 200
queue-list 1 queue 2 byte-count 600 limit 200
queue-list 1 queue 3 byte-count 500 limit 200
```

Traffic Shaping with ForeSight Example

The following example illustrates a router configuration with traffic shaping enabled. DLCIs 100 and 101 on subinterfaces Serial 13.2 and Serial 13.3 inherit class parameters from the main interface. The traffic shaping for these two VCs will be adaptive to the ForeSight notification.

For Serial 0, the output rate for DLCI 103 will not be affected by the router ForeSight function.

```
interface Serial0
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay traffic-shaping
!
interface Serial0.2 point-to-point
 ip address 10.128.30.17 255.255.255.248
 frame-relay interface-dlci 102
 class fast_vcs
!
interface Serial0.3 point-to-point
 ip address 10.128.30.5 255.255.255.248
```

```
 ip ospf cost 200
 frame-relay interface-dlci 103
 class slow_vcs
!
interface serial 3
 no ip address
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay class fast_vcs
!
interface Serial3.2 multipoint
 ip address 100.120.20.13 255.255.255.248
 frame-relay map ip 100.120.20.6 16 ietf broadcast
!
interface Serial3.3 point-to-point
 ip address 100.120.10.13 255.255.255.248
 frame-relay interface-dlci 101
!
map-class frame-relay slow_vcs
 frame-relay adaptive-shaping becn
 frame-relay traffic-rate 4800 9600
!
map-class frame-relay fast_vcs
 frame-relay adaptive-shaping foresight
 frame-relay traffic-rate 16000 64000
 frame-relay cir 56000
 frame-relay bc 64000
```

ELMI Configuration Examples

The following sections provide ELMI configuration examples:

- ELMI and Frame Relay Traffic Shaping Example
- Configuring the IP Address for ELMI Address Registration Example
- Disabling ELMI Address Registration on an Interface Example

ELMI and Frame Relay Traffic Shaping Example

The following configuration shows a Frame Relay interface enabled with QoS autosense. The router receives messages from the Cisco switch, which is also configured with QoS autosense enabled. When ELMI is configured in conjunction with traffic shaping, the router will receive congestion information through BECN or router ForeSight congestion signalling and reduce its output rate to the value specified in the traffic shaping configuration.

```
interface serial0
  no ip address
  encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay traffic-shaping
  frame-relay QoS-autosense
!
interface serial0.1 point-to-point
  no ip address
  frame-relay interface-dlci 101
```

Configuring the IP Address for ELMI Address Registration Example

The following example shows how to configure the IP address to be used for ELMI address registration. Automatic IP address selection is automatically disabled when the IP address is configured. ELMI is enabled on serial interface 0.

```
interface Serial 0
 no ip address
 encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay qos-autosense
!
frame-relay address registration ip address 139.85.242.195
!
```

Disabling ELMI Address Registration on an Interface Example

In the following example, ELMI address registration is disabled on serial interface 0. This interface will share an IP address of 0.0.0.0 and an ifIndex of 0. Automatic IP address selection is enabled by default when ELMI is enabled, so the management IP address of other interfaces on this router will be chosen automatically.

```
interface Serial 0
 no ip address
 encapsulation frame-relay
  frame-relay lmi-type ansi
  frame-relay qos-autosense
  no frame-relay address-reg-enable
!
```

Backward Compatibility Example

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
```

Booting from a Network Server over Frame Relay Example

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.

Frame Relay Switching Examples

The following sections provide examples of configuring one or more routers as Frame Relay switches:

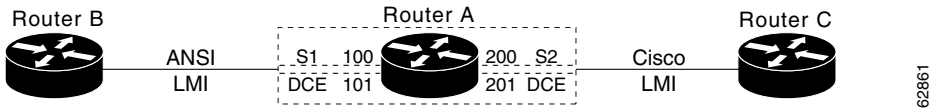
- PVC Switching Configuration Example
- Pure Frame Relay DCE Example
- Hybrid DTE/DCE PVC Switching Example
- Switching over an IP Tunnel Example
- Frame Relay Switching over ISDN B Channels Example
- Traffic Shaping on Switched PVCs Example
- Traffic Policing on a UNI DCE Example
- Congestion Management on Switched PVCs Example
- Congestion Management on the Traffic-Shaping Queue of a Switched PVC Example
- FRF.12 Fragmentation on a Switched PVC Configuration Example

PVC Switching Configuration Example

You can configure your router as a dedicated, DCE-only Frame Relay switch. Switching is based on DLCIs. The incoming DLCI is examined, and the outgoing interface and DLCI are determined. Switching takes place when the incoming DLCI in the packet is replaced by the outgoing DLCI, and the packet is sent out the outgoing interface.

In Figure 10, the router switches two PVCs between serial interfaces 1 and 2. Frames with DLCI 100 received on serial 1 will be transmitted with DLCI 200 on serial 2.

Figure 10 PVC Switching Configuration



The following example shows one router with two interfaces configured as DCEs. The router switches frames from the incoming interface to the outgoing interface on the basis of the DLCI alone.

Configuration for Router A

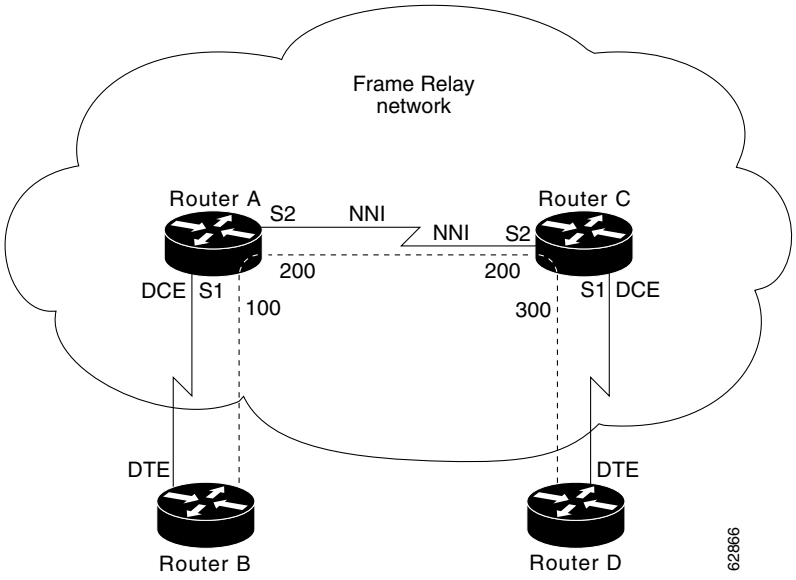
```
frame-relay switching

interface Serial1
 no ip address
 encapsulation frame-relay
 keepalive 15
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 100 interface Serial2 200
 frame-relay route 101 interface Serial2 201
 clockrate 2000000
!
interface Serial2
 encapsulation frame-relay
 keepalive 15
 frame-relay intf-type dce
 frame-relay route 200 interface Serial1 100
 frame-relay route 201 interface Serial1 101
 clockrate 64000
```

Pure Frame Relay DCE Example

Using the PVC switching feature, it is possible to build an entire Frame Relay network using routers. In Figure 11, router A and router C act as Frame Relay switches implementing a two-node network. The standard Network-to-Network Interface (NNI) signalling protocol is used between router A and router C.

The following example shows a Frame Relay network with two routers functioning as switches and standard NNI signalling used between them.

Figure 11 Frame Relay DCE Configuration



Configuration for Router A

```
frame-relay switching
!
interface serial 1
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay lmi-type ansi
 frame-relay route 100 interface serial 2 200
!
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay intf-type nni
 frame-relay lmi-type q933a
 frame-relay route 200 interface serial 1 100
 clockrate 2048000
!
```

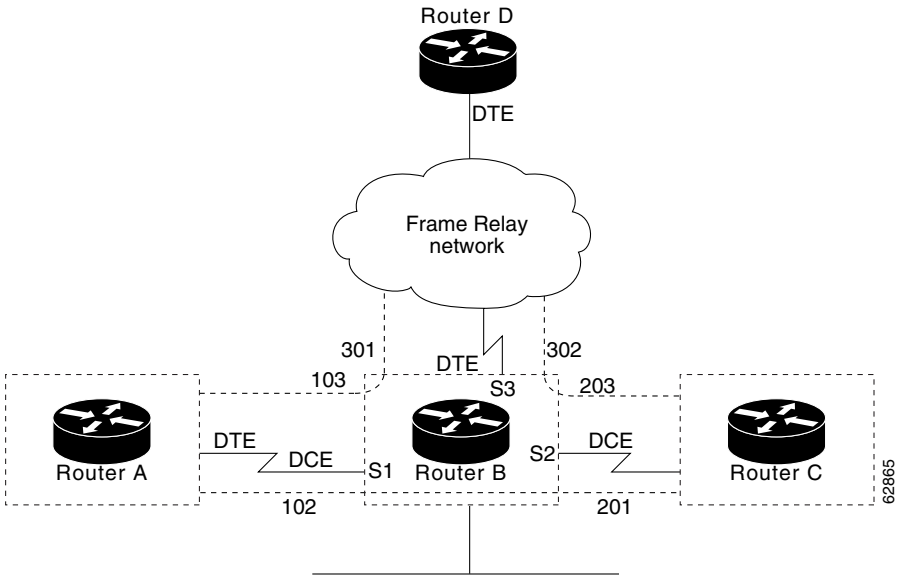
Configuration for Router C

```
frame-relay switching
!
interface serial 1
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 300 interface serial 2 200
!
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay intf-type nni
 frame-relay lmi-type q933a
 frame-relay route 200 interface serial 1 300
!
```

Hybrid DTE/DCE PVC Switching Example

Routers can be configured as hybrid DTE/DCE Frame Relay switches, as shown in Figure 12.

Figure 12 Hybrid DTE/DCE PVC Switching



The following example shows one router configured with both DCE and DTE interfaces (router B acts as a hybrid DTE/DCE Frame Relay switch). It can switch frames between two DCE ports and between a DCE port and a DTE port. Traffic from the Frame Relay network can also be terminated locally. In the example, three PVCs are defined as follows:

- Serial 1, DLCI 102, to serial 2, DLCI 201—DCE switching
- Serial 1, DLCI 103, to serial 3, DLCI 301—DCE/DTE switching
- Serial 2, DLCI 203, to serial 3, DLCI 302—DCE/DTE switching

DLCI 400 is also defined for locally terminated traffic.

Configuration for Router B

```
frame-relay switching
!
interface ethernet 0
 ip address 131.108.123.231 255.255.255.0
!
interface ethernet 1
 ip address 131.108.5.231 255.255.255.0
!
interface serial 0
 no ip address
 shutdown :Interfaces not in use may be shut down; shut down is not required.
!
interface serial 1
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 102 interface serial 2 201
 frame-relay route 103 interface serial 3 301
!
```

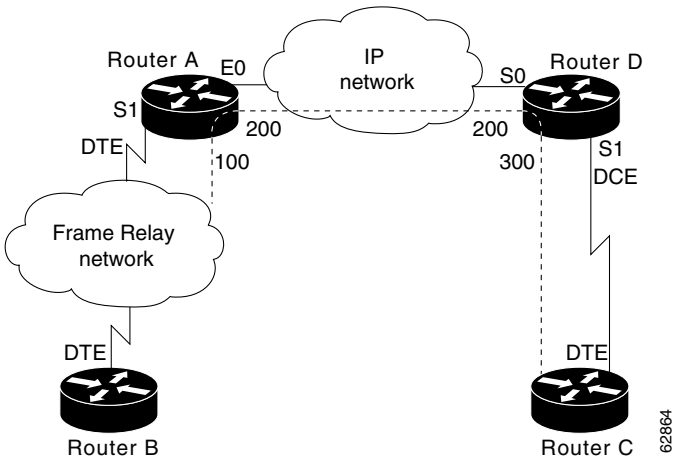
```
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 201 interface serial 1 102
 frame-relay route 203 interface serial 3 302
!
interface serial 3
 ip address 131.108.111.231
 encapsulation frame-relay
 frame-relay lmi-type ansi
 frame-relay route 301 interface serial 1 103
 frame-relay route 302 interface serial 1 203
 frame-relay map ip 131.108.111.4 400 broadcast
```

Switching over an IP Tunnel Example

You can achieve switching over an IP tunnel by creating a point-to-point tunnel across the internetwork over which PVC switching can take place, as shown in Figure 13.

Note Static routes cannot be configured over tunnel interfaces on the Cisco 800 series, 1600 series, and 1700 series platforms. Static routes can only be configured over tunnel interfaces on platforms that have the Enterprise feature set.

Figure 13 Frame Relay Switch over IP Tunnel



The following example shows two routers configured to switch Frame Relay PVCs over a point-to-point IP tunnel, which is the IP network configuration depicted in Figure 13.

Configuration for Router A

```
frame-relay switching
!
interface ethernet0
 ip address 108.131.123.231 255.255.255.0
!
interface ethernet1
 ip address 131.108.5.231 255.255.255.0
!
```

```
interface serial0
 no ip address
 shutdown : Interfaces not in use may be shut down; shutdown is not required.
!
interface serial1
 ip address 131.108.222.231 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 131.108.222.4 400 broadcast
 frame-relay route 100 interface Tunnel1 200
!
interface tunnel1
 tunnel source Ethernet0
 tunnel destination 150.150.150.123
```

Configuration for Router D

```
frame-relay switching
!
interface ethernet0
 ip address 131.108.231.123 255.255.255.0
!
interface ethernet1
 ip address 131.108.6.123 255.255.255.0
!
interface serial0
 ip address 150.150.150.123 255.255.255.0
 encapsulation ppp
!
interface tunnel1
 tunnel source Serial0
 tunnel destination 108.131.123.231
!
interface serial1
 ip address 131.108.7.123 255.255.255.0
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 300 interface Tunnel1 200
```

Frame Relay Switching over ISDN B Channels Example

The following example illustrates Frame Relay switching over an ISDN dialer interface:

```
frame-relay switching
  !
  interface BRI0
    isdn switch-type basic-5ess
    dialer pool-member 1
    dialer pool-member 2
  !
  interface dialer1
    encapsulation frame-relay
    dialer pool 1
    dialer-group 1
    dialer caller 60038
    dialer string 60038
    frame-relay intf-type dce
  !
  interface dialer2
    encapsulation frame-relay
    dialer pool 2
    dialer-group 1
    dialer caller 60039
    dialer string 60039
```

```
  frame-relay intf-type dce
!
interface serial0
  encapsulation frame-relay
  frame-relay intf-type dce
!
connect one serial0 16 dialer1 100
connect two serial0 17 dialer2 100
dialer-list 1 protocol ip permit
```

Note Note that when Frame Relay switching is performed by using a dialer profile, encapsulation of the underlying physical (BRI) interface must be configured as high-level data link control (HDLC).

Traffic Shaping on Switched PVCs Example

In the example that follows, traffic on serial interface 0 is being shaped prior to entry to the Frame Relay network. PVC 100/16 is shaped according to the "shape256K" class. PVC 200/17 is shaped using the "shape64K" class inherited from the interface.

```
frame-relay switching
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
    frame-relay traffic-shaping
    frame-relay class shape64K
    frame-relay interface-dlci 16 switched
      class shape256K
  !
  interface serial1
    encapsulation frame-relay
    frame-relay intf-type dce
  !
  connect one serial0 16 serial1 100
  connect two serial0 17 serial1 200
  !
  map-class frame-relay shape256K
    frame-relay traffic-rate 256000 512000
  !
  map-class frame-relay shape64K
    frame-relay traffic-rate 64000 64000
```

Traffic Policing on a UNI DCE Example

In the following example, incoming traffic is being policed on serial interface 1. The interface uses policing parameters configured in map class "police256K". PVC 100/16 inherits policing parameters from the interface. PVC 200/17 uses policing parameters configured in "police64K".

```
frame-relay switching
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
  !
  interface serial1
    encapsulation frame-relay
    frame-relay policing
    frame-relay class police256K
```

```
  frame-relay intf-type dce
  frame-relay interface-dlci 200 switched
    class police64K
!
connect one serial0 16 serial1 100
connect two serial0 17 serial1 200
!
map-class frame-relay police256K
  frame-relay cir 256000
  frame-relay bc 256000
  frame-relay be 0
!
map-class frame-relay police64K
  frame-relay cir 64000
  frame-relay bc 64000
  frame-relay be 64000
```

Congestion Management on Switched PVCs Example

The following example illustrates the configuration of congestion management and DE discard levels for all switched PVCs on serial interface 1. Policing is configured on PVC 16.

```
frame-relay switching
  !
  interface serial0
    encapsulation frame-relay
    frame-relay intf-type dce
    frame-relay policing
    frame-relay interface-dlci 16 switched
      class 256K
  !
  interface serial1
    encapsulation frame-relay
    frame-relay intf-type dce
    frame-relay congestion-management
      threshold ecn be 0
      threshold ecn bc 20
      threshold de 40
  !
  connect one serial1 100 serial0 16
  !
  map-class frame-relay 256K
    frame-relay cir 256000
    frame-relay bc 256000
    frame-relay be 256000
```

Congestion Management on the Traffic-Shaping Queue of a Switched PVC Example

The following example illustrates the configuration of congestion management in a class called "perpvc_congestion". The class is associated with the traffic-shaping queue of DLCI 200 on serial interface 3.

```
  map-class frame-relay perpvc_congestion
    frame-relay holdq 100
    frame-relay congestion threshold ecn 50

  interface Serial3
    frame-relay traffic-shaping
    frame-relay interface-dlci 200 switched
      class perpvc_congestion
```

FRF.12 Fragmentation on a Switched PVC Configuration Example

In the following example, FRF.12 fragmentation is configured in a map class called “data”. The “data” map class is assigned to switched pvc 20 on serial interface 3/3.

```
frame-relay switching
!
interface Serial3/2
 encapsulation frame-relay
 frame-relay intf-type dce
!
interface Serial3/3
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay interface-dlci 20 switched
  class data
 frame-relay intf-type dce
!
map-class frame-relay data
 frame-relay fragment 80 switched
 frame-relay cir 64000
 frame-relay bc 640
!
connect data Serial3/2 16 Serial3/3 20
```

Frame Relay End-to-End Keepalive Examples

The following sections provide examples of Frame Relay end-to-end keepalive in different modes and configurations:

- End-to-End Keepalive Bidirectional Mode with Default Configuration Example
- End-to-End Keepalive Request Mode with Default Configuration Example
- End-to-End Keepalive Request Mode with Modified Configuration Example

End-to-End Keepalive Bidirectional Mode with Default Configuration Example

In the following example, the devices at each end of a VC are configured so that a DLCI is assigned to a Frame Relay serial interface, a map class is associated with the interface, and Frame Relay end-to-end keepalive is configured in bidirectional mode using default values:

```
! router1
router1(config) interface serial 0/0.1 point-to-point
router1(config-if) ip address 10.1.1.1 255.255.255.0
router1(config-if) frame-relay interface-dlci 16
router1(config-if) frame-relay class vcgrp1
router1(config-if) exit
!
router1(config)# map-class frame-relay vcgrp1
router1(config-map-class)# frame-relay end-to-end keepalive mode bidirectional
! router2
router2(config) interface serial 1/1.1 point-to-point
router2(config-if) ip address 10.1.1.2 255.255.255.0
router2(config-if) frame-relay interface-dlci 16
router2(config-if) frame-relay class vceek
router1(config-if) exit
!
router2(config)# map-class frame-relay vceek
```

```
router2(config-map-class)# frame-relay end-to-end keepalive mode bidirectional
```

End-to-End Keepalive Request Mode with Default Configuration Example

In the following example, the devices at each end of a VC are configured so that a DLCI is assigned to a Frame Relay serial interface and a map class is associated with the interface. One device is configured in request mode while the other end of the VC is configured in reply mode.

```
! router1
router1(config) interface serial 0/0.1 point-to-point
router1(config-if) ip address 10.1.1.1 255.255.255.0
router1(config-if) frame-relay interface-dlci 16
router1(config-if) frame-relay class eek
router1(config-if) exit
!
router1(config)# map-class frame-relay eek
router1(config-map-class)# frame-relay end-to-end keepalive mode request

! router2
router2(config) interface serial 1/1.1 point-to-point
router2(config-if) ip address 10.1.1.2 255.255.255.0
router2(config-if) frame-relay interface-dlci 16
router2(config-if) frame-relay class group_3
router1(config-if) exit
!
router2(config)# map-class frame-relay group_3
router2(config-map-class)# frame-relay end-to-end keepalive mode reply
```

End-to-End Keepalive Request Mode with Modified Configuration Example

In the following example, the devices at each end of a VC are configured so that a DLCI is assigned to a Frame Relay serial interface and a map class is associated with the interface. One device is configured in request mode while the other end of the VC is configured in reply mode. The event window, error threshold, and success events values are changed so that the interface will change state less frequently:

```
! router1
router1(config) interface serial 0/0.1 point-to-point
router1(config-if) ip address 10.1.1.1 255.255.255.0
router1(config-if) frame-relay interface-dlci 16
router1(config-if) frame-relay class eek
router1(config-if) exit
!
router1(config)# map-class frame-relay eek
router1(config-map-class)# frame-relay end-to-end keepalive mode request
router1(config-map-class)# frame-relay end-to-end keepalive event-window send 5
router1(config-map-class)# frame-relay end-to-end keepalive error-threshold send 3
router1(config-map-class)# frame-relay end-to-end keepalive success-events send 3

! router2
router2(config) interface serial 1/1.1 point-to-point
router2(config-if) ip address 10.1.1.2 255.255.255.0
router2(config-if) frame-relay interface-dlci 16
router2(config-if) frame-relay class group_3
router1(config-if) exit
!
router2(config)# map-class frame-relay group_3
router2(config-map-class)# frame-relay end-to-end keepalive mode reply
```

PPP over Frame Relay Examples

The following sections provide examples of PPP over Frame Relay from the DTE and DCE end of the network:

- PPP over Frame Relay DTE Example
- PPP over Frame Relay DCE Example

PPP over Frame Relay DTE Example

The following example configures a router as a DTE device for PPP over Frame Relay. Subinterface 2.1 contains the necessary DLCI and virtual template information. Interface Virtual-Template 1 contains the PPP information that is applied to the PPP session associated with DLCI 32 on serial subinterface 2.1.

```
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface serial 2.1 point-to-point
 frame-relay interface-dlci 32 ppp virtual-template1
!
interface Virtual-Template1
 ip unnumbered ethernet 0
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulation; therefore, **encapsulation ppp** will not appear when you view the configuration of the router.

PPP over Frame Relay DCE Example

The following example configures a router to act as a DCE device. Typically, a router is configured as a DCE if it is connecting directly to another router or if connected to a 90i D4 channel unit, which is connected to a telco channel bank. The three commands required for this type of configuration are the **frame-relay switching, frame-relay intf-type dce,** and **frame-relay route** commands:

```
frame-relay switching
!
interface Serial2/0:0
 no ip address
 encapsulation frame-relay IETF
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 31 interface Serial1/2 100
 frame-relay interface-dlci 32 ppp Virtual-Template1
!
interface Serial2/0:0.2 point-to-point
 no ip address
 frame-relay interface-dlci 40 ppp Virtual-Template2
!
interface Virtual-Template1
 ip unnumbered Ethernet0/0
 peer default ip address pool default
 ppp authentication chap pap
 !
interface Virtual-Template2
 ip address 100.1.1.2 255.255.255.0
```

```
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulation; therefore, **encapsulation ppp** will not appear when you view the configuration of the router.

Frame Relay Fragmentation Configuration Examples

The following sections provide examples of Frame Relay fragmentation configuration:

- FRF.12 Fragmentation Example
- Frame Relay Fragmentation with Hardware Compression Example

FRF.12 Fragmentation Example

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 traffic-shaping
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 cir 128000
router(config-map-class)# frame-relay bc 1280
router(config-map-class)# frame-relay fragment 40
router(config-map-class)# frame-relay fair-queue
```

Frame Relay Fragmentation with Hardware Compression Example

In the following example, FRF.12 fragmentation and FRF.9 hardware compression are configured on multipoint interface 3/1 and point-to-point interface 3/1.1:

```
interface serial3/1
 ip address 10.1.0.1 255.255.255.0
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay class frag
 frame-relay map ip 10.1.0.2 110 broadcast ietf payload-compression frf9 stac
!
interface serial3/1.1 point-to-point
 ip address 10.2.0.1 255.255.255.0
 frame-relay interface-dlci 120 ietf
 frame-relay payload-compression frf9 stac
!
map-class frame-relay frag
 frame-relay cir 64000
 frame-relay bc 640
 frame-relay fragment 100
```

Payload Compression Configuration Examples

The following sections provide examples of various methods of configuring payload compression:

- FRF.9 Compression for Subinterfaces Using the frame-relay map Command Example
- FRF.9 Compression for Subinterfaces Example
- Data-Stream Hardware Compression with TCP/IP Header Compression on a Point-to-Point Subinterface Example
- Data-Stream Hardware Compression with TCP/IP Header Compression on a Multipoint Subinterface Example
- Data-Stream Hardware Compression with RTP Header Compression and Frame Relay Fragmentation Example

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.

FRF.9 Compression for Subinterfaces Using the frame-relay map Command Example

The following example shows a subinterface being configured for FRF.9 compression using the **frame-relay map** command:

```
interface serial2/0/1
 ip address 172.16.1.4 255.255.255.0
 no ip route-cache
 encapsulation frame-relay IETF
 no keepalive
 frame-relay map ip 172.16.1.1 105 IETF payload-compression FRF9 stac
```

FRF.9 Compression for Subinterfaces Example

The following example shows a subinterface being configured for FRF.9 compression:

```
interface serial2/0/0
 no ip address
 no ip route-cache
 encapsulation frame-relay
 ip route-cache distributed
 no keepalive
!
interface serial2/0/0.500 point-to-point
 ip address 172.16.1.4 255.255.255.0
 no cdp enable
 frame-relay interface-dlci 500 IETF
 frame-relay payload-compression FRF9 stac
```

Data-Stream Hardware Compression with TCP/IP Header Compression on a Point-to-Point Subinterface Example

The following example shows the configuration of data-stream hardware compression and TCP header compression on point-to-point interface 1/0.1:

```
interface serial1/0
  encapsulation frame-relay
  frame-relay traffic-shaping
 !
 interface serial1/0.1 point-to-point
  ip address 10.0.0.1 255.0.0.0
  frame-relay interface-dlci 100
```

```
 frame-relay payload-compression data-stream stac
 frame-relay ip tcp header-compression
```

Data-Stream Hardware Compression with TCP/IP Header Compression on a Multipoint Subinterface Example

The following example shows the configuration of data-stream hardware compression and TCP header compression on multipoint interface 3/1:

```
interface serial3/1
 ip address 10.1.0.1 255.255.255.0
 encapsulation frame-relay
 frame-relay traffic-shaping
 frame-relay map ip 10.1.0.2 110 broadcast cisco payload-compression data-stream stac
 frame-relay ip tcp header-compression
```

Data-Stream Hardware Compression with RTP Header Compression and Frame Relay Fragmentation Example

The following example shows the configuration of data-stream hardware compression, RTP header compression, and FRF.12 fragmentation on point-to-point interface 1/0.1:

```
interface serial1/0
  encapsulation frame-relay
  frame-relay traffic-shaping
 !
 interface serial1/0.1 point-to-point
  ip address 10.0.0.1 255.0.0.0
  frame-relay interface-dlci 100
  frame-relay class frag
  frame-relay payload-compression data-stream stac
  frame-relay ip rtp header-compression
 !
 map-class frame-relay frag
  frame-relay cir 64000
  frame-relay bc 640
  frame-relay be 0
  frame-relay fragment 100
  frame-relay ip rtp priority 16000 16000 20
```

TCP/IP Header Compression Examples

The following sections provide examples of configuring various combinations of TCP/IP header compression, encapsulation characteristics on the interface, and the effect on the inheritance of those characteristics on a Frame Relay IP map:

- IP Map with Inherited TCP/IP Header Compression Example
- Using an IP Map to Override TCP/IP Header Compression Example
- Disabling Inherited TCP/IP Header Compression Example
- Disabling Explicit TCP/IP Header Compression Example

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.

IP Map with Inherited TCP/IP Header Compression Example

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.

Using an IP Map to Override TCP/IP Header Compression Example

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
```

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.

Disabling Inherited TCP/IP Header Compression Example

In this example, the 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.

Disabling Explicit TCP/IP Header Compression Example

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.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Frame Relay 64-Bit Counters

Feature History

Release	Modification
12.0(17)S	This feature was introduced on the Cisco 12000 series.
12.2(4)T	This feature was integrated into Cisco IOS Release 12.2(4)T.
12.2(4)T3	Support for the Cisco 7500 series routers was added.
12.0(21)S	The **frame-relay ifmib-counter64** command was introduced.
12.3(10)	The **frame-relay ifmib-counter64** command was integrated into Cisco IOS Release 12.3(10).
12.3(11)T	The **frame-relay ifmib-counter64** command was integrated into Cisco IOS Release 12.3(11)T.
12.2(18)SXE	The **frame-relay ifmib-counter64** command was integrated into Cisco IOS Release 12.2(18)SXE.

This document describes the Frame Relay 64-Bit Counters feature in Cisco IOS Release 12.2(4)T. It includes the following sections:

- Feature Overview, page 2
- Supported Platforms, page 2
- Supported Standards, MIBs, and RFCs, page 3
- Prerequisites, page 3
- Configuration Tasks, page 3
- Monitoring and Maintaining Frame Relay 64-Bit Counters, page 4
- Configuration Examples, page 4
- Command Reference, page 8

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

Feature Overview

The Frame Relay 64-Bit Counters feature provides 64-bit counter support on Frame Relay interfaces and subinterfaces. This feature enables the gathering of statistics through Simple Network Management Protocol (SNMP) for faster interfaces operating at OC-3, OC-12, and OC-48 speeds.

The following counters are supported by this feature: Bytes In, Bytes Out, Packets In, and Packets Out.

The **show frame-relay pvc** command has been modified to display the 64-bit counters.

Benefits

The values in 32-bit counters sometime wrap because the field is too small. Wrapping causes the values in these fields to become meaningless. The 64-bit counters support the reliable gathering of statistics by SNMP by preventing the wrapping of counter values.

Restrictions

SNMP cannot retrieve 64-bit virtual-circuit (VC) counters.

Related Documents

For information on configuring Frame Relay using Cisco IOS software, refer to the following documents:

- The chapter "*Configuring Frame Relay*" in the *Cisco IOS Wide-Area Networking Configuration Guide*, Release 12.2
- The chapter "*Frame Relay Commands*" in the *Cisco IOS Wide-Area Networking Command Reference*, Release 12.2

For information on configuring SNMP using Cisco IOS software, see the following documents:

- The chapter "*Configuring Simple Network Management Protocol*" in the *Cisco IOS Configuration Fundamentals Configuration Guide*, Release 12.2
- The chapter "*SNMP Commands*" in the *Cisco IOS Configuration Fundamentals Command Reference*, Release 12.2

Supported Platforms

- Cisco 7200 series
- Cisco 7500 series (Cisco IOS Release 12.2(4)T3 and later)

Platform Support Through Feature Navigator

Cisco IOS software is packaged in feature sets that support specific platforms. To get updated information regarding platform support for this feature, access Feature Navigator. Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Feature Navigator is a web-based tool that enables you to quickly determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image.

To access Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions at http://www.cisco.com/register.

Feature Navigator is updated when major Cisco IOS software releases and technology releases occur. As of May 2001, Feature Navigator supports M, T, E, S, and ST releases. You can access Feature Navigator at the following URL:

http://www.cisco.com/go/fn

Supported Standards, MIBs, and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

The **frame-relay ifmib-counter64** command modifies the interface MIB (IF-MIB) by allowing slower Frame Relay interfaces and subinterfaces to be included in the 64-bit interface MIB counters.

To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

This document assumes that you know how to configure Frame Relay and SNMP support using Cisco IOS software.

Configuration Tasks

This section contains the following task:

- Enabling Frame Relay Interfaces to Be Included in 64-Bit Interface MIB Counters

Enabling Frame Relay Interfaces to Be Included in 64-Bit Interface MIB Counters

Note This task is supported in Cisco IOS releases 12.0(21)S, 12.3(10), 12.3(11)T, 12.2(18)SXE, and later releases.

Frame Relay interfaces and subinterfaces that have a line speed greater than 20 Mbps are included in the 64-bit interface MIB counters by default. Perform this task to enable Frame Relay interfaces and subinterfaces that have a line speed of less than 20 Mbps to be included in the 64-bit interface MIB counters.

	Command	Purpose
Step 1	`Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	`Router# configure terminal`	Enters global configuration mode.
Step 3	`Router(config)# interface serial interface-number`	Specifies an interface to be configured and enters interface configuration mode.
Step 4	`Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
Step 5	`Router(config-if)# frame-relay ifmib-counter64 [if | subif]`	Enables Frame Relay interfaces and subinterfaces that have a line speed of less than 20 Mbps to be included in 64-bit interface MIB counters. • This command allows Frame Relay interfaces and subinterfaces that have a line speed of less than 20 Mbps to be included in the following 64-bit interface MIB counters: – ifHCinOctets – ifHCOutOctets – ifHCinUcastPkts – ifHCOutUcastPkts

Monitoring and Maintaining Frame Relay 64-Bit Counters

To view the values of the Frame Relay 64-bit counters, use the following command in EXEC mode:

Command	Purpose
`Router# show frame-relay pvc 64-bit [interface interface] [dlci]`	Displays statistics about permanent virtual circuits (PVCs) for Frame Relay interfaces.

Configuration Examples

This section contains the following example:

- Enabling Slower Frame Relay Interfaces and Subinterfaces to Be Included in 64-Bit Interface MIB Counters: Example
- Enabling Only Slower Frame Relay Subinterfaces to Be Included in 64-Bit Interface MIB Counters: Example

Enabling Slower Frame Relay Interfaces and Subinterfaces to Be Included in 64-Bit Interface MIB Counters: Example

In the following example, the **frame-relay ifmib-counter64** command is used with the **if** keyword to enable serial interfaces 6/0/1:0, 6/0/2:0, and 6/0/3:0 and related subinterfaces to be included in the 64-bit interface MIB counters. The example also shows corresponding output for the **show frame-relay pvc** command and the corresponding statistics for the 64-bit interface MIB counters.

```
interface Serial6/0/1:0
 ip address 1.1.1.1 255.255.255.0
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 frame-relay interface-dlci 101
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 if

interface Serial6/0/2:0
 no ip address
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 if
!
interface Serial6/0/2:0.1 point-to-point
 ip address 2.1.1.1 255.255.255.0
 no ip route-cache
 frame-relay interface-dlci 201
!
interface Serial6/0/3:0
 ip address 3.1.1.1 255.255.255.0
 encapsulation frame-relay
 frame-relay interface-dlci 301
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 if

interface Serial6/0/3:0.1 point-to-point
 ip address 3.1.2.1 255.255.255.0
 frame-relay interface-dlci 302
```

The following example shows corresponding sample output for the **show frame-relay pvc** command with the **64-bit** keyword. Note that the **frame-relay ifmib-counter64** command does not affect the output of the **show frame-relay pvc** command.

```
Router# show frame-relay pvc 101 64-bit

DLCI = 101, INTERFACE = Serial6/0/1:0
  input pkts 231                    output pkts 228
  in bytes 23604                    out bytes 23502
Router#
Router# show frame-relay pvc 201 64-bit

DLCI = 201, INTERFACE = Serial6/0/2:0.1
  input pkts 1453                   output pkts 1408
  in bytes 335024                   out bytes 327272
Router#
Router# show frame-relay pvc 301 64-bit

DLCI = 301, INTERFACE = Serial6/0/3:0
  input pkts 510                    output pkts 508
```

```
  in bytes 52690                    out bytes 52622
Router#
Router# show frame-relay pvc 302 64-bit

DLCI = 302, INTERFACE = Serial6/0/3:0.1
  input pkts 957                    output pkts 912
  in bytes 283246                   out bytes 275493
Router#
```

The following output from an SNMP inquiry shows that the 64-bit interface MIB counters include the interfaces configured above:

```
ifHCInOctets.5 = 0x000000000
ifHCInOctets.16 = 0x000000000
ifHCInOctets.17 = 0x003360d33
ifHCInOctets.18 = 0x000000000
ifHCInOctets.19 = 0x000000000
ifHCInOctets.20 = 0x000000000
ifHCInOctets.24 = 0x000000000
ifHCInOctets.25 = 0x000000000
ifHCInOctets.26 = 0x0001a7afc !! This is serial interface 6/0/1:0
ifHCInOctets.28 = 0x0001a7370 !! This is serial interface 6/0/2:0
ifHCInOctets.34 = 0x00006a45a !! This is serial interface 6/0/3:0
ifHCInOctets.36 = 0x000051cb0 !! This is serial subinterface 6/0/2:0.1
ifHCInOctets.37 = 0x00004526e !! This is serial subinterface 6/0/3:0.1
```

Enabling Only Slower Frame Relay Subinterfaces to Be Included in 64-Bit Interface MIB Counters: Example

In the following example, the **frame-relay ifmib-counter64** command is used with the **subif** keyword to enable subinterfaces that are associated with serial interfaces 6/0/1:0, 6/0/2:0, and 6/0/3:0 to be included in the 64-bit interface MIB counters. Slower main interfaces are not included. The example also shows the corresponding statistics for the 64-bit interface MIB counters.

```
interface Serial6/0/1:0
 ip address 1.1.1.1 255.255.255.0
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 frame-relay interface-dlci 101
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 subif

interface Serial6/0/2:0
 no ip address
 encapsulation frame-relay
 no ip route-cache cef
 no ip route-cache
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 subif

interface Serial6/0/2:0.1 point-to-point
 ip address 2.1.1.1 255.255.255.0
 no ip route-cache
 frame-relay interface-dlci 201
!
interface Serial6/0/3:0
 ip address 3.1.1.1 255.255.255.0
 encapsulation frame-relay
 frame-relay interface-dlci 301
```

```
 no frame-relay inverse-arp
 frame-relay ifmib-counter64 subif

interface Serial6/0/3:0.1 point-to-point
 ip address 3.1.2.1 255.255.255.0
 frame-relay interface-dlci 302
```

The following example shows corresponding sample output for the **show frame-relay pvc** command with the **64-bit** keyword. Note that the **frame-relay ifmib-counter64** command does not affect the output of the **show frame-relay pvc** command.

```
Router# show frame-relay pvc 101 64-bit

DLCI = 101, INTERFACE = Serial6/0/1:0
  input pkts 231                    output pkts 228
  in bytes 23604                    out bytes 23502
Router#
Router# show frame-relay pvc 201 64-bit

DLCI = 201, INTERFACE = Serial6/0/2:0.1
  input pkts 1453                   output pkts 1408
  in bytes 335024                   out bytes 327272
Router#
Router# show frame-relay pvc 301 64-bit

DLCI = 301, INTERFACE = Serial6/0/3:0
  input pkts 510                    output pkts 508
  in bytes 52690                    out bytes 52622
Router#
Router# show frame-relay pvc 302 64-bit

DLCI = 302, INTERFACE = Serial6/0/3:0.1
  input pkts 957                    output pkts 912
  in bytes 283246                   out bytes 275493
```

The following output from an SNMP inquiry shows that the 64-bit interface MIB counters include the subinterfaces configured above:

```
ifHCInOctets.5 = 0x000000000
ifHCInOctets.16 = 0x000000000
ifHCInOctets.17 = 0x00337a158
ifHCInOctets.18 = 0x000000000
ifHCInOctets.19 = 0x000000000
ifHCInOctets.20 = 0x000000000
ifHCInOctets.24 = 0x000000000
ifHCInOctets.25 = 0x000000000
ifHCInOctets.36 = 0x000051cb0 !! This is serial subinterface 6/0/2:0.1
ifHCInOctets.37 = 0x00004526e !! This is serial subinterface 6/0/3:0.1
```

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **frame-relay ifmib-counter64**
- **show frame-relay pvc**

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Frame Relay Queueing and Fragmentation at the Interface

Feature History

Release	Modification
12.2(11)S	This feature was introduced.
12.2(13)T	This feature was integrated into Cisco IOS Release 12.2(13)T

This document describes the Frame Relay Queueing and Fragmentation at the Interface feature. This document includes the following sections:

- Feature Overview, page 1
- Supported Platforms, page 3
- Supported Standards, MIBs, and RFCs, page 4
- Prerequisites, page 5
- Configuration Tasks, page 5
- Monitoring and Maintaining Frame Relay Queueing and Fragmentation at the Interface, page 10
- Configuration Examples, page 10
- Command Reference, page 12

Feature Overview

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. This new 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.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

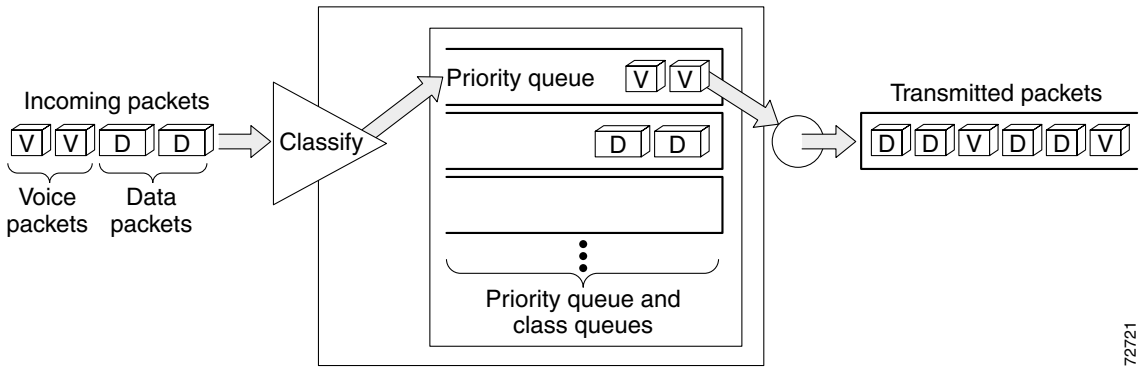
© 2007 Cisco Systems, Inc. All rights reserved.

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. Figure 1 illustrates the interface queueing and fragmentation process.

Figure 1 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

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.

Restrictions

- 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.

Related Documents

For more information about shaping and low-latency queueing for Frame Relay, refer to the following documents:

- *Cisco IOS Quality of Service Solutions Configuration Guide*, Release 12.2
- *Cisco IOS Quality of Service Solutions Command Reference*, Release 12.2
- *Low Latency Queueing for Frame Relay*, Cisco IOS Release 12.1(2)T feature module

For more information about Frame Relay fragmentation, refer to the following documents:

- *Cisco IOS Wide-Area Networking Configuration Guide*, Release 12.2
- *Cisco IOS Wide-Area Networking Command Reference*, Release 12.2

Supported Platforms

- Cisco 800 series
- Cisco 1400 series
- Cisco 1600 series

- Cisco 2500 series
- Cisco 2600 series
- Cisco 3600 series
- Cisco 7100 series
- Cisco 7200 series
- Cisco 7500 series
- Cisco AS5300 series
- Cisco AS5400
- Cisco AS5800
- Cisco MC3810
- Cisco ubr7200 series

Determining Platform Support Through Cisco Feature Navigator

Cisco IOS software is packaged in feature sets that support specific platforms. To get updated information regarding platform support for this feature, access Cisco Feature Navigator. Cisco Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Cisco Feature Navigator is a web-based tool that enables you to determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image. You can search by feature or release. Under the release section, you can compare releases side by side to display both the features unique to each software release and the features in common.

To access Cisco Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions at http://www.cisco.com/register.

Cisco Feature Navigator is updated regularly when major Cisco IOS software releases and technology releases occur. For the most current information, go to the Cisco Feature Navigator home page at the following URL:

http://www.cisco.com/go/fn

Availability of Cisco IOS Software Images

Platform support for particular Cisco IOS software releases is dependent on the availability of the software images for those platforms. Software images for some platforms may be deferred, delayed, or changed without prior notice. For updated information about platform support and availability of software images for each Cisco IOS software release, refer to the online release notes or, if supported, Cisco Feature Navigator.

Supported Standards, MIBs, and RFCs

Standards

FRF.12, *Frame Relay Fragmentation Implementation Agreement*, December 1997

MIBs

No new or modified MIBs are supported by this feature.

To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

The tasks in this document assume that you know how to configure low-latency queueing and shaping service policies.

The following prerequisites are specific to the Cisco 7500 series:

- The Frame Relay Queueing and Fragmentation at the Interface feature is supported on VIP-based interfaces with VIP2-50 or higher.
- Distributed Cisco Express Forwarding (dCEF) must be enabled both globally and on the Frame Relay interface.

Configuration Tasks

See the following sections for configuration tasks for the Frame Relay Queueing and Fragmentation at the Interface feature. Each task in the list is identified as either required or optional.

- Configuring Class Policy for the Priority Queue (required)
- Configuring Class Policy for the Bandwidth Queues (optional)
- Configuring the Shaping Policy Using the Class-Default Class (optional)
- Configuring Queueing and Fragmentation on the Frame Relay Interface (required)
- Verifying Frame Relay Queueing and Fragmentation at the Interface (optional)

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:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	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 2	`Router(config-pmap)# class class-name`	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 3	`Router(config-pmap-c)# priority bandwidth-kbps`	Creates a strict priority class and specifies the amount of bandwidth, in kbps, to be assigned to the class.

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:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	Specifies the name of the policy map to be created or modified. • The bandwidth queues and the priority queue use the same policy map.
Step 2	`Router(config-pmap)# class class-name`	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 3	`Router(config-pmap-c)# bandwidth bandwidth-kbps`	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.

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:

	Command	Purpose
Step 1	`Router(config)# policy-map policy-map`	Specifies the name of the policy map to be created or modified. • Use this command to define the shaping policy.
Step 2	`Router(config-pmap)# class class-default`	Specifies the default class so that you can configure or modify its policy.
Step 3	`Router(config-pmap-c)# shape [average | peak] mean-rate [[burst-size] [excess-burst-size]]`	(Optional) Shapes traffic to the indicated bit rate according to the algorithm specified.
Step 4	`Router(config-pmap-c)# service-policy policy-map-name`	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).

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:

	Command	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	Purpose
Step 3	`Router(config-if)# 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.
Step 4	`Router(config-if)# 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.

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:

Step 1 Enter the **show running-config** command to verify the configuration.

```
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:

```
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:

```
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

This section provides the following configuration examples:

- Frame Relay Queueing, Shaping, and Fragmentation at the Interface Example
- Frame Relay Queueing and Fragmentation at the Interface Example

Frame Relay Queueing, Shaping, and Fragmentation at the Interface Example

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
```

Frame Relay Queueing and Fragmentation at the Interface Example

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
```

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **frame-relay fragment end-to-end**
- **show interfaces serial**

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Frame Relay PVC Bundles with QoS Support for IP and MPLS

First Published: November 25, 2002
Last Updated: February 28, 2006

Frame Relay permanent virtual circuit (PVC) bundle functionality allows you to associate a group of Frame Relay PVCs with a single next-hop address. When Frame Relay PVC bundles are used with IP, packets are mapped to specific PVCs in the bundle on the basis of the precedence value or differentiated services code point (DSCP) settings in the type of service (ToS) field of the IP header. Each packet is treated differently according to the QoS configured for each PVC.

MPLS QoS support for Frame Relay PVC bundles extends Frame Relay PVC bundle functionality to support the mapping of Multiprotocol Label Switching (MPLS) packets to specific PVCs in the bundle. MPLS packets are mapped to PVCs according to the settings of the experimental (EXP) bits in the MPLS packet header.

History for the Frame Relay PVC Bundles with QoS Support for IP and MPLS Feature

Release	Modification
12.2(13)T	This feature was introduced.
12.2(28)SB	This feature was integrated into Cisco IOS Release 12.2(28)SB.

Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

Contents

- Prerequisites for Frame Relay PVC Bundles with QoS Support for IP and MPLS, page 2
- Restrictions for Frame Relay PVC Bundles with QoS Support for IP and MPLS, page 2
- Information About Frame Relay PVC Bundles with QoS Support for IP and MPLS, page 3

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

- How to Configure Frame Relay PVC Bundles with QoS Support for IP and MPLS, page 6
- Configuration Examples for Frame Relay PVC Bundles with QoS Support for IP and MPLS, page 15
- Additional References, page 22
- Command Reference, page 23
- Glossary, page 24

Prerequisites for Frame Relay PVC Bundles with QoS Support for IP and MPLS

To implement Frame Relay PVC bundles between two routers, you must enable IP Cisco Express Forwarding switching on the routers.

To configure MPLS EXP levels on bundle member PVCs, you must have tag-switching enabled on the interface.

It is recommended (but not required) that you implement PVC Interface Priority Queueing (PIPQ) in conjunction with Frame Relay PVC bundles. This will ensure that if the interface becomes congested, higher-priority traffic can exit the interface ahead of lower-priority traffic.

Restrictions for Frame Relay PVC Bundles with QoS Support for IP and MPLS

- A PVC can be a part of one and only one PVC bundle.
- A PVC bundle may contain no more than eight PVCs.
- A PVC that is a bundle member cannot be used in any other capacity, For example a PVC bundle member cannot be configured in a map statement.
- A PVC bundle cannot perform precedence and DSCP matching at the same time. If the wrong matching scheme is configured, unpredictable behavior will result.
- A PVC bundle will not come up unless all the precedence, DSCP, or EXP levels are configured in the bundle.
- Voice over Frame Relay (VoFR) is not supported on PVC-bundle members.
- Fast switching over Frame Relay PVC bundles is not supported.

Information About Frame Relay PVC Bundles with QoS Support for IP and MPLS

Before configuring and implementing Frame Relay PVC Bundles with QoS Support for IP and MPLS, you should understand the following concepts:

- Benefits of Frame Relay PVC Bundles with QoS Support for IP and MPLS, page 3
- Frame Relay PVC Bundle Support, page 3
- Frame Relay PVC Bundle Management, page 4

Benefits of Frame Relay PVC Bundles with QoS Support for IP and MPLS

- IP or MPLS packets carrying different types of traffic can be transported on different PVCs within the same PVC bundle.
- Precedence-based PVC bundles can be converted to EXP-based PVC bundles by enabling tag-switching. EXP-based PVC bundles can be converted to precedence-based PVC bundles by disabling tag-switching.
- This feature provides flexible PVC management within a PVC bundle by allowing traffic assigned to a failed PVC to be redirected to an alternate PVC within the bundle. This feature also allows you to configure the bundle to go down when certain PVCs go down.

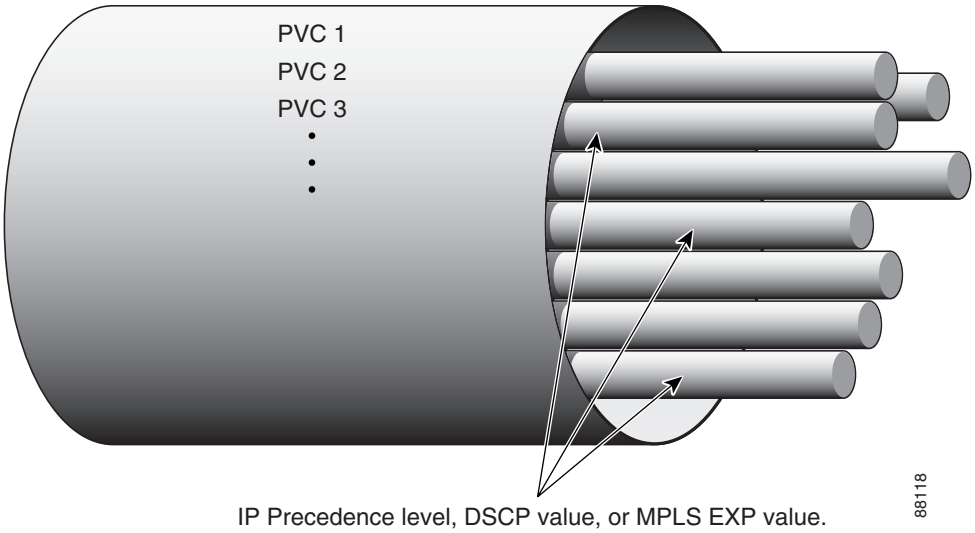
Frame Relay PVC Bundle Support

The use of Frame Relay PVC bundles allows you to configure multiple PVCs with different QoS characteristics between any pair of Frame Relay-connected routers. As shown in Figure 1, a PVC bundle may contain up to eight PVCs. The individual PVCs within a bundle are called *bundle members*.

To determine which PVC in a bundle will be used to forward a specific type of traffic, the router maps the IP precedence level or DSCP value in an IPv4 packet header to a PVC configured with the same value. In the case of MPLS, packets are mapped to specific PVCs in a bundle based on the settings of the EXP bits in the MPLS packet headers.

Once you define a Frame Relay bundle and add PVCs to it, you can configure attributes and characteristics to discrete PVC bundle members, or you can apply them collectively at the bundle level. Frame Relay traffic shaping may be applied to every PVC within a bundle. As with individual PVCs, you can enable rate adaptation to occur in response to incoming backward explicit congestion notifications (BECN) from the network.

Figure 1 Frame Relay PVC bundle



You can create differentiated service using PVC bundles by distributing IP precedence levels or DSCP values over the various bundle members. You can map either a single precedence level or a range of precedence levels to each PVC in the bundle. Thus, either you can limit an individual PVC to carry only packets marked with a specific precedence level or you can enable a PVC to carry packets marked with different precedence levels.

Service Levels and PVC Selection Criteria

The DSCP and Precedence bits classify IP packet service levels. The Precedence field consists of the first three bits of the ToS octet in the IPv4 header. These bits define eight precedence levels. When DSCP mapping is used, the DSCP octet replaces the ToS octet in the IPv4 header. Currently the first six bits are used, defining 64 service levels.

Using precedence-based or DSCP-based mapping, each IPv4 packet is mapped to a specific PVC in the bundle, according to the value of the ToS or DSCP octet in the IP header. There is no special treatment for broadcast or multicast or IP routing packets; the only differentiation in treatment is a result of the ToS or DSCP octet settings.

The MPLS EXP bits make up a three-bit experimental field in the MPLS packet header. They are a bit-by-bit copy of the IP Precedence bits and provide the same eight QoS levels. Under MPLS EXP-based mapping, each MPLS packet is mapped to a specific PVC in the bundle, according the setting of the EXP bits.

Frame Relay PVC Bundle Management

In addition to mapping specific traffic types to specific PVCs according to QoS parameters designated by the ToS or DSCP values in the IPv4 headers or EXP values in the MPLS headers, PVC bundle management takes care of handling non-IP traffic and determining what happens if a PVC goes down.

By default, Inverse Address Resolution Protocol (ARP) traffic and other critical non-IP traffic is carried by the PVC configured for carrying IP Precedence or EXP level 6 or DSCP level 63. You can select a PVC with a different QoS to carry Inverse ARP traffic if required. Noncritical non-IP traffic is carried by the PVC that configured for carrying IP precedence, EXP, or DSCP level 0.

It is important during configuration to account for every precedence, EXP, or DSCP level in the configuration of the PVC bundle members. If all the packet service levels are not accounted for, the PVC bundle will never come up.

Once a PVC bundle is up, if an individual bundle member goes down, an attempt is made to identify an alternate PVC to handle the packet service level or levels that were carried by the downed PVC. If no alternate PVC is found, the entire PVC bundle is brought down.

Traffic Bumping

You can configure each PVC bundle member to bump traffic to another PVC in the bundle in the event that the bundle member goes down. You can specify whether the bumping will be implicit or explicit bumping. You can also specify that a particular PVC will never accept bumped traffic from another PVC. The default conditions are to perform implicit traffic bumping and to accept bumped traffic.

Implicit bumping diverts the traffic from a failed PVC to the PVC having the next-lower service level. Explicit bumping forces the traffic to a specific PVC rather than allowing it to find a PVC carrying traffic of the next-lower service level. For example, PVC *x*, responsible for carrying precedence level 3 traffic, can be configured to bump its traffic to PVC *y*, responsible for carrying precedence level 6 traffic—provided that PVC *y* is configured to accept bumped traffic. If PVC *x* goes down, PVC *y* takes over. If PVC *y* is already down or goes down later, the alternate PVC selected will depend on the bumping rule for PVC *y*. If no alternate PVC can be found for bumped traffic, the entire PVC bundle goes down.

PVC-Bundle Protection Rules

Traffic bumping provides a way to keep a PVC bundle up and traffic flowing even though some individual PVCs may be down. Protection rules provide a way to force the PVC bundle down even though some individual PVCs are up and might be able to handle all the traffic, though perhaps not in a satisfactory manner.

You can configure a PVC bundle member as an individually protected PVC or as part of a PVC bundle protected group. Only one protected group may exist within a PVC bundle; however, many individually protected PVCs may exist. The protection rules add flexibility for controlling the PVC bundle state.

When any one individually protected PVC goes down, the entire bundle goes down. If all the PVCs in a protected group go down, the entire bundle goes down.

If no protection rule is specified, the PVC bundle goes down only when all the PVCs go down. However, protection is overridden if a PVC that has no place to bump its traffic goes down. In this case, the entire bundle will go down despite any protection rules that have been set up.

MPLS EXP-based Mapping

To enable MPLS EXP-based mapping, tag-switching must be enabled on the interface or subinterface by using the **tag-switching ip** command. When tag-switching is enabled, MPLS and IP packets can flow across the interface and PVC bundles that are configured for IP Precedence mapping are converted to MPLS EXP mapping. The PVC bundle functionality remains the same with respect to priority levels, bumping, and so on, but the **match precedence** command is replaced by **match exp**, and each **precedence** command is replaced by the **exp** command. The effect is that a bundle member PVC previously configured to carry precedence level 1 IP traffic now carries EXP level 1 MPLS traffic.

PVC bundles configured for DSCP mapping go down when tag-switching is enabled. The DSCP configuration for each bundle member PVC is reset, resulting in the PVCs being unmapped and Inverse ARP, bumping, and protection settings being unconfigured. The **match dscp** command is replaced by **match exp** command.

When tag-switching is disabled, the **match precedence** and **match dscp** commands are restored and the **exp** commands are replaced by **precedence** commands.

When tag-switching is enabled or disabled, PVC bundles configured for IP precedence mapping or MPLS EXP mapping will stay up and traffic will transmit over the appropriate bundle member PVCs.

How to Configure Frame Relay PVC Bundles with QoS Support for IP and MPLS

This section contains the following configuration tasks.

- Configuring Frame Relay PVC Bundles with IP QoS Support, page 6 (required)
- Configuring Frame Relay PVC Bundles with MPLS QoS Support, page 10 (required)
- Verifying Frame Relay PVC Bundles Configuration, page 13 (optional)
- Monitoring and Maintaining Frame Relay PVC Bundles, page 14 (optional)

Configuring Frame Relay PVC Bundles with IP QoS Support

To configure Frame Relay PVC bundles for handling IP packets, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **ip cef**
5. **interface** *type number*
   or
   interface *type number.subinterface-number* [**multipoint** | **point-to-point**]
6. **encapsulation frame-relay** [**cisco** | **ietf**]
7. **ip address** *ip-address mask* [**secondary**]
8. **frame-relay map** *protocol protocol-address* {*dlci* | **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf** | **cisco**]
9. **frame-relay vc-bundle** *vc-bundle-name*
10. **encapsulation** [**cisco**| **ietf**]
11. **match** {**dscp** *dscp-value* | **precedence** *precedence-value*}
12. **pvc** *dlci* [*vc-name*]
13. **class** *name*

14. **precedence** {*level* | **other**}
    or
    dscp {*level* | **other**}
15. **bump** {**explicit** *level* | **implicit** | **traffic**}
16. **protect** {**group** | **vc**}
17. **inarp**
18. **end**
19. Configure the PVC bundle on the peer router.

DETAILED STEPS

	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.
Step 4	**ip cef** **Example:** `Router(config)# ip cef`	Enables Cisco Express Forwarding. **Note** For the Cisco 7500, enter **ip cef distributed**.

Step 5	**interface** *type number*
	
	or
	
	interface *type number.subinterface-number* [**multipoint** | **point-to-point**]
	
	Example:
	`Router(config)# interface serial 0`
	
	or
	
	Example:
	`Router(config)# interface serial 1.1 multipoint`
		Specifies the interface type and number and enters interface configuration mode.
		• Physical interfaces are multipoint subinterfaces by default.
		or
		Specifies the interface type and subinterface and enters subinterface configuration mode.
		• Once you create a specific type of subinterface (point-to-point or multipoint), you cannot change it without a reload. To change it, you must either reload the router or create another subinterface.

	Command or Action	Purpose
Step 6	**encapsulation frame-relay** [**cisco** |**ietf**] **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation. • The default encapsulation method is **cisco**.
Step 7	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.1.1.1 255.0.0.0`	Sets a primary IP address for the interface. • The optional **secondary** keyword specifies that the configured address is a secondary IP address. If this keyword is omitted, the configured address is the primary IP address.
Step 8	**frame-relay map** *protocol protocol-address* {*dlci* | **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf** | **cisco**] **Example:** `Router(config-if)# frame-relay map ip 10.2.2.2 vc-bundle MAIN-1`	(Optional) Maps between a next-hop protocol address and a data-link connection identifier (DLCI) destination address, and creates a PVC bundle if it does not already exist. • The protocol-address is the destination IP address. • The **frame-relay map** command is required for multipoint interfaces if Inverse ARP has been disabled or is not supported at the other end of the connection.
Step 9	**frame-relay vc-bundle** *vc-bundle-name* **Example:** `Router(config-if)# frame-relay vc-bundle MAIN-1`	Creates a PVC bundle if it does not already exist, and enters Frame Relay VC-bundle configuration mode.
Step 10	**encapsulation** [**cisco** | **ietf**] **Example:** `Router(config-fr-vcb)# encapsulation ietf`	(Optional) Overrides the encapsulation type configured on the interface and configures the Frame Relay encapsulation type for the PVC bundle. • This command is available only when the PVC bundle is configured on a point-to-point subinterface.
Step 11	**match** {**dscp** *dscp-value* | **precedence** *precedence-value*} **Example:** `Router(config-fr-vcb)# match precedence 5`	Establishes the type of matching to use between incoming packet headers and PVC-bundle members. • The default match type is **precedence.**
Step 12	**pvc** *dlci* [*vc-name*] **Example:** `Router(config-fr-vcb)# pvc 100 1a`	Creates a PVC-bundle member and enters Frame Relay VC-bundle-member configuration mode. • The *vc-name* argument is an optional name that can be used for referring to the PVC.
Step 13	**class** *name* **Example:** `Router(config-fr-vcb-vc)# class premium`	(Optional) Assigns a map class to the PVC-bundle member defined in the previous step.

	Command or Action	Purpose

Step 14	**precedence** {*level* | **other**}
	
	or
	
	dscp {*level* | **other**}
	
	Example:
	`Router(config-fr-vcb-vc)# precedence 6-7`
	
	or
	
	Example:
	`Router(config-fr-vcb-vc)# dscp other`
		(Optional) Enters the mapped service level or range for the PVC-bundle member.
		• The **precedence** command is available when the PVC-bundle match type is set to **precedence**.
		• The precedence range is from 0 to 7.
		• The **dscp** command is available when the PVC-bundle match type is set to **dscp**.
		• The **dscp** range is from 0 to 63.
		• The **other** keyword is used to designate a PVC to handle all the remaining levels that have not been assigned to other PVCs in the bundle.
		• Critical non-IP traffic will automatically use precedence level 0.

Step 15	**bump** {**explicit** *level* | **implicit** | **traffic**} **Example:** `Router(config-fr-vcb-vc)# bump explicit 7`	(Optional) Specifies the bumping rule for the PVC-bundle member. • The default bumping rule is implicit bumping. • Use the **explicit** *level* option to specify the service level to which traffic on this PVC will be bumped if the PVC goes down. In that event, the traffic will be directed to a PVC mapped with the service level configured here. If the PVC that picks up and carries the traffic also goes down, the traffic is subject to the bumping rules for that PVC. You can specify only one service level for bumping. • The PVC-bundle member accepts bumped traffic by default when the PVC-bundle match type is **precedence**. To configure the PVC to reject bumped traffic from another PVC-bundle member, use the **no bump traffic** command.
Step 16	**protect** {**group** | **vc**} **Example:** `Router(config-fr-vcb-vc)# protect group`	(Optional) Specifies the protection rule for the PVC-bundle member. • By default, the PVC-bundle member is not protected. • If you use the **vc** keyword, the PVC bundle goes down whenever this PVC goes down. • If you use the **group** keyword, the PVC bundle goes down when the last PVC in the protected group goes down.
Step 17	**inarp** **Example:** `Router(config-fr-vcb-vc)# inarp`	(Optional) Enables Inverse ARP for the PVC-bundle member. • By default, Inverse ARP traffic uses the PVC configured for precedence level 6 or DSCP level 63.

	Command or Action	Purpose
Step 18	**end** **Example:** `Router(config-fr-vcb-vc)# end`	Exits to privileged EXEC mode.
Step 19	Configure the PVC bundle on the peer router.	(Optional) While it is not necessary to configure a PVC bundle on the peer router, it is recommended that you do so for applications that rely on communications on the same PVC (such as TCP header-compression.)

Configuring Frame Relay PVC Bundles with MPLS QoS Support

To configure Frame Relay PVC bundles for handling MPLS packets, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **ip cef**
5. **interface** *type number*

   or

   interface {*type slot* | *port-adapter* | *port***.***subinterface-number*}[**multipoint** | **point-to-point**]
6. **encapsulation frame-relay** [**cisco** | **ietf**]
7. **tag-switching ip**
8. **ip address** *ip-address mask* [**secondary**]
9. **frame-relay map** *protocol protocol-address* {*dlci* | **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf** | **cisco**]
10. **frame-relay vc-bundle** *vc-bundle-name*
11. **encapsulation** [**ietf** | **cisco**]
12. **pvc** *dlci* [*vc-name*]
13. **class** *name*
14. **exp** {*level* | **other**}
15. **bump** {**explicit** *level* | **implicit** | **traffic**}
16. **protect** {**group** | **vc**}
17. **inarp**
18. **end**
19. Configure the PVC bundle on the peer router.

DETAILED STEPS

	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.
Step 4	**ip cef** **Example:** `Router(config)# ip cef`	Enables Cisco Express Forwarding. **Note** For the Cisco 7500, enter **ip cef distributed**.

Step 5	**interface** *type number*
	
	or
	
	interface {*type slot* | *port-adapter* | *port.subinterface-number*}[**multipoint** | **point-to-point**]
	
	Example:
	`Router(config)# interface serial 0`
	
	or
	
	Example:
	`Router(config)# interface serial 1.1 multipoint`
		Specifies the interface type and number and enters interface configuration mode.
		• Physical interfaces are multipoint subinterfaces by default.
		or
		Specifies the interface type and subinterface and enters subinterface configuration mode.
		• Once you create a specific type of subinterface (point-to-point or multipoint), you cannot change it without a reload. To change it, you need to either reload the router or create another subinterface.

Step 6	**encapsulation frame-relay** [**cisco** | **ietf**] **Example:** `Router(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation. • The default encapsulation method is **cisco**.
Step 7	**tag-switching ip** **Example:** `Router(config-if)# tag-switching ip`	Enables label switching of IPv4 packets on an interface.

	Command or Action	Purpose
Step 8	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.1.1.1 255.0.0.0`	Sets a primary IP address for the interface. • The optional **secondary** keyword specifies that the configured address is a secondary IP address. If this keyword is omitted, the configured address is the primary IP address.
Step 9	**frame-relay map** *protocol protocol-address* {*dlci* | **vc-bundle** *vc-bundle-name*} [**broadcast**] [**ietf** | **cisco**] **Example:** `Router(config-if)# frame-relay map ip 10.2.2.2 vc-bundle MAIN-1`	(Optional) Maps between a next-hop protocol address and a DLCI destination address, and creates a PVC bundle if it does not already exist. • The protocol-address is the destination IP address. • The **frame-relay map** command is required for multipoint interfaces if Inverse ARP has been disabled or is not supported at the other end of the connection.
Step 10	**frame-relay vc-bundle** *vc-bundle-name* **Example:** `Router(config-if)# frame-relay vc-bundle MAIN-1`	Creates a PVC bundle if it does not already exist, and enters Frame Relay VC-bundle configuration mode.
Step 11	**encapsulation** [**ietf** | **cisco**] **Example:** `Router(config-fr-vcb)# encapsulation ietf`	(Optional) Overrides the encapsulation type configured on the interface and configures the Frame Relay encapsulation type for the PVC bundle. • This command is available only when the PVC bundle is configured on a point-to-point subinterface.
Step 12	**pvc** *dlci* [*vc-name*] **Example:** `Router(config-fr-vcb)# pvc 100 1a`	Creates a PVC-bundle member and enters Frame Relay VC-bundle-member configuration mode. • The *vc-name* argument is an optional name that can be used for referring to the PVC.
Step 13	**class** *name* **Example:** `Router(config-fr-vcb-vc)# class premium`	(Optional) Assigns a map class to the PVC-bundle member.
Step 14	**exp** {*level* | **other**} **Example:** `Router(config-fr-vcb-vc)# exp 6-7`	(Optional) Enters the mapped EXP level or range for the PVC-bundle member. • The **exp** command is available only when tag-switching has been enabled. • The EXP level values are from 0 to 7. • The **other** keyword is used to designate a PVC to handle all the remaining levels that have not been assigned to other PVCs in the bundle.

	Command or Action	Purpose
Step 15	**bump** {**explicit** *level* | **implicit** | **traffic**} **Example:** `Router(config-fr-vcb-vc)# bump explicit 7`	(Optional) Specifies the bumping rule for the PVC-bundle member defined above. • The default bumping rule is implicit bumping. • Use the **explicit** *level* option to specify the EXP level to which traffic on this PVC will be bumped if the PVC goes down. In that event, the traffic will be directed to a PVC mapped with the EXP level configured here. If the PVC that picks up and carries the traffic also goes down, the traffic is subject to the bumping rules for that PVC. You can specify only one EXP level for bumping. • To configure the PVC to reject bumped traffic from another PVC-bundle member, use the **no bump traffic** command.
Step 16	**protect** {**group** | **vc**} **Example:** `Router(config-fr-vcb-vc)# protect group`	(Optional) Specifies the protection rule for the PVC-bundle member defined above. • By default, the PVC-bundle member is not protected. • If you use the **vc** keyword, the PVC bundle goes down whenever this PVC goes down. • If you use the **group** keyword, the PVC bundle goes down when the last PVC in the protected group goes down.
Step 17	**inarp** **Example:** `Router(config-fr-vcb-vc)# inarp`	(Optional) Enables Inverse ARP for the PVC-bundle member defined above. • By default, Inverse ARP traffic uses the PVC configured for EXP level 6.
Step 18	**end** **Example:** `Router(config-fr-vcb-vc)# end`	(Optional) Exits to privileged EXEC mode.
Step 19	Configure the PVC bundle on the peer router.	(Optional) While it is not necessary to configure a PVC bundle on the peer router, it is recommended that you do so for applications that rely on communications on the same PVC (such as TCP header-compression.)

Verifying Frame Relay PVC Bundles Configuration

To verify the configuration and operation of Frame Relay PVC bundles with QoS support, perform the following optional steps:

SUMMARY STEPS

1. **enable**
2. **show frame-relay vc-bundle** *vc-bundle-name* [**detail**]
3. **show frame-relay map**

4. **show frame-relay pvc**
5. **show frame-relay ip rtp header-compression** [**interface** *type number*]
6. **show frame-relay ip tcp header-compression** [**interface** *type number*]
7. **show adjacency** [*type number*] [**detail**] [**summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show frame-relay vc-bundle** *vc-bundle-name* [**detail**] **Example:** `Router# show frame-relay vc-bundle mp-3-static`	Displays status, bumping information, protection information, and active and configured precedence or DSCP levels for the PVCs in a PVC bundle.
Step 3	**show frame-relay map** **Example:** `Router# show frame-relay map`	Displays the current Frame Relay map entries and information about the connections.
Step 4	**show frame-relay pvc** **Example:** `Router# show frame-relay pvc`	Displays PVC statistics for the PVC-bundle members.
Step 5	**show frame-relay ip rtp header-compression** [**interface** *type number*] **Example:** `Router# show frame-relay ip rtp header-compression`	Displays Frame Relay Real-Time Transport Protocol (RTP) header compression statistics for PVC bundles.
Step 6	**show frame-relay ip tcp header-compression** [**interface** *type number*] **Example:** `Router# show frame-relay ip tcp header-compression serial 1/4`	Displays Frame Relay TCP/IP header compression statistics for PVC bundles.
Step 7	**show adjacency** [*type number*] [**detail**] [**summary**] **Example:** `Router# show adjacency`	Displays Cisco Express Forwarding adjacency table information.

Monitoring and Maintaining Frame Relay PVC Bundles

To monitor and maintain Frame Relay PVC bundles, perform this task.

SUMMARY STEPS

1. **enable**
2. **debug frame-relay adjacency** {**pvc** [*dlci*] | **vc-bundle** [*vc-bundle-name*]}
3. **debug frame-relay vc-bundle** {**detail** | **state-change**} [*vc-bundle-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug frame-relay adjacency** {**pvc** [*dlci*] | **vc-bundle** [*vc-bundle-name*]} **Example:** `Router# debug frame-relay adjacency pvc`	Displays information pertaining to an adjacent node that has one or more Frame Relay PVCs or PVC bundles. • Use this command to monitor adjacency activity.
Step 3	**debug frame-relay vc-bundle** {**detail** | **state-change**} [*vc-bundle-name*] **Example:** `Router# debug frame-relay vc-bundle state-change`	Displays information about the Frame Relay PVC bundles configured on a router. • Use this command to monitor state changes and Inverse ARP activity for one or all of the PVC bundles and bundle members configured on a router. **Note** Using the **detail** keyword generates a large number of debugs that can quickly fill up a log buffer.

Configuration Examples for Frame Relay PVC Bundles with QoS Support for IP and MPLS

This section provides the following configuration examples:

- PVC Bundles with IP QoS Support on Main, Multipoint, and Point-to-Point Interfaces Example, page 16
- PVC Bundle with IP QoS Support with Multiple QoS Parameters Example, page 17
- PVC Bundle with MPLS QoS Support Example, page 18
- Verifying Frame Relay PVC Bundle Configuration Examples, page 18
- Monitoring and Maintaining Frame Relay PVC Bundles Examples, page 20

PVC Bundles with IP QoS Support on Main, Multipoint, and Point-to-Point Interfaces Example

The following example shows the configuration of five PVC bundles with IP precedence and DSCP mapping. Two bundles are configured on the main interface, one bundle with static mapping and one with dynamic mapping. Two bundles are configured on a multipoint subinterface, one bundle with static mapping and one with dynamic mapping. One bundle is configured on a point-to-point subinterface.

```
configure terminal
ip routing
ip cef
interface Serial 1/4
 encapsulation frame-relay
 frame-relay intf-type dte
 ip address 10.1.1.1 255.0.0.0
 frame-relay map ip 192.168.2.2 vc-bundle MAIN-1-static
 frame-relay vc-bundle MAIN-1-static
 match precedence
 pvc 100 1a
 precedence other
 pvc 101 1b
 precedence 1
 pvc 102 1c
 precedence 2
 pvc 103 1d
 precedence 3
 pvc 104 1e
 precedence 4
 pvc 105 1f
 precedence 5
 pvc 106 1g
 precedence 6
 pvc 107 1h

 frame-relay vc-bundle MAIN-2-dynamic
 match precedence
 pvc 200
 precedence 0
 pvc 201
 precedence 1
 pvc 202
 precedence 2
 pvc 203
 precedence 3
 pvc 204
 precedence 4
 pvc 205
 precedence 5
 pvc 206
 precedence 6
 pvc 207
 precedence 7

interface Serial 1/4.1 multipoint
 ip address 172.16.1.1 255.0.0.0
 frame-relay map ip 172.17.2.2 vc-bundle MP-3-static
 frame-relay vc-bundle MP-3-static
  match precedence
 pvc 300 3a
 precedence 0
 pvc 301 3b
 precedence 1
```

```
 pvc 302 3c
 precedence 2
 pvc 303 3d
 precedence 3
 pvc 304 3e
 precedence 4
 pvc 305 3f
 precedence 5
 pvc 306 3g
 precedence 6
 pvc 307 3h
 precedence 7

interface Serial 1/4.1 multipoint
 frame-relay vc-bundle MP-4-dynamic
 match precedence
 match dscp
 pvc 400 4a
 dscp other
 pvc 401 4b
 dscp 10-19
 pvc 402 4c
 dscp 20-29
 pvc 403 4d
 dscp 30-39
 pvc 404 4e
 dscp 40-49
 pvc 405 4f
 dscp 50-59
 pvc 406 4g
 dscp 60-62
 pvc 407 4h
 dscp 63
 end

interface Serial 1/4.2 point-to-point
 ip address 192.168.2.1 255.0.0.0
 frame-relay vc-bundle P2P-5
 match precedence
 pvc 500 5a
 precedence 0
 pvc 501 5b
 precedence 1
 pvc 502 5c
 precedence 2
 pvc 503 5d
 precedence 3
 pvc 504 5e
 precedence 4
 pvc 505 5f
 precedence 5
 pvc 506 5g
 precedence 6
 pvc 507 5h
 precedence 7
```

PVC Bundle with IP QoS Support with Multiple QoS Parameters Example

The following example shows the configuration of a Frame Relay PVC bundle with DSCP-based mapping. The bundle member PVCs are configured with bumping, protection, and other parameters.

```
interface Serial 1/4.2 point-to-point
```

```
 frame-relay vc-bundle BUNDLE-SEFEN
 encapsulation ietf
 match dscp
  pvc 301
 dscp other
 bump explicit 45
 protect group
 class CIR-64000
 pvc 302
 dscp 40-49
 bump explicit 20
 no bump traffic
 protect vc
 inarp
 pvc 303
 dscp 30-39
 bump implicit
 protect group
```

PVC Bundle with MPLS QoS Support Example

The following example shows the configuration of four Frame Relay PVC bundle members with MPLS EXP level support in the PVC bundle named “user1”.

```
interface serial 0.1 point-to-point
 encapsulation frame-relay
 ip address 10.1.1.1
 tag-switching ip
 frame-relay vc-bundle user1
 pvc 100 ny-control
 class control
 exp 7
 protect vc
 pvc 101 ny-premium
 class premium
 exp 6-5
 bump explicit 7
 no bump traffic
 protect group
 pvc 102 my-priority
 class priority
 exp 4-2
 protect group
 pvc 103 ny-basic
 class basic
 exp other
```
protect group

Verifying Frame Relay PVC Bundle Configuration Examples

The following examples show output for the commands that can be used to verify Frame Relay PVC bundle configuration.

Sample Output for the show frame-relay vc-bundle Command

The following example shows the Frame Relay PVC bundle named “MP-4-dynamic” with PVC protection applied. Note that in this PVC bundle, DLCI 400 is configured to bump traffic explicitly to the PVC that handles DSCP level 40, which is DLCI 404. All the other DLCIs are configured for implicit bumping. In addition, all the DLCIs are configured to accept bumped traffic.

The asterisk (*) before PVC 4a indicates that this PVC was configured with the **precedence other** command, which means the PVC will handle all levels that are not explicitly configured on other PVCs.

In this example all PVCs are up so the values in the “Active level” fields match the values in the “Config level” fields. If a PVC goes down and its traffic is bumped, the “Active level” field value for the PVC that went down is cleared. The “Active level” field values for the PVC that the traffic bumped to will be updated to include the levels of the PVC that went down.

The first three PVCs in the following example make up a protected group. All three of these PVCs must go down before the bundle will go down. The last two PVCs are protected PVCs: if either of these PVCs go down, the bundle will go down.

```
Router# show frame-relay vc-bundle MP-4-dynamic

MP-4-dynamic on Serial 1/4.1 - Status: UP Match-type: DSCP

Name    DLCI    Config. Active     Bumping     PG/     CIR     Status
                level   level      to/accept   PV      kbps
*4a     400     0-9     0-9        40/Yes      pg              up
4b      401     10-19   10-19      9/Yes       pg              up
4c      402     20-29   20-29      19/Yes      pg              up
4d      403     30-39   30-39      29/Yes      -               up
4e      404     40-49   40-49      39/Yes      -               up
4f      405     50-59   50-59      49/Yes      -               up
4g      406     60-62   60-62      59/Yes      pv              up
4h      407     63      63         62/Yes      pv              up

Packets sent out on vc-bundle MP-4-dynamic : 0:
Router#
```

The following example shows the detail output of a PVC bundle. Note in this example that because all packet service levels are not handled, and because the PVCs are currently down, this bundle can never come up.

```
Router# show frame-relay vc-bundle x41 detail

x41 on Serial1/1 - Status: DOWN Match-type: DSCP

Name    DLCI    Config. Active     Bumping     PG/     CIR     Status
                level   level      to/accept   PV      kbps

        410     50-62              49/Yes      -               down
        411     30,32,34,36,3..    29/Yes      -               down

Packets sent out on vc-bundle x41 : 0

Active configuration and statistics for each member PVC
DLCI    Output pkts    Active level
410     0              50-62
411     0              30,32,34,36,38-40
Router#
```

Sample Output for the show frame-relay map Command

The following sample output displays map and connection information for a PVC bundle called "MAIN-1-static":

```
Router# show frame-relay map

Serial1/4 (up):ip 10.2.2.2 vc-bundle MAIN-1-static, static,
              CISCO, status up
```

Sample Output for the show frame-relay pvc Command

The following sample output indicates that PVC 202 is a member of VC bundle "MAIN-1-static":

```
Router# show frame-relay pvc 202

PVC Statistics for interface Serial1/4 (Frame Relay DTE)

DLCI = 202, DLCI USAGE = LOCAL, PVC STATUS = STATIC, INTERFACE = Serial1/4

  input pkts 0             output pkts 45           in bytes 0
  out bytes 45000          dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
  in DE pkts 0             out DE pkts 0
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 2000 bits/sec, 2 packets/sec
  pvc create time 00:01:25, last time pvc status changed 00:01:11
  VC-Bundle MAIN-1-static
```

Sample Output for the show adjacency Command

The following is sample output for the **show adjacency** command for a PVC bundle configured on serial subinterface 1/4.1. Each bundle member is listed. The bundle itself is indicated by "incomplete" because no traffic actually transmitted on that entry.

```
Router# show adjacency

      Protocol Interface                Address
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(4)
      IP       Serial1/4.1              10.2.2.2(5) (incomplete)
```

Monitoring and Maintaining Frame Relay PVC Bundles Examples

The following examples show output for the **debug frame-relay adjacency** and **debug frame-relay vc-bundle** commands, which can be used to troubleshoot Frame Relay PVC bundle operation. "FR-VCB" indicates output from the **debug frame-relay vc-bundle** command, and "FR-ADJ" indicates output from the **debug frame-relay adjacency** command.

Note Debug messages that are prefixed with "FR_ADJ" (instead of FR-ADJ") or "FR_VCB" (instead of "FR-VCB") indicate serious failures in the Frame Relay PVC bundle performance. Contact the Cisco Technical Assistance Center (TAC) if you see debug messages with these prefixes.

The following is sample output that shows a PVC bundle that uses static map coming up. PVC bundle member 100 comes up first, then the PVC bundle itself can come up.

```
Router# debug frame-relay vc-bundle state-change
Router# debug frame-relay adjacency vc-bundle

00:35:58:FR-VCB:MAIN-1-static:member 100 state changed to UP
00:35:58:FR-VCB:MAIN-1-static:state changed to UP
00:35:58:FR-ADJ:vcb MAIN-1-static:ip 10.2.2.2:adding primary adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:adding adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 0 00:35:58:FR-ADJ:vcb
MAIN-1-static:member 100:locking adj at index 1
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 2
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 3
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 4
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 5
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 6
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:locking adj at index 7
00:35:58:%FR-5-DLCICHANGE:Interface Serial1/4 - DLCI 100 state changed to ACTIVE
00:35:58:FR-VCB:MAIN-1-static:member 101 state changed to UP
00:35:58:FR-ADJ:vcb MAIN-1-static:ip 10.2.2.2:updating primary adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:updating adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 101:adding adj
00:35:58:FR-ADJ:vcb MAIN-1-static:member 100:unlocking adj at index 1
00:35:58:FR-ADJ:vcb MAIN-1-static:member 101:locking adj at index 1
```

The following is sample output that shows a PVC bundle going down. Each bundle member PVC is marked for removal from Cisco Express Forwarding adjacency table, and then the adjacency for the PVC bundle itself is marked for removal. The adjacencies are actually removed from the table later when a background clean-up process runs.

```
00:38:35:FR-VCB:MP-3-static:state changed to DOWN
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 300:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 301:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 302:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 303:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 304:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:member 305:removing adj
00:38:35:FR-ADJ:vcb MP-3-static:ip 172.17.2.2:removing primary adj
```

The following is sample output that shows Inverse ARP information for the PVC bundle. PVC bundle member 406 is the only PVC in the bundle to handle Inverse ARP packets. The Inverse ARP packets coming in on other bundle member PVCs are dropped.

```
00:23:48:FR-VCB:MP-4-dynamic:inarp received on elected member 406
00:23:48:FR-VCB:MP-4-dynamic:installing dynamic map
00:23:48:FR-VCB:MP-4-dynamic:dropping inarp received on member 407
00:23:52:FR-VCB:MP-4-dynamic:sending inarp pkt on member 406
```

In the following example the PVC bundle goes down because the protected group goes down. All information about active transmission on each PVC is removed.

```
00:58:27:FR-VCB:MP-4-dynamic:member 402 state changed to DOWN
00:58:27:FR-VCB:MP-4-dynamic:protected group is DOWN
00:58:27:FR-VCB:MP-4-dynamic:state changed to DOWN
00:58:27:FR-VCB:MP-4-dynamic:active table reset
```

Additional References

The following sections provide references related to Frame Relay PVC Bundles with QoS Support for IP and MPLS.

Related Documents

Related Topic	Document Title
Frame Relay configuration tasks	*"Configuring Frame Relay" chapter in the Cisco IOS Wide-Area Networking Configuration Guide*, Release 12.2
Frame Relay commands	*"Frame Relay Commands" chapter in the Cisco IOS Wide-Area Networking Command Reference*, Release 12.2
Frame Relay PVC interface priority queueing configuration tasks	"Configuring Weighted Fair Queueing" section in the Congestion Management chapter in the *Cisco IOS Quality of Service Configuration Guide*, Release 12.2

Standards

Standards	Title
None	—

MIBs

MIBs	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

RFCs	Title
None	—

Technical Assistance

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

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **bump (Frame Relay VC-bundle-member)**
- **class**
- **debug frame-relay adjacency**
- **debug frame-relay vc-bundle**
- **dscp (Frame Relay VC-bundle-member)**
- **encapsulation (Frame Relay VC-bundle)**
- **exp**
- **frame-relay inverse-arp**
- **frame-relay map**
- **frame-relay vc-bundle**
- **inarp (Frame Relay VC-bundle-member)**
- **match**
- **precedence (Frame Relay VC-bundle-member)**
- **protect (Frame Relay VC-bundle-member)**
- **pvc (Frame Relay VC-bundle)**
- **show frame-relay ip rtp header-compression**
- **show frame-relay ip tcp header-compression**
- **show frame-relay map**
- **show frame-relay pvc**
- **show frame-relay vc-bundle**

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.

VoFR—Voice over Frame Relay. Enables a router to carry voice traffic over a Frame Relay network. When voice traffic is sent over Frame Relay, the voice traffic is segmented and encapsulated for transit across the Frame Relay network using FRF.12 encapsulation.

Voice over Frame Relay—See VoFR.

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.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Frame Relay PVC Interface Priority Queueing

Feature History

Release	Modification
Cisco IOS	For information about feature support in Cisco IOS software, use Cisco Feature Navigator.

This feature module describes the Frame Relay PVC Interface Priority Queueing (FR PIPQ) feature. It includes information on the benefits of this new feature, supported platforms, related documents, and so on.

This document includes the following sections:

- Feature Overview, page 1
- Supported Platforms, page 3
- Supported Standards, MIBs, and RFCs, page 3
- Prerequisites, page 4
- Configuration Tasks, page 4
- Monitoring and Maintaining FR PIPQ, page 5
- Configuration Examples, page 6
- Command Reference, page 7
- Glossary, page 7

Feature Overview

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.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

FR PIPQ provides four levels of priority: high, medium, normal, and low. The Frame Relay packet is examined at the interface for the data-link connection identifier (DLCI) value. The packet is then sent to the correct priority queue based on the priority level configured for that DLCI.

Note When using FR PIPQ, configure the network so that different types of traffic are transported on separate PVCs. FR PIPQ is not meant to be used when an individual PVC carries different traffic types that have different quality of service (QoS) requirements.

You assign priority to a PVC within a Frame Relay map class. All PVCs using or inheriting that map class will be classed according to the configured priority. If a PVC does not have a map class associated with it, or if the map class associated with it does not have priority explicitly configured, then the packets on that PVC will be queued on the default "normal" priority queue.

If you do not enable FR PIPQ on the interface using the **frame-relay interface-queue priority** command in interface configuration mode, configuring PVC priority within a map class will not be effective. At this time you have the option to also set the size (in maximum number of packets) of the four priority queues.

FR PIPQ works with or without Frame Relay traffic shaping (FRTS) and FRF.12. The interface-level priority queueing takes the place of the FIFO queueing or dual FIFO queueing normally used by FRTS and FRF.12. PVC priority assigned within FR PIPQ takes precedence over FRF.12 priority, which means that all packets destined for the same PVC will be queued on the same interface queue whether they were fragmented or not.

Note Although high priority PVCs most likely will transport only small packets of voice traffic, you may want to configure FRF.12 on these PVCs anyway to guard against any unexpectedly large packets.

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.

Restrictions

The following restrictions apply to FR PIPQ:

- 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.

Related Features and Technologies

The following features and technologies are related to FR PIPQ:

- FRTS
- FRF.12

Related Documents

The following documents provide information related to FR PIPQ:

- *Cisco IOS Wide-Area Networking Configuration Guide*, Release 12.1
- *Cisco IOS Wide-Area Networking Command Reference*, Release 12.1

Supported Platforms

- Cisco 1000 series
- Cisco 1400 series
- Cisco 1600
- Cisco 1700
- Cisco 1750
- Cisco 2500
- Cisco 2600
- Cisco 3600
- Cisco 3810
- Cisco 4500
- Cisco 4700
- Cisco 7200
- Cisco 7500 (in nondistributed mode)

Supported Standards, MIBs, and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

No new or modified MIBs are supported by this feature.

For descriptions of supported MIBs and how to use MIBs, see the Cisco MIB web site on CCO at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

RFCs

No new or modified standards are supported by this feature.

Prerequisites

The following prerequisites apply to FR PIPQ:

- 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.

Configuration Tasks

See the following sections for configuration tasks for the FR PIPQ feature. Each task in the list is identified as either optional or required:

- Configuring PVC Priority in a Map Class (Required)
- Enabling FR PIPQ and Setting Queue Limits (Required)
- Assigning a Map Class to a PVC (Required)

Configuring PVC Priority in a Map Class

To configure PVC priority within a map class, use the following commands beginning in global configuration mode:

	Command	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:

	Command	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:

	Command	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.
`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

This section provides configuration examples for FR PIPQ.

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
```

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **frame-relay interface-queue priority**
- **show frame-relay pvc**
- **show interfaces**
- **show queueing**
- **debug priority**

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.

VoFR—Voice over Frame Relay. Enables a router to carry voice traffic over a Frame Relay network. When voice traffic is sent over Frame Relay, the voice traffic is segmented and encapsulated for transit across the Frame Relay network using FRF.12 encapsulation.

Voice over Frame Relay—See VoFR.

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.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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

© 2007, Cisco Systems, Inc. All rights reserved.

Frame Relay IP RTP Priority

This feature module describes the Frame Relay IP RTP Priority feature. It includes the following sections:

- Feature Overview, page 1
- Supported Platforms, page 2
- Supported Standards, MIBs, and RFCs, page 3
- Prerequisites, page 3
- Configuration Tasks, page 3
- Monitoring and Maintaining Frame Relay IP RTP Priority, page 4
- Configuration Examples, page 4
- Command Reference, page 5

Feature Overview

The Frame Relay IP RTP Priority feature provides a strict priority queueing scheme on a Frame Relay permanent virtual circuit (PVC) for delay-sensitive data such as voice. Voice traffic can be identified by its Real-Time Transport Protocol (RTP) port numbers and classified into a priority queue configured by the **frame-relay ip rtp priority** command. The result of using this feature is that voice is serviced as strict priority in preference to other nonvoice traffic.

This feature extends the functionality offered by the **ip rtp priority** command by supporting Frame Relay PVCs. This feature allows you to specify a range of User Datagram Protocol (UDP) ports whose voice traffic is guaranteed strict priority service over any other queues or classes using the same output interface. Strict priority means that if packets exist in the priority queue, they are dequeued and sent first—that is, before packets in other queues are dequeued.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

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.

Related Features and Technologies

The Frame Relay IP RTP Priority feature is related to the following features:

- IP RTP Priority
- Class-based weighted fair queueing (CBWFQ)
- Priority queueing
- Weighted fair queueing (WFQ)

Related Documents

- *Quality of Service Solutions Configuration Guide*, Cisco IOS Release 12.0
- *Quality of Service Solutions Command Reference*, Cisco IOS Release 12.0
- *Class-Based Weighted Fair Queueing*
- *IP RTP Priority*

Supported Platforms

- Cisco 1003
- Cisco 1004
- Cisco 1005
- Cisco 1600 series
- Cisco 2500 series
- Cisco 2600 series
- Cisco 3600 series
- Cisco 3800 series
- Cisco 4000 series (Cisco 4000, 4000-M, 4500, 4500-M, 4700, 4700-M)
- Cisco 5200 series
- Cisco 7000 series
- Cisco 7200 series
- Cisco 7500 series

This feature runs on the platforms listed. However, it is most useful on voice supported platforms, such as the Cisco 2600 series, Cisco 3600 series, Cisco 7200 series, and Cisco 7500 Route Switch Processor (RSP) series.

Supported Standards, MIBs, and RFCs

Standards

None

MIBs

No new or modified MIBs are supported by this feature.

RFCs

None

Prerequisites

Frame Relay traffic shaping (FRTS) and Frame Relay Fragmentation (FRF.12) must be configured before the Frame Relay IP RTP Priority feature is used.

Configuration Tasks

See the following sections for configuration tasks for the Frame Relay IP RTP Priority feature. Each task in the list is identified as either optional or required.

- Configuring Frame Relay IP RTP Priority (Required)
- Verifying Frame Relay IP RTP Priority (Optional)

Configuring Frame Relay IP RTP Priority

To reserve a strict priority queue on a Frame Relay PVC for a set of RTP packet flows belonging to a range of UDP destination ports, use the following command in map-class configuration mode:

Command	Purpose
`Router(config-map-class)# frame-relay ip rtp priority` *`starting-rtp-port-number port-number-range bandwidth`*	Reserves a strict priority queue for a set of RTP packet flows belonging to a range of UDP destination ports.

Note Because the **frame-relay ip rtp priority** command gives absolute priority over other traffic, it should be used with care. In the event of congestion, if the traffic exceeds the configured bandwidth, then all the excess traffic is dropped.

Verifying Frame Relay IP RTP Priority

To verify the Frame Relay IP RTP Priority feature, use one of the following commands in EXEC mode:

Command	Purpose
`Router# show frame relay pvc`	Displays statistics about PVCs for Frame Relay interfaces.
`Router# show queue interface-type interface-number`	Displays fair queueing configuration and statistics for a particular interface.
`Router# show traffic-shape queue`	Displays information about the elements queued at a particular time at the VC data link connection identifier (DLCI) level.

Monitoring and Maintaining Frame Relay IP RTP Priority

To tune your RTP bandwidth or decrease RTP traffic if the priority queue is experiencing drops, use the following command in EXEC mode:

Command	Purpose
`Router# debug priority`	Displays priority queueing output if packets are dropped from the priority queue.

Configuration Examples

This section provides the following configuration examples:

- Frame Relay IP RTP Priority Configuration Example

Frame Relay IP RTP Priority Configuration Example

The following example first configures the Frame Relay map class called voip and then applies the map class to PVC 100 to provide strict priority service to matching RTP packets:

```
map-class frame-relay voip
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay be 600
 frame-relay mincir 256000
 no frame-relay adaptive-shaping
 frame-relay fair-queue
 frame-relay fragment 250
 frame-relay ip rtp priority 16384 16380 210

interface Serial5/0
 ip address 10.10.10.10 255.0.0.0
 no ip directed-broadcast
 encapsulation frame-relay
 no ip mroute-cache
 load-interval 30
 clockrate 1007616
 frame-relay traffic-shaping
```

```
frame-relay interface-dlci 100
 class voip
frame-relay ip rtp header-compression
frame-relay intf-type dce
```

In this example, RTP packets on PVC 100 with UDP ports in the range 16384 to 32764 will be matched and given strict priority service.

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **frame-relay ip rtp priority**

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

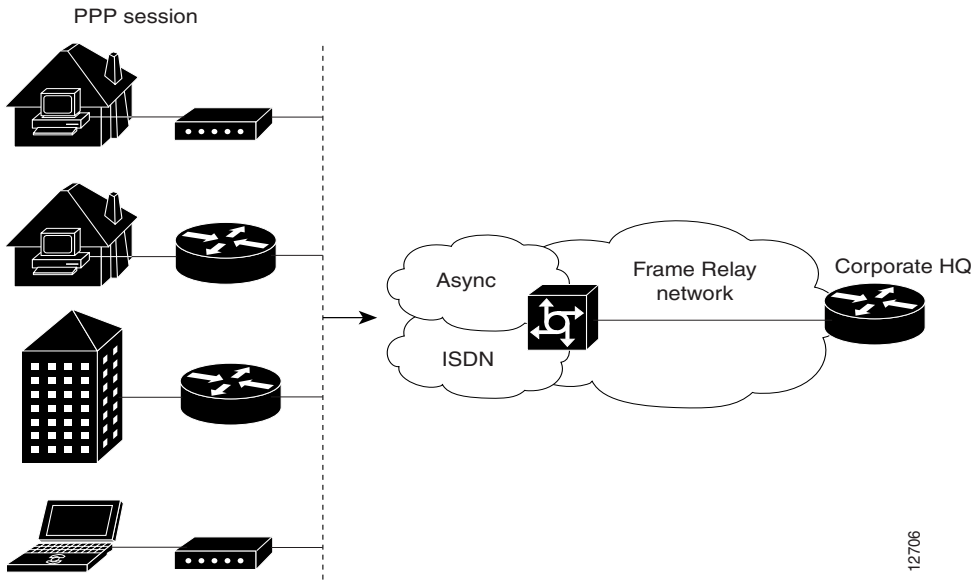
© 2007 Cisco Systems, Inc. All rights reserved.

PPP over Frame Relay

Feature Summary

The PPP over Frame Relay feature allows a router to establish end-to-end Point-to-Point Protocol (PPP) sessions over Frame Relay. IP datagrams are transported over the PPP link using RFC 1973 compliant Frame Relay framing. This feature is useful for remote users running PPP to access their Frame Relay corporate networks as shown in Figure 1. Figure 2 shows a connectivity scenario using the Cisco 90i D4 channel card, which is capable of supporting Integrated Services Digital Network (ISDN) Digital Service Loop (DSL), PPP, or Frame Relay, which connects to an Internet Service Provider (ISP) or corporate network.

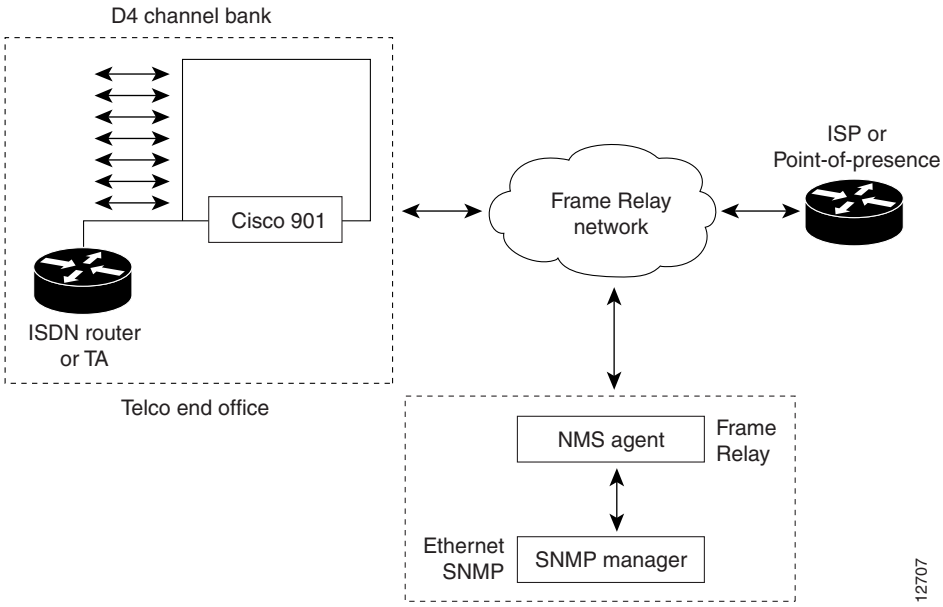
Figure 1 PPP Over Frame Relay Scenario



Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

Figure 2 PPP over Frame Relay Using the Cisco 90i D4 Channel Unit



Benefits

PPP over Frame Relay provides the following benefits:

- Allows end-to-end PPP sessions over Frame Relay.
- Supports the 90i IDSL Channel Unit that supports both Frame Relay and Point-to-Point Protocol (PPP) on an ISDN DSL.

List of Terms

data-link connection identifier (DLCI)— A value that specifies a PVC or SVC in a Frame Relay network. In the basic Frame Relay specification, DLCIs are locally significant (connected devices might use different values to specify the same connection). In the LMI extended specification, DLCIs are globally significant (DLCIs specify individual end devices).

Integrated Services Digital Network (ISDN)—Communication protocols offered by telephone companies that permit telephone networks to carry data, voice, and other source traffic.

Link Control Protocol (LCP)—A protocol that establishes, configures, and tests data link connections used by PPP.

permanent virtual circuit (PVC)—Virtual circuit that is permanently established. PVCs save bandwidth associated with circuit establishment and tear down in situations where certain virtual circuits must exist all the time.

Point-to-Point Protocol (PPP)—A protocol that encapsulates network layer protocol information over point-to-point links. The RFC for PPP is RFC 1661.

virtual circuit (VC)—A logical circuit created to ensure reliable communication between two network devices. A virtual circuit can be either permanent (a PVC) or switched (an SVC). Virtual circuits are used in Frame Relay and X.25.

Restrictions

The following restrictions apply when using PPP over Frame Relay:

- Only Frame Relay PVCs are supported.
- Only the Internet Protocol (IP) is supported.

Platforms

This feature is supported on these platforms:

- Cisco 1600 series
- Cisco 2500 series
- Cisco 2600 series
- Cisco 3600 series
- Cisco 4000 series (Cisco 4000, 4000-M, 4500, 4500-M, 4700, 4700-M)
- Cisco 7200 series
- Cisco 7500 series

Prerequisites

Before you can configure PPP over Frame Relay, Frame Relay must be enabled on the router using the **encapsulation frame-relay** command.

Supported MIBs and RFCs

This feature supports the RFC 1973.

No new MIBs are supported by this feature.

Functional Description

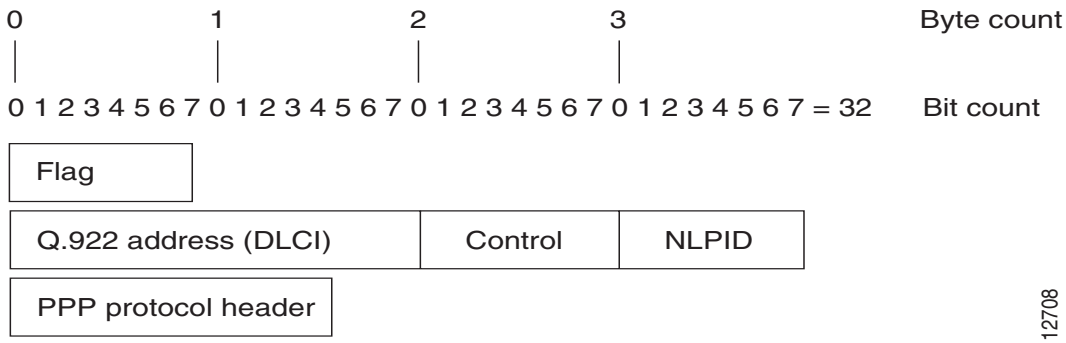
PPP over Frame Relay is compliant to the functionality and encapsulation specifications as outlined in RFC 1973. The frame format is shown in Figure 3.

A PPP connection is established over the Frame Relay permanent virtual circuit (PVC). The PPP session does not occur unless the associated Frame Relay PVC is in an “active” state. The Frame Relay VC can coexist with other circuits using different Frame Relay encapsulation methods, such as RFC 1490 and Cisco proprietary, over the same Frame Relay link. There can be multiple PPP-in-Frame Relay circuits existing on one Frame Relay link.

One PPP connection resides on one virtual access interface, which is internally created from a virtual template interface. A virtual access interface is cloned from a virtual template interface. The virtual access interfaces is coexistent with the creation of the Frame Relay circuit when the corresponding DLCI is configured. One virtual template interface, containing all the necessary PPP and network protocol information is shared by multiple virtual access interfaces. Hardware compression and fancy queuing

algorithms, such as weighted fair queuing, custom queuing, and priority queuing, are not applied to virtual access interfaces. Once a Frame Relay circuit is established using PPP over Frame Relay, all incoming and outgoing packets on this circuit are under RFC 1973 PPP-in-Frame-Relay encapsulation compliance until this DLCI is removed from the configuration. Refer to the Cisco IOS Release 11.3 *Wide Area Configuration Guide* and *Command Reference documents* for information about Frame Relay configuration options.

Figure 3 PPP over Frame Relay Frame Format



The breakdown of the Frame Relay frame format components is listed in Table 1.

Table 1 PPP Frame Relay Format Descriptions

Field	Description
Flag	A single byte that indicates the beginning or end of a frame.
Address	A two byte field that indicates the logical connection that maps to the physical channel; the DLCI.
Control	A single byte that calls for transmission of user data. PPP over Frame Relay uses a value of 0X03, which indicates the frame is an unnumbered information (UI) frame.
NLPID	Network layer protocol ID, which is a single byte that uniquely identifies a PPP packet to Frame Relay.
PPP protocol	Identifies the PPP packet type.

Configuration Task

The only task required to implement PPP over Frame Relay is to configure the interface with the locally terminated PVC and the associated virtual template for PPP and IP, as described in the following section.

Enable PPP over Frame Relay

After you configure the Cisco router or access server for Frame Relay encapsulation, you must configure the physical interface with the PVC and apply a virtual template with PPP encapsulation to the DLCI that it applies to. To configure the physical interface that will carry the PPP session and link it to the appropriate virtual template interface, perform the following task in interface configuration mode:

Task	Command
Define the PVC and map it to the virtual template.	frame-relay interface-dlci *dlci* [ppp *virtual-template-name-string*]

Configuration Examples

This section provides the following examples:

- PPP over Frame Relay DTE Example
- PPP over Frame Relay DCE Example

PPP over Frame Relay DTE Example

The following example configures a router as a data terminating equipment (DTE) device for PPP over Frame Relay. Subinterface 2.1 contains the necessary DLCI and virtual template information. The virtual template interface (interface virtual-template 1) contains the PPP information that is applied to the PPP session associated with DLCI 32 on serial subinterface 2.1. Refer to the Cisco IOS Release 11.3 *Wide-Area Configuration Guide* and *Wide-Area Networking Command Reference* for information about Frame Relay configuration options.

```
interface serial 2
 no ip address
 encapsulation frame-relay
 frame-relay lmi-type ansi
!
interface serial 2.1 point-to-point
 frame-relay interface-dlci 32 ppp virtual-template1
!
interface Virtual-Template1
 ip unnumbered ethernet 0
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulaiton; therefore, **encapsulation ppp** will not show up when viewing the router's configuration.

PPP over Frame Relay DCE Example

The following example configures a router to act as a data communications equipment (DCE) device. Typically, a router is configured for a DCE if connecting directly to another router or if connected to a 90i D4 channel unit, which is connected to a telco channel bank. The three commands required for this type of configuration are the **frame-relay switching**, **frame-relay intf-type dce**, and **frame-relay route** commands. In this configuration

```
frame-relay switching
!
interface Serial2/0:0
 no ip address
 encapsulation frame-relay IETF
 frame-relay lmi-type ansi
 frame-relay intf-type dce
 frame-relay route 31 interface Serial1/2 100
 frame-relay interface-dlci 32 ppp Virtual-Template1
!
interface Serial2/0:0.2 point-to-point
 no ip address
 frame-relay interface-dlci 40 ppp Virtual-Template2
!
interface Virtual-Template1
 ip unnumbered Ethernet0/0
 peer default ip address pool default
 ppp authentication chap pap
 !
interface Virtual-Template2
 ip address 100.1.1.2 255.255.255.0
 ppp authentication chap pap
```

Note By default, the encapsulation type for a virtual template interface is PPP encapsulaiton; therefore, **encapsulation ppp** will not show up when viewing the router's configuration.

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **frame-relay interface-dlci**
- **show frame-relay pvc**
- **debug frame-relay ppp**

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream,

Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Multilink Frame Relay (FRF.16.1)

First Published: May 14, 2001
Last Updated: August 08, 2007

The Multilink Frame Relay (FRF.16.1) feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1). This feature provides a cost-effective way to increase 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 User-to-Network Interfaces (UNI) and Network-to-Network Interfaces (NNI) in Frame Relay networks.

History for the Multilink Frame Relay (FRF.16.1) Feature

Release	Modification
12.0(17)S	This feature was introduced on the Cisco 12000 series.
12.2(8)T	This feature was integrated into Cisco IOS Release 12.2(8)T.
12.2(14)S	This feature was integrated into Cisco IOS Release 12.2(14)S.
12.3(9)	Frame Relay fragmentation (FRF.12) support was integrated into Cisco IOS Release 12.3(9).
12.3(11)T	Frame Relay fragmentation (FRF.12) support was integrated into Cisco IOS Release 12.3(11)T.
12.0(30)S	Variable bandwidth class support was integrated into Cisco IOS Release 12.0(30)S.
12.4(2)T	Variable bandwidth class support was integrated into Cisco IOS Release 12.4(2)T.
12.2(30)S	Frame Relay fragmentation (FRF.12) support was integrated into Cisco IOS Release 12.2(30)S.
12.2(28)SB	This feature was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This feature was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	Support was added for the Cisco 10000 Series Router.
12.2(33)SXH	This feature was integrated into Cisco IOS Release 12.2(33)SXH.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

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

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

Contents

- Prerequisites for Multilink Frame Relay (FRF.16.1), page 2
- Restrictions for Multilink Frame Relay (FRF.16.1), page 2
- Information About Multilink Frame Relay (FRF.16.1), page 2
- How to Enable Multilink Frame Relay (FRF.16.1), page 4
- Configuration Examples for Multilink Frame Relay (FRF.16.1), page 11
- Additional References, page 12
- Command Reference, page 13
- Glossary, page 14

Prerequisites for Multilink Frame Relay (FRF.16.1)

- Multilink Frame Relay must be configured on the peer device.

Restrictions for Multilink Frame Relay (FRF.16.1)

- ISDN interfaces and any type of virtual interface cannot be a bundle link.
- Frame Relay fragmentation (FRF.12) is not supported in Cisco IOS releases 12.0(17)S, 12.2(8)T, and 12.2(14)S.
- The multilink Frame Relay MIB (RFC 3020) is not supported.
- FRF.9 hardware compression over multilink Frame Relay is not supported.

Information About Multilink Frame Relay (FRF.16.1)

To enable multilink Frame Relay (FRF.16.1) variable bandwidth class support, you should understand the following concepts:

- Benefits of Multilink Frame Relay (FRF.16.1), page 2
- Link Integrity Protocol Control Messages, page 3
- Variable Bandwidth Class Support, page 3
- Load Balancing with Multilink Frame Relay (FRF.16.1), page 4

Benefits of Multilink Frame Relay (FRF.16.1)

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. Multilink Frame Relay provides a cost-effective solution to this problem by allowing multiple T1 lines to be aggregated into a single bundle of bandwidth.

Greater Service Resilience When Links Fail

Greater service resilience is provided when multiple physical interfaces are provisioned as a single bundle. When a link fails, the bundle continues to support the Frame Relay service by transmitting across the remaining bundle links.

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). For a bundle link to be brought up, each end of the link must complete an exchange of ADD_LINK and ADD_LINK_ACK messages. To maintain the link, both ends periodically initiate the exchange of HELLO and HELLO_ACK messages. This exchange of hello messages and acknowledgments serves 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 (nonoperational).

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 protocol status is considered up when the Frame Relay data-link layer at the local router and peer device is synchronized 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

Multilink Frame Relay (FRF.16.1) variable bandwidth class support allows you to specify the criterion used to activate or deactivate a Frame Relay bundle. Consistent with the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1), bandwidth classes A (single link), B (all links), and C (threshold) are supported.

Class A (Single Link)

The Frame Relay bundle is provisioned when one or more bundle links indicate by issuing a BL_ACTIVATE message that 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 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.

Class B (All Links)

The Frame Relay bundle is provisioned when all bundle links indicate by issuing a BL_ACTIVATE message that 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)

The Frame Relay bundle is provisioned when the minimum number of links in the configured bundle issue a BL_ACTIVATE message. When this occurs, the bundle emulates a physical link by issuing a PH_ACTIVATE message to the data-link layer.

When the number of bundle links that are 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 with Multilink Frame Relay (FRF.16.1)

Multilink Frame Relay provides load balancing across the bundle links within a bundle. If a bundle link chosen for transmission happens to be busy transmitting a long packet, the load-balancing mechanism can try another link, thus solving the problems seen when delay-sensitive packets have to wait.

How to Enable Multilink Frame Relay (FRF.16.1)

This section contains the following procedures:

- Configuring a Multilink Frame Relay Bundle, page 4
- Configuring a Multilink Frame Relay Bundle Link, page 7
- Monitoring and Maintaining Multilink Frame Relay (FRF.16.1), page 9

Configuring a Multilink Frame Relay Bundle

To configure the bundle interface for multilink Frame Relay, perform the steps in this section.

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**
6. **frame-relay multilink bid** *name*
7. **frame-relay multilink output-threshold** *bytes*
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

	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 mfr mfr1`	Configures a multilink Frame Relay bundle interface.

Step 4	**frame-relay multilink bandwidth-class** [**a** | **b** | **c** [*threshold*]]
	
	Example:
	`Router(config-if)# frame-relay multilink bandwidth-class a`
	or
	`Router(config-if)# frame-relay multilink bandwidth-class b`
	or
	`Router(config-if)# frame-relay multilink bandwidth-class c 3`
		(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 will deactivate when all bundle links are down (default).
		• Class B (all links)—The bundle will activate when all bundle links are up and will 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 will 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).

	Command or Action	Purpose
Step 5	**frame-relay intf-type dce** **Example:** `Router(config-if)# frame-relay intf-type dce`	Configures a device to function as the data circuit-terminating equipment (DCE). • Only one end of a link should be configured as the DCE. The other end will function as the data terminal equipment (DTE), which is the default setting. • This command can be used only if Frame Relay switching has been enabled by entering the **frame-relay switching** command in global configuration mode.
Step 6	**frame-relay multilink bid** *name* **Example:** `Router(config-if)# frame-relay multilink bid router1`	(Optional) Assigns a bundle identification name to a multilink Frame Relay 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.
Step 7	**frame-relay multilink output-threshold** *bytes* **Example:** `Router(config-if)# frame-relay multilink output-threshold 500`	(Optional) Configures the number of bytes that a bundle link will transmit before the load-balancing mechanism causes transmission to roll over to the next available link. • When configured on the bundle interface, this command applies to all bundle links in the bundle.
Step 8	**interface mfr** *interface-number.subinterface-number* **point-to-point** **Example:** `Router(config-if)# interface mfr1.1 point-to-point`	Configures a point-to-point multilink Frame Relay subinterface.
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-subif)# ip address 10.0.1.1 255.255.255.0`	Configures the IP address for the subinterface.
Step 10	**frame-relay interface-dlci** *dlci* **Example:** `Router(config-subif)# frame-relay interface-dlci 100`	Assigns a data-link connection identifier (DLCI) to a Frame Relay subinterface.

	Command or Action	Purpose
Step 11	**end** **Example:** `Router(config-subif)# end`	Ends the configuration session 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 a Multilink Frame Relay Bundle Link

To configure a bundle link interface for multilink Frame Relay, perform the steps in this section.

Tip To minimize latency that results from the arrival order of packets, we recommend bundling physical links of the same line speed in one bundle.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface serial** *number*
4. **encapsulation frame-relay mfr** *number* [*name*]
5. **frame-relay multilink output-threshold** *bytes*
6. **frame-relay multilink lid** *name*
7. **frame-relay multilink hello** *seconds*
8. **frame-relay multilink ack** *seconds*
9. **frame-relay multilink retry** *number*
10. **end**
11. **show frame-relay multilink**

DETAILED STEPS

	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	**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 a multilink Frame Relay bundle link and associates the link with a bundle.
Step 5	**frame-relay multilink output-threshold** *bytes* **Example:** `Router(config-if)# frame-relay multilink output-threshold 500`	(Optional) Configures the number of bytes that a bundle link will transmit before the load-balancing mechanism causes transmission to roll over to the next available link.
Step 6	**frame-relay multilink lid** *name* **Example:** `Router(config-if)# frame-relay multilink lid first-link`	(Optional) Assigns a bundle link identification name with a multilink Frame Relay bundle link. • The bundle link identification (LID) 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.
Step 7	**frame-relay multilink hello** *seconds* **Example:** `Router(config-if)# frame-relay multilink hello 9`	(Optional) Configures the interval at which a bundle link will send out hello messages. • The default value is 10 seconds.
Step 8	**frame-relay multilink ack** *seconds* **Example:** `Router(config-if)# frame-relay multilink ack 6`	(Optional) Configures the number of seconds that a bundle link will wait for a hello message acknowledgment before resending the hello message. • The default value is 4 seconds.
Step 9	**frame-relay multilink retry** *number* **Example:** `Router(config-if)# frame-relay multilink retry 3`	(Optional) Configures the maximum number of times that a bundle link will resend a hello message while waiting for an acknowledgment. • The default value is 2 tries.
Step 10	**end** **Example:** `Router(config-if)# end`	Ends the configuration session and returns to privileged EXEC mode.
Step 11	**show frame-relay multilink** **Example:** `Router# show frame-relay multilink`	(Optional) Displays the current Frame Relay multilink configuration.

Monitoring and Maintaining Multilink Frame Relay (FRF.16.1)

To monitor and maintain multilink Frame Relay, perform the steps in this section.

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

	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 multilink Frame Relay 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 multilink Frame Relay 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 example shows output for the **show frame-relay multilink** command. Because a particular bundle or bundle link is not specified, information for all bundles and bundle links is displayed:

```
Router# show frame-relay multilink

Bundle: MFR0, state up, class A, no fragmentation
 ID: Bundle-Dallas
 Serial5/1, state up/up, ID: BL-Dallas-1
 Serial5/3, state up/add-sent, ID: BL-Dallas-3

Bundle: MFR1, state down, class B, fragmentation
 ID: Bundle-NewYork#1
 Serial3/0, state up/up, ID: BL-NewYork-1
 Serial3/2, state admin-down/idle, ID: BL-NewYork-2
```

The following example shows output for the **show frame-relay multilink** command when a Frame Relay bundle is configured as bandwidth class C (threshold):

```
Router# show frame-relay multilink

Bundle: MFR0, state down, class C (threshold 3), no fragmentation
 ID: Bundle-Dallas
 Serial5/1, state up/up, ID: BL-Dallas-1
 Serial5/3, state up/add-sent, ID: BL-Dallas-3
```

The following example shows output for the **show frame-relay multilink** command when the **serial** *number* keyword and argument are specified. It displays information about the specified bundle link:

```
Router# show frame-relay multilink serial 3/2

 Bundle links :
 Serial3/2, HW state :down, Protocol state :Down_idle, LID :Serial3/2
 Bundle interface = MFR0,  BID = MFR0
```

The following examples show output for the **show frame-relay multilink** command when the **serial** *number* keyword and argument and the **detailed** option are specified. Detailed information about the specified bundle links is displayed. The first example shows a bundle link in the “idle” state. The second example shows a bundle link in the “up” state:

```
Router# show frame-relay multilink serial 3 detail

Bundle links:

  Serial3, HW state = up, link state = Idle, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 0 ms
    Statistics:
    Add_link sent = 0, Add_link rcv'd = 10,
    Add_link ack sent = 0, Add_link ack rcv'd = 0,
    Add_link rej sent = 10, 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

Router# show frame-relay multilink serial 3 detail

Bundle links:

  Serial3, HW state = up, link state = Up, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 4 ms
    Statistics:
    Add_link sent = 1, Add_link rcv'd = 20,
    Add_link ack sent = 1, Add_link ack rcv'd = 1,
    Add_link rej sent = 19, 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 = 1,
    Hello_ack sent = 1, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
```

Configuration Examples for Multilink Frame Relay (FRF.16.1)

This section provides the following configuration examples:

- Configuring Multilink Frame Relay: Example, page 11
- Configuring Variable Bandwidth Class Support: Example, page 11

Configuring Multilink Frame Relay: Example

The following example shows the configuration of bundle "MFR1." Serial interfaces 5/0 and 6/0 are configured as bundle links:

```
interface MFR1
 no ip address
 mls qos trust dscp
 frame-relay intf-type dce
 frame-relay multilink bid router1
!
interface MFR1.1 point-to-point
 ip address 10.0.1.1 255.255.255.0
 ip pim sparse-mode
 mls qos trust dscp
 frame-relay interface-dlci 100

interface Serial5/0
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 9
 frame-relay multilink retry 3

interface Serial6/0
 encapsulation frame-relay MFR1
 frame-relay multilink ack 4
```

Configuring Variable Bandwidth Class Support: Example

The following example configures the Frame Relay bundle "MFR1" to use the class B (all links) criterion to be activated or deactivated:

```
interface MFR1
 ip address 10.1.1.1 255.255.255.0
 frame-relay interface-dlci 100
 frame-relay multilink bandwidth-class b
```

Additional References

The following sections provide references related to multilink Frame Relay (FRF.16.1).

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

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **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 output-threshold**
- **frame-relay multilink retry**
- **interface mfr**
- **show frame-relay multilink**

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.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Distributed Multilink Frame Relay (FRF.16)

Feature History

Release	Modification
12.0(24)S	The Distributed Multilink Frame Relay feature was introduced. This feature introduced Multilink Frame Relay (FRF.16) on VIP-enabled Cisco 7500 series routers.
12.3(4)T	This feature on VIP-enabled Cisco 7500 series routers was integrated into Cisco IOS Release 12.3(4)T.

This document describes the Distributed Multilink Frame Relay (dMFR) feature in Cisco IOS Release 12.0(24)S and Cisco IOS Release 12.3(4)T. The dMFR feature introduces MFR on VIP-enabled Cisco 7500 series routers. For information on MFR on other platforms, see the *Multilink Frame Relay (FRF.16)* document.

This document includes the following sections:

- Feature Overview, page 1
- Supported Platforms, page 4
- Supported Standards, MIBs, and RFCs, page 5
- Prerequisites, page 5
- Configuration Tasks, page 5
- Monitoring and Maintaining Distributed Multilink Frame Relay, page 8
- Configuration Examples, page 9
- Command Reference, page 10
- Glossary, page 10

Feature Overview

The Distributed Multilink Frame Relay feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16) to VIP-enabled Cisco 7500 series routers. The Distributed Multilink Frame Relay feature provides a cost-effective way to increase

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

bandwidth for particular applications by enabling multiple serial links to be aggregated into a single bundle of bandwidth. Multilink Frame Relay is supported on User-to-Network Interfaces (UNI) and Network-to-Network Interfaces (NNI) in Frame Relay networks.

Multilink Frame Relay Bundles and Bundle Links

The Multilink Frame Relay feature enables you to create a virtual interface called a *bundle* or *bundle interface*. The bundle interface emulates a physical interface for the transport of frames. The Frame Relay data link runs on the bundle interface, and Frame Relay virtual circuits are built upon it.

The bundle is made up of multiple serial links, called *bundle links*. Each bundle link within a bundle corresponds to a physical interface. Bundle links are invisible to the Frame Relay data-link layer, so Frame Relay functionality cannot be configured on these interfaces. Regular Frame Relay functionality that you want to apply to these links must be configured on the bundle interface. Bundle links are visible to peer devices. The local router and peer devices exchange link integrity protocol control messages to determine which bundle links are operational and to synchronize which bundle links should be associated with which bundles.

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.

Load Balancing

Distributed Multilink Frame Relay provides load balancing across the bundle links within a bundle. If a bundle link chosen for transmission happens to be busy transmitting a long packet, the load balancing mechanism can try another link, thus solving the problems seen when delay-sensitive packets have to wait.

Benefits

Flexible Pool of Bandwidth

By combining multiple physical interfaces into a bundle, you can design a Frame Relay interface with 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. Distributed Multilink Frame Relay provides a cost-effective solution to this problem by allowing multiple T1 lines to be aggregated into a single bundle of bandwidth.

Greater Service Resilience When Links Fail

Greater service resilience is provided when multiple physical interfaces are provisioned as a single bundle. When a link fails, the bundle continues to support the Frame Relay service by transmitting across the remaining bundle links.

Restrictions

The Distributed Multilink Frame Relay feature has the following restrictions:

- ISDN interfaces and any type of virtual interfaces cannot be a bundle link.
- Distributed CEF is limited to IP traffic only; all other protocols are processed using the Route Switch Processor (RSP).
- Frame Relay fragmentation (FRF.12) is not supported.
- Multilink Frame Relay MIB (RFC 3020) is not supported.
- FRF.9 hardware compression over multilink Frame Relay is not supported.
- Each link in a bundle must reside on the same port adapter and all links in a bundle must have identical configurations. The same bandwidth for each link in the bundle is also recommended because bundles that contain individual links with different bandwidths process packets less efficiently.
- Fragmentation is not supported on the transmitting interface when used in conjunction with Distributed Multilink Frame Relay.
- The maximum differential delay is 50 ms.
- All T1 lines can be combined into one bundle. A VIP2-50 with 4 or 8 MB of SRAM supports up to 16 T1 bundles per VIP and a VIP2-50 with 2 MB of SRAM supports up to 8 T1 bundles per VIP. A maximum of 40 T1 bundles per VIP can be used on a VIP4-80.
- All E1 lines can be combined into one bundle. A VIP2-50 with 4 or 8 MB of SRAM supports up to 12 E1 bundles per VIP and a VIP2-50 with 2 MB or SRAM supports up to 8 E1 bundles per VIP. A maximum of 32 E1 bundles per VIP can be used on a VIP4-80.

Related Documents

- *Cisco IOS Wide-Area Networking Configuration Guide*, Release 12.3
- *Cisco IOS Wide-Area Networking Command Reference*, Release 12.3
- *Multilink Frame Relay (FRF.16)* (provides information on nondistributed Multilink Frame Relay)
- Cisco IOS Release 12.3 T command references (information on 12.0 S and 12.3 T commands)

Supported Platforms

- Cisco 7500 series router with a VIP2-50 or greater

  This feature works on the following port adapters:

  - PA-MC-T3
  - PA-MC-2T3+
  - PA-MC-E3
  - PA-MC-2E1
  - PA-MC-2T1
  - PA-MC-4T1
  - PA-MC-8T1
  - PA-MC-8E1
  - PA-MC-STM-1
  - PA-MC-8TE1+
  - PA-4T+
  - PA-8T

Determining Platform Support Through Cisco Feature Navigator

Cisco IOS software is packaged in feature sets that are supported on specific platforms. To get updated information regarding platform support for this feature, access Cisco Feature Navigator. Cisco Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Cisco Feature Navigator is a web-based tool that enables you to determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image. You can search by feature or release. Under the release section, you can compare releases side by side to display both the features unique to each software release and the features in common.

To access Cisco Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions found at this URL:

http://www.cisco.com/register

Cisco Feature Navigator is updated regularly when major Cisco IOS software releases and technology releases occur. For the most current information, go to the Cisco Feature Navigator home page at the following URL:

http://www.cisco.com/go/fn

Availability of Cisco IOS Software Images

Platform support for particular Cisco IOS software releases is dependent on the availability of the software images for those platforms. Software images for some platforms may be deferred, delayed, or changed without prior notice. For updated information about platform support and availability of software images for each Cisco IOS software release, refer to the online release notes or, if supported, Cisco Feature Navigator.

Supported Standards, MIBs, and RFCs

Standards

Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16.1), July 2001

MIBs

No new or modified MIBs are supported 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://tools.cisco.com/ITDIT/MIBS/servlet/index

If Cisco MIB Locator does not support the MIB information that you need, you can also obtain a list of supported MIBs and download MIBs from the Cisco MIBs page at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

To access Cisco MIB Locator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions found at this URL:

http://www.cisco.com/register

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

- Distributed Cisco Express Forwarding (CEF) must be enabled globally.
- Multilink Frame Relay must be configured on the peer device.
- The multilink Frame Relay peer device must not send frames that require assembly.

Configuration Tasks

See the following sections for configuration tasks for the Distributed Multilink Frame Relay feature. Each task in the list is identified as either optional or required.

- Configuring a Multilink Frame Relay Bundle (required)
- Configuring a Multilink Frame Relay Bundle Link (required)
- Verifying Multilink Frame Relay (optional)

Configuring a Multilink Frame Relay Bundle

To configure the bundle interface for Distributed Multilink Frame Relay, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface mfr number`	Configures a multilink Frame Relay bundle interface.
Step 2	`Router(config-if)# frame-relay multilink bid name`	(Optional) Assigns a bundle identification name to a multilink Frame Relay bundle. **Note** 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 **shut** and **no shut** commands in interface configuration mode.

Configuring a Multilink Frame Relay Bundle Link

To configure a bundle link interface for multilink Frame Relay, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface serial number`	Selects a physical interface and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation frame-relay mfr number [name]`	Creates a multilink Frame Relay bundle link and associates the link with a bundle. **Tip** To minimize latency that results from the arrival order of packets, we recommend bundling physical links of the same line speed in one bundle.
Step 3	`Router(config-if)# frame-relay multilink lid name`	(Optional) Assigns a bundle link identification name to a multilink Frame Relay bundle link. **Note** The bundle link identification (LID) 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 **shut** and **no shut** commands in interface configuration mode.
Step 4	`Router(config-if)# frame-relay multilink hello seconds`	(Optional) Configures the interval at which a bundle link will send out hello messages. The default value is 10 seconds.

	Command	Purpose
Step 5	`Router(config-if)# frame-relay multilink ack` *seconds*	(Optional) Configures the number of seconds that a bundle link will wait for a hello message acknowledgment before resending the hello message. The default value is 4 seconds.
Step 6	`Router(config-if)# frame-relay multilink retry` *number*	(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.

Verifying Multilink Frame Relay

To verify multilink Frame Relay configuration, use the **show frame-relay multilink** command.

The following example shows output for the **show frame-relay multilink** command. Because a particular bundle or bundle link is not specified, information for all bundles and bundle links is displayed.

```
Router# show frame-relay multilink

Bundle: MFR0, state up, class A, no fragmentation
 ID: Bundle-Dallas
 Serial5/1, state up/up, ID: BL-Dallas-1
 Serial5/3, state up/add-sent, ID: BL-Dallas-3

Bundle: MFR1, state down, class B, fragmentation
 ID: Bundle-NewYork#1
 Serial3/0, state up/up, ID: BL-NewYork-1
 Serial3/2, state admin-down/idle, ID: BL-NewYork-2
```

The following example shows output for the **show frame-relay multilink** command with the **serial** *number* option. It displays information about the specified bundle link.

```
Router# show frame-relay multilink serial3/2

 Bundle links :
 Serial3/2, HW state :Administratively down, Protocol state :Down_idle, LID :Serial3/2
 Bundle interface = MFR0,  BID = MFR0
```

The following examples show output for the **show frame-relay multilink** command with the **serial** *number* and **detail** options. Detailed information about the specified bundle links is displayed. The first example shows a bundle link in the “idle” state. The second example shows a bundle link in the “up” state.

```
Router# show frame-relay multilink serial3 detail
 Bundle links:

  Serial3, HW state = up, link state = Idle, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 0 ms
    Statistics:
    Add_link sent = 0, Add_link rcv'd = 10,
    Add_link ack sent = 0, Add_link ack rcv'd = 0,
    Add_link rej sent = 10, 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

Router# show frame-relay multilink serial3 detail
 Bundle links:

  Serial3, HW state = up, link state = Up, LID = Serial3
  Bundle interface = MFR0,  BID = MFR0
    Cause code = none, Ack timer = 4, Hello timer = 10,
    Max retry count = 2, Current count = 0,
    Peer LID = Serial5/3, RTT = 4 ms
    Statistics:
    Add_link sent = 1, Add_link rcv'd = 20,
    Add_link ack sent = 1, Add_link ack rcv'd = 1,
    Add_link rej sent = 19, 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 = 1,
    Hello_ack sent = 1, Hello_ack rcv'd = 0,
    outgoing pak dropped = 0, incoming pak dropped = 0
```

Monitoring and Maintaining Distributed Multilink Frame Relay

To monitor and maintain Distributed Multilink Frame Relay, use one or more of the following commands in privileged EXEC mode:

Command	Purpose
Router# **debug frame-relay multilink** [**control** [**mfr** *number* | **serial** *number*]]	Displays debug messages for multilink Frame Relay bundles and bundle links.
Router# **show frame-relay multilink** [**mfr** *number* | **serial** *number*] [**detailed**]	Displays configuration information and statistics about multilink Frame Relay bundles and bundle links.
Router# **show interfaces mfr** *number*	Displays information and packet statistics for the bundle interface.

Configuration Examples

This section provides a Distributed Multilink Frame Relay configuration example.

Distributed Multilink Frame Relay Configuration Example

The following example shows the configuration of bundle “MFR1”. Serial interfaces 5/0 and 6/0 are configured as bundle links.

```
interface MFR1
 frame-relay multilink bid first-bundle
 frame-relay traffic-shaping
 frame-relay class ocean

interface MFR1.1 point-to-point
 ip address 1.1.1.1 255.255.255.0
 frame-relay interface-dlci 100

interface Serial5/0
 encapsulation frame-relay MFR1
 frame-relay multilink lid first-link
 frame-relay multilink hello 9
 frame-relay multilink retry 3

interface Serial6/0
 encapsulation frame-relay MFR1
 frame-relay multilink ack 4
```

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **debug frame-relay multilink**
- **encapsulation frame-relay mfr**
- **frame-relay multilink ack**
- **frame-relay multilink bid**
- **frame-relay multilink hello**
- **frame-relay multilink lid**
- **frame-relay multilink retry**
- **interface mfr**
- **show frame-relay multilink**

Glossary

BID—bundle identification. BID is the name used to identify the bundle. The BID can be assigned or the default can be used.

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. Value that identifies a permanent virtual circuit (PVC) in Frame Relay network.

HELLO message—Message that notifies a peer endpoint that the local endpoint is in the operational state (up).

HELLO_ACK—Message that notifies a peer endpoint that a hello message has been received.

LID—link identification. 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. 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.

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.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Layer 2 Tunnel Protocol Version 3

First Published: December 15, 2001
Last Updated: November 20, 2009

The Layer 2 Tunnel Protocol Version 3 feature expands Cisco support of the Layer 2 Tunnel Protocol Version 3 (L2TPv3). L2TPv3 is an Internet Engineering Task Force (IETF) l2tpext working group draft that provides several enhancements to L2TP for the capability to tunnel any Layer 2 payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network using Layer 2 virtual private networks (VPNs). Benefits of this feature include the following:

- L2TPv3 simplifies deployment of VPNs.
- L2TPv3 does not require Multiprotocol Label Switching (MPLS).
- L2TPv3 supports Layer 2 tunneling over IP for any payload.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Layer 2 Tunnel Protocol Version 3" section on page 112.

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

Contents

- Prerequisites for Layer 2 Tunnel Protocol Version 3, page 2
- Restrictions for Layer 2 Tunnel Protocol Version 3, page 2
- Information About Layer 2 Tunnel Protocol Version 3, page 29
- How to Configure Layer 2 Tunnel Protocol Version 3, page 47
- Configuration Examples for Layer 2 Tunnel Protocol Version 3, page 89

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Additional References, page 109
- Feature Information for Layer 2 Tunnel Protocol Version 3, page 112
- Glossary, page 116

Prerequisites for Layer 2 Tunnel Protocol Version 3

- Before you configure an xconnect attachment circuit for a customer edge (CE) device (see the section "Configuring the Xconnect Attachment Circuit"), the CEF feature must be enabled. To enable CEF 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 provider edge (PE) device at the other end of an L2TPv3 control channel.
- To enable Simple Network Management Protocol (SNMP) notifications of L2TP session up and down events, enter the **snmp-server enable traps l2tun session** command before configuring L2TPv3.

Restrictions for Layer 2 Tunnel Protocol Version 3

The following subsections contain information on restrictions:

- Supported Port Adapters for the Cisco 7200 Series and Cisco 7500 Series Routers
- General L2TPv3 Restrictions
- Cisco 7200 Series-Specific Restrictions
- Cisco 7301 Specific Restrictions
- Cisco 7304 Specific Restrictions
- Cisco 7500 Series-Specific Restrictions
- Supported Shared Port Adapters for the Cisco 7600 Series Router
- Cisco 7600 Series-Specific Restrictions
- Cisco 10720-Specific Restrictions
- Cisco 12000 Series-Specific Restrictions
- Frame Relay-Specific Restrictions
- VLAN-Specific Restrictions
- ATM VP Mode Single Cell Relay over L2TPv3 Restrictions
- ATM AAL5 SDU over L2TPv3 and Single Cell Relay VC Mode over L2TPv3 Restrictions
- ATM Port Mode Cell Relay over L2TPv3 Restrictions
- ATM Cell Packing over L2TPv3 Restrictions
- Protocol Demultiplexing for L2TPv3 Restrictions
- L2TPv3 Control Message Hashing Restrictions
- L2TPv3 Digest Secret Graceful Switchover Restrictions
- Quality of Service Restrictions in L2TPv3 Tunneling

Supported Port Adapters for the Cisco 7200 Series and Cisco 7500 Series Routers

The following port adapters support L2TPv3 on the Cisco 7200 series and Cisco 7500 series routers:

- Single-port Fast Ethernet 100BASE-TX
- Single-port Fast Ethernet 100BASE-FX
- Dual-port Fast Ethernet 100BASE-TX
- Dual-port Fast Ethernet 100BASE-FX
- Gigabit Ethernet port adapter
- 12-port Ethernet/2-port FE adapter
- 4-port synchronous serial port adapter
- Enhanced 4-port synchronous serial port adapter
- 8-port synchronous serial port adapter
- Single-port HSSI adapter
- Dual-port HSSI adapter
- Single-port enhanced OC-3 ATM port adapter
- 8-port multichannel E1 G.703/G.704 120-ohm interfaces
- 2-port multichannel E1 G.703/G.704 120-ohm interfaces
- 8-port multichannel T1 with integrated data service units (DSUs)
- 8-port multichannel T1 with integrated channel service units (CSUs) and DSUs
- 4-port multichannel T1 with integrated CSUs and DSUs
- 2-port multichannel T1 with integrated CSUs and DSUs
- 8-port multichannel T1/E1
- 1-port multichannel T3 interface
- 1-port multichannel E3 interface
- 2-port enhanced multichannel T3 port adapter
- Single-port T3 port adapter
- Single-port E3 port adapter
- 2-port T3 port adapter
- 2-port T3 port adapter
- Single-port Packet over SONET (PoS), single-mode, long reach
- Single-port PoS, single-mode, intermediate reach
- Single-port PoS, multimode
- Eight-port T1 ATM port adapter with inverse multiplexing over ATM (IMA)
- Eight-port E1 ATM port adapter with IMA

The following port adapters support L2TPv3 on the Cisco 7200 series routers only:

- 8-port Ethernet adapter
- 4-port Ethernet adapter

General L2TPv3 Restrictions

- CEF must be enabled for the L2TPv3 feature to function. The xconnect configuration mode is blocked until CEF is enabled. On distributed platforms, such as the Cisco 7500 series, if CEF is disabled while a session is established, the session is torn down and remains down until CEF is reenabled. To enable CEF, use the **ip cef** or **ip cef distributed** command.
- The IP local interface must be a loopback interface. Configuring any other interface with the **ip local interface** command will result in a nonoperational setting.
- The number of sessions on a PPP, High-Level Data Link Control (HDLC), Ethernet, or 802.1q VLAN port 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.
- 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.
- The **ip pmtu** command used to configure the pseudowire class (see the section "Configuring the L2TPv3 Pseudowire") is not supported for static L2TPv3 sessions. As a result, Layer 2 fragmentation of IP packets and Intermediate System-to-Intermediate System (IS-IS) fragmentation through a static L2TPv3 session are not supported.
- The L2TPv3 Layer 2 (IP packet) fragmentation feature (see "Configuring the L2TPv3 Pseudowire") is not supported when the 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 only supported 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.

Cisco 7200 Series-Specific Restrictions

- ATM port mode cell relay is supported only on the PA-A3-T3, PA-A3-E3, and PA-A3-OC-3 ATM port adapters.
- VPI or VPI/VCI rewrite is not supported for any ATM transport mode. Both pairs of PE to CE peer routers must be configured with matching VPI and VCI values except in OAM local emulation mode. For example, if PE1 and CE1 are connected by PVC 10/100, PE2 and CE2 should also be connected by PVC 10/100.

- In OAM local emulation mode only, the VPI/VCI values used for each pair of PE to CE routers need not match. PE1 and CE1 may be configured with one VPI/VCI value, and PE2 and CE2 may be configured with a different VPI/VCI value. For example, if PE1 and CE1 are connected by PVC 10/100, PE2 and CE2 may be connected by PVC 20/200.

Cisco 7301 Specific Restrictions

- The ATM VP Mode Single Cell Relay over L2TPv3 feature is supported only on the PA-A3-T3, PA-A3-E3, and PA-A3-OC-3 ATM port adapters.
- The ATM Single Cell Relay VC Mode over L2TPv3 feature is supported only on the PA-A3-T3, PA-A3-E3, and PA-A3-OC-3 ATM port adapters.

Cisco 7304 Specific Restrictions

- The L2TPv3 Distributed Sequencing feature in Cisco IOS Release 12.2(27)SBC is supported only on the Cisco 7304 NPE-G100.
- The Protocol Demultiplexing feature in Cisco IOS Release 12.2(27)SBC is supported only on the Cisco 7304 NPE-G100.
- On the Cisco 7304 platforms, ATM cell relay is supported only on the PA-A3-T3, PA-A3-E3, and PA-A3-OC-3 ATM port adapters. ATM cell relay is not supported on the native line cards 7300-1OC-12ATM and 7300-2OC-3ATM.

Cisco 7500 Series-Specific Restrictions

- Distributed sequencing is supported on Cisco 7500 series routers only. The **ip cef distributed** command must be configured.
- ATM port mode cell relay is supported only on the PA-A3-T3, PA-A3-E3, and PA-A3-OC-3 ATM port adapters.
- VPI or VPI/VCI rewrite is not supported for any ATM transport mode. The peer routers must be configured with matching VPI or VCI values.

Supported Shared Port Adapters for the Cisco 7600 Series Router

The following shared port adapters (SPAs) support L2TPv3 on the Cisco 7600 series routers.

Ethernet

- SPA_TYPE_ETHER_2xGE (2-port Gigabit Ethernet)
- SPA_TYPE_ETHER_2xGE_V2 (2-port Gigabit Ethernet)
- SPA_TYPE_ETHER_5xGE_V2 (5-port Gigabit Ethernet)
- SPA_TYPE_ETHER_1x10GE_V2 (single-port 10-Gigabit Ethernet)

ATM

- SPA_TYPE_KATM_2xOC3 (ATM, 2-port OC3)
- SPA_TYPE_KATM_4xOC3 (ATM, 4-port OC3)
- SPA_TYPE_KATM_1xOC12 (ATM, 1-port OC12)
- SPA_TYPE_KATM_1xOC48 (ATM, 1-port OC48)
- SPA_TYPE_CEOP_24xT1E1(CEoP 24-port T1/E1)
- SPA_TYPE_CEOP_1xOC3 (CEoP 1-port OC3)
- SPA_TYPE_CEOP_2xT3E3 (CEoP 2-port T3/E3)

Cisco 7600 Series-Specific Restrictions

On the Cisco 7600 series routers, L2TPv3 is a linecard feature that is implemented only on the SIP-400 linecard. The minimum hardware requirement for enabling the L2TPv3 service on a 7600 router are an L2TPv3-aware linecard(such as the SIP-400) at the Layer 2 CE- facing side and an IP interface on any linecard at the IP coring-facing side. A service card is not required for L2TPv3.

General Restrictions

L2TPv3 imposes the following general restrictions:

- The layer 2-facing linecard must be an L2TPv3-supporting linecard.
- There must be at least one distinct L2TPv3 tunnel per Layer 2-facing linecard.
- Only IPv4 tunneling is supported for Layer 2 frames (configurations such as EoL2TPv3oMPLS (on the encapsulating provider edge (PE) device are not supported).

EVC/EFP Restrictions

L2TPv3 is not supported in conjunction with EVC features. L2TPv3 can coexist with EVC on the same port, meaning that while one subinterface is used to tunnel dot1q-tagged traffic over L2TP, another subinterface can be used to perform EVC features.

SVI VLAN Interfaces Restrictions

L2TPv3 is not supported on SVI VLAN interfaces.

MIB Support Restrictions

There is no L2TPv3-specific MIB support.

Layer Frame Fragmentation Restrictions

Layer 2 frame fragmentation is not supported. Even if the Layer 2 frame recovered after the L2TPv3 decapsulation exceeds the Layer 2 MTU on the CE-facing interface, the SIP-400 linecard still sends the entire Layer 2 frame to the CE device. The Layer 2 frame may be dropped on the CE device because of MRU violations.

Layer 2 Virtual Private Network Interworking Restrictions

The SIP-400 linecard does not support Layer 2 VPN interworking ("like to like" is the only mode supported for L2TPv3 tunneling).

Packet Sequencing Restrictions

The initial release of L2TPv3 focuses on tunneling Ethernet and ATM traffic over L2TPv3. Because of performance issues, the SIP-400 linecard does not support L2TPv3 packet sequencing for Ethernet and ATM traffic. As a result, the 4-byte Layer 2-specific sublayer control word is not supported for Ethernet pseudowires. Configuring sequencing on a pseudowire will cause L2VPN traffic corruption.

By default, sequencing is disabled. However, you can configure sequencing in the pseudowire class, because the pseudowire class may be applied to pseudowires on other 7600 linecards that support sequencing. You must keep sequencing disabled when the pseudowire is handled on the SIP-400 linecard.

Counters Restrictions

Per-session counters are provided by the linecard. Per-tunnel counters are not provided.

Security and QoS ACLs Restrictions

The security QoS ACLs are not supported on the Layer 2 interfaces facing customer device, which means that you cannot apply ACLs to Layer 2 VPN traffic. (The Security ACL and the QoS ACL can still be applied to the IP interfaces at the core-facing side.)

DF Bit Reflection from Inner IP to Outer IP Restrictions

You cannot enable or disable DF bit reflection. The SIP-400 linecard makes DF bit reflection a default behavior for traffic on Ethernet interfaces. When an Ethernet frame is received from the CE device, the SIP-400 linecard checks the IP header inside the frame. Once an IP header is found, the DF bit is copied to the outer tunnel IP during L2TPv3 encapsulation. If no IP header is found inside the Layer 2 frame, the DF bit in the outer IP is set 0.

Traffic on ATM interfaces may have a deep stack of Layer 2 encapsulations. For example, the IP packet may be embedded first in Ethernet, then in SNAP and AAL5. There is no guarantee that the SIP-400 linecard will find the IP packet inside the AAL5 envelope. Therefore, DF bit reflection from inner IP to outer IP is not the default behavior for traffic on ATM interfaces.

Session Cookie

A cookie check is supported for data packets. Cookies (remote and local) can be part of the decapsulation table indexed by *session-id*.

Scalability

Up to 8,000 pseudowires and 512 tunnels are supported.

Set DF Bit in Outer IP

When the **ip dfbit set** command is configured for the pseudowire, the SIP-400 linecard sets the DF bit in the outer IP header during L2TPv3 encapsulation. This DF bit handling is subject to ISIS packet fragmentation.

Set TTL in Outer IP

When the **ip ttl value** command is configured for the pseudowire, the SIP-400 linecard sets the TTL value in the outer IP header during L2TPv3 encapsulation. When the TTL value is not set, the TTL value in the outer IP header is set to 254.

Layer 2-Specific Sublayer Control Word

The Layer 2-specific sublayer control word is defined in L2TPv3 RFCs solely for the purpose of packet sequencing (with the exception of AAL5 payload). On Cisco 7200 series, Cisco 7500 series, and Cisco 12000 series routers, the control word is omitted when sequencing is disabled on non-ATM AAL5 pseudowires. To interoperate with Cisco 7200 series, Cisco 7500 series, and Cisco 12000 series routers, the SIP-400 linecard does not support control words on all non-AAL5 pseudowire types in the initial release.

Table 1 Layer 2 VPN over L2TPv3 Protocol Stack (without Sequencing)

L2TPv3 Packet Stack for AAL5 Payload	L2TPv3 Packet Stack for Non-AAL5 Payload
20 bytes IP header Protocol ID = 115	20 bytes IP header Protocol ID = 115
4 bytes session ID	4 bytes session ID
0, 4 or 8 bytes cookie	0, 4 or 8 bytes cookie
4 bytes control word	Layer 2 frame (non-AAL5
AAL5 frame	

MTU Support

MTU processing is done on the ingress path on the SIP-400 linecard. The SIP-400 linecard enforces Layer 2 MRU checking for every Layer 2 frame received from the CE device. All frames that fail MRU checking are dropped, and the accepted frames are entered into the L2TPv3 encapsulation process. During the process, the whole L2TPV3 packets (including outer IP) are checked again using IP MTU. The packets that pass IP MTU checking are sent to Enhanced Address Recognition Logic (EARL) for IP routing. The failed packets are sent to RP for IP fragmentation or for drop accounting and notifying.

Path MTU discovery is enabled when the **ip pmtu** command is configured for the pseudowire. This feature requires an ingress Layer 2 frame to be dropped if, after L2TPv3 encapsulation, the total packet length exceeds L2TP tunnel path MTU, and the DF bit of the IP header inside the Layer 2 frame is 1. To support this feature, the SIP-400 linecard performs tunnel path MTU checking on each ingress Layer 2 frame during L2TPv3 encapsulation phase. If the total packet length after encapsulation exceeds path MTU, the SIP-400 linecard forwards the original Layer 2 frame to the route processor. On receiving the Layer 2 frame, the route processor may send an Internet Control Message Protocol (ICMP) unreachable message to the source of the IP packet, depending on how deep the IP packet is embedded in the Layer 2 frame.

L2TPv3 IP packet fragmentation and reassembly is done by software on the route processor. The SIP-400 linecard performs core-facing interface IP MTU checking on all packets encapsulated in L2TPv3. If the MTU checking fails, the original Layer 2 frames are sent to the route processor for IP fragmentation. Fragmented L2TPV3 IP packets received from the IP core are received by the route processor from the core facing interface by EARL. The route processor handles L2TPv3 packet reassembly and recovers the inner Layer 2 frame. The route processor also sends the Layer 2 frame to the CE-facing interface by using index-directed WAN dbus frames.

With ISIS packet fragmentation, ISIS packets are often padded to the maximum MTU size. L2TPv3 encapsulation increases the packet size by 28 to 36 bytes. A Layer 2 frame with an ISIS packet embedded may exceed the tunnel path MTU after L2TPV3 encapsulation. Therefore, L3 fragmentation is often needed. To support fragmentation, the SIP-400 linecard searches for ISIS packets in a Layer 2 Frame. If an ISIS packet is found during L2TPv3 encapsulation, the SIP-400 linecard clears the DF bit in the outer IP and sets IP precedence to 6. This allows the IP packet to be fragmented when traveling through the IP core.

Ethernet Attachment Circuits

The SIP-400 linecard supports Ethernet over L2TPv3 in compliance with RFC4719. Two types of pseudowire are supported: Ethernet VLAN pseudowire type (0x0004) and Ethernet pseudowire type (0x0005). When xconnect is configured on an Ethernet main interface, Ethernet frames are tunneled over L2TPv3 using Ethernet port pseudowires (type 0x0005). In this mode, Ethernet frames received on the port (tagged or untagged) are delivered to the remote CE device unaltered.

When xconnect is configured on a dot1q subinterface, the tagged Ethernet frames are tunneled using an Ethernet VLAN pseudowire (type 0x0004). In this case, the pseudowire connects one Ethernet VLAN to another Ethernet VLAN. Received Ethernet VLAN frames from the CE device are tunneled over L2TPv3 unchanged. When arriving on the destination PE device, the original VLAN tag is written to use the destination VLAN ID. While doing so, the priority field in the VLAN tag is preserved.

Ethernet OAM Support

The SIP-400 linecard supports service-level OAM and link-level OAM features on Ethernet interfaces.

Service-level OAM packets, also known as Connectivity Fault Management (CFM) packets, are sent using SNAP header with type 0x0126. Link-level OAM packets, also known as Link Monitoring (LM) packets are sent on Ether-Type 0x8809.

The SIP-400 linecard monitors the above two types of ingress OAM frames from the CE device. When the OAM frames are found and OAM features are configured on the Ethernet interface, the OAM frames are intercepted and forwarded to the route processor. If there is no Ethernet OAM configuration, all OAM frames are tunneled in L2TPv3 as normal data frames.

ATM Attachment Circuits

The SIP-400 linecard supports ATM over L2TPv3 in compliance with RFC 4454 with minor deviation. RFC 4454 defines four types of ATM pseudowire:

- ATM AAL5 SDU VCC transport (0x0002)
- ATM Cell transport port mode (0x0003)
- ATM cell transport VCC mode (0x0009)
- ATM cell transport VPC mode (0x000A).

ATM cell transport port mode is not supported

When xconnect is configured on a PVC with encapsulation AAL5, ATM AAL5 pseudowire (0x0002) is used to tunnel AAL5 frames between PE devices. The SIP-400 linecard supports Layer 2 sublayer-specific control words for AAL5 pseudowire. This is the only type of pseudowire allowed to carry control words.

When xconnect is configured on PVC in AAL0 mode, an ATM cell transport VCC pseudowire (type 0x0009) is used. When xconnect is configured on PVP in AAL0 mode, an ATM cell transport VPC pseudowire (type 0x000A) is used. In both types of pseudowire, each L2TPv3 packet carries one ATM cell. Cell packing is not supported.

ATM OAM Cells

The SIP-400 linecard supports ATM OAM cells operating at VP and VC levels. F4 cells operate at the VP level. They use the same VPI as the user data cells. However, they use two different reserved VCIs, as follows:

- VCI = 3 Segment OAM F4 cells
- VCI = 4 End-to-end OAM F4 cells

OAM F5 cells operate at the VC level. They use the same VPI and VCI as the user cells. To distinguish between data and OAM cells, the PTI field is used as follows:

- PTI = 100 (4) Segment OAM F5 cells processed by the next segment
- PTI =101 (5) End-to-end OAM F5 cells which are only processed by end stations terminating an ATM link

In the ingress direction (CE to PE), because of OAM emulation not supported in the 12.2(33)SRC release, all OAM cells are handled the same as data cells on the SIP-400 linecard. Both segment and end-to-end OAM F4/F5 cells are tunneled over L2TPv3 to the remote PE device. They are sent transparently across the IP core in L2TPv3 tunnels.

In the egress direction (PE to CE), the SIP-400 linecard sends all OAM cells to the CE device similar to sending ATM data cells.

Loopback Interface Reservation

You must reserve a loopback interface used as a source of the L2TPv3 tunnel for a particular linecard to prevent it from being used across multiple linecards. These loopback interfaces host the local IP addresses used by the L2TP tunnels. A minimum of one such IP address is needed for every CE-facing linecard. In most cases, you must create multiple loopback interfaces to accommodate routing protocol configuration and L2TPv3 configuration. Also, you must explicitly use the **mpls ldp router-id** command to avoid LDP router ID changes after system reload.

To reserve a loopback interface, use the following command on the route processor in interface configuration mode:

mls reserve l2tpv3 slot *slot-number* [**processor** *processor-number*]

This command binds the loopback interface to the specified slot/NP. Once configured, the loopback cannot be used to configure L2TPv3 tunnels on other LC/NPs. You must create another loopback interface in order to configure an L2TPv3 pseudowire on an interface that resides on another LC/NP.

QoS

QoS is handled on the linecard. EARL does not perform QoS operations on L2TPv3 packets.

QoS at L2TPv3 Tunnel Ingress

The SIP-400 linecard applies QoS to ingress traffic before doing L2TPv3 encapsulation. Given the order of traffic processing, the SIP-400 linecard can support full-fledged interface/PVC level MQC on Layer 2 traffic. QoS on IP tunnel traffic is limited to ToS marking only.

The supported QoS-on-ingress Layer 2 frames are as follows.

- Classification. Ethernet interfaces: match on vlan, cos, ip dscp, ip precedence. ATM interfaces: match on atm clp
- Marking:
  - Ethernet interfaces: set cos
  - ATM interfaces: none

- Policing on both Ethernet and ATM interfaces
- Queuing on Ethernet interfaces

QoS at L2TPv3 Tunnel Egress

With egress traffic flow on the SIP-400 linecard, QoS is again applied to Layer 2 traffic after L2TPv3 de-encapsulation. While the SIP-400 linecard can support full-fledged Layer 2 MQC at the interface/PVC level, no QoS can be done on the IP tunnel traffic.

The supported QoS-on-egress Layer 2 frames are as follows.

- Classification:
  - Ethernet interfaces: match on vlan, cos, ip dscp, ip precedence
  - ATM interfaces: none
- Marking:
  - Ethernet interfaces: set cos, ip dscp, ip precedence
  - ATM interfaces: set atm clp
- Policing on both Ethernet and ATM interfaces
- Queuing on both Ethernet and ATM interfaces

L2TPv3 Packet ToS Marking

L2TPv3 packet ToS marking is done on the SIP-400 ingress path. There are three ways of marking the ToS field:

- Configure the **ip tos value** *value* command on each pseudowire to set the ToS field
- Configure the **ip tos reflect** command on each pseudowire to allow the inner IP ToS copied to the outer IP ToS
- By default, Layer 2 QoS is automatically reflected to outer IP ToS. For example, if the Layer 2 frame is an 802.Q frame, the 3-bit priority field in the VLAN tag is copied to the precedence bits in the outer IP ToS field

When the **ip tos reflect** command is configured, the SIP-400 linecard searches for an IP header inside each received Layer 2 frame. If an IP packet is found, its ToS is copied to the outer ToS. Otherwise, the ToS value in the L2TPv3 IP header is set 0.

When neither the **ip tos value** command nor the **ip tos reflect** command is configured, the SIP-400 linecard searches for a VLAN tag in each Ethernet frame. If a tag is found, the inner Layer 2 QoS is reflected to the outer IP ToS. Otherwise, the L2TPv3 IP ToS field is set 0.

Cisco 10720-Specific Restrictions

- Variable cookie size and L2TPv3 sequencing are not supported.
- Starting in Cisco IOS Release 12.0(32)SY, the L2TPv3 Layer 2 Fragmentation feature is supported on the Cisco 10720 Internet router to enable the fragmentation of IP packets to occur before data enters the pseudowire. When you enable this feature in an L2TPv3 pseudowire configuration using the **ip pmtu** command, the Don't Fragment (DF) bit in the outer Layer 2 packet header is automatically set on so that (for performance reasons) tunneled packets are not reassembled on the decapsulation router.

- The Cisco 10720 Internet router supports the reassembly only of fragmented IS-IS packets in an L2TPv3 pseudowire. IS-IS packet reassembly is performed by the Route Processor (RP) at the process level, not in the Parallel eXpress Forwarding (PXF) forwarding path.
- On the Cisco 10720 Internet router, the **uti translation** command is not migrated for xconnect service and is not supported. Although the **uti** command is supported in L2TPv3 releases, the **translation** option is lost in the migration.
- On the Cisco 10720 Internet router, although it is not required, we highly recommend that you configure a loopback interface as the IP local interface.

  You can also configure a LAN interface as the IP local interface so that the tunnel control session is tied to an operational LAN (Gigabit Ethernet or Fast Ethernet) interface or subinterface. However, in this case, the tunnel control plane is used only as long as the Gigabit Ethernet or Fast Ethernet interface is operational.

Cisco 12000 Series-Specific Restrictions

Tunnel Server Card Versus Native L2TPv3 Implementation

On the Cisco 12000 series Internet router, L2TPv3 is implemented in two different ways:

- The 1-port OC-48c/STM-16c POS/SDH line card is required as the dedicated tunnel server card (TSC) to accelerate the encapsulation and decapsulation of Layer 2 data on engine 2 (and earlier engine types) line cards in an L2TPv3 tunnel session.
- The enhanced edge capabilities of IP services engine (ISE) and engine 5 line cards do not require a tunnel server card for Layer 2 data encapsulation and decapsulation in an L2TPv3 tunnel. This is called a *native L2TPv3* session.

Note Native L2TPv3 tunnel sessions on customer-facing ISE and Engine 5 line cards can coexist with tunnel sessions that use a tunnel server card.

Different combinations of engine types are supported as customer-facing and backbone-facing line cards for encapsulation and decapsulation in L2TPv3 tunneling.

Note If you have native cards (engine 3 and engine 5) in the PE routers and the Tunnel Server Card is configured to support the non-native cards, then you must remove the TSC configuration by using the **no hw-module slot** <*number*> **mode server** command. If the TSC configuration exists in the PE router and the TSC card is removed, all the tunnels will fail.

L2TPv3 Encapsulation

When a Layer 2 packet arrives on a customer-facing interface, if the interface is bound to an L2TPv3 tunnel, L2TPv3 encapsulation is supported as follows:

- If the customer-facing line card is engine 2 or an earlier engine type, the line card forwards the packet to the tunnel server card, which performs L2TPv3 encapsulation.
- If the customer-facing line card is ISE or engine 5, the line card performs L2TPv3 encapsulation.

A backbone-facing line card of any engine type sends the packet across the service provider backbone network.

L2TPv3 Decapsulation

When an L2TPv3 packet arrives on a backbone-facing interface, L2TPv3 decapsulation is supported as follows:

- If the backbone-facing line card is non-ISE/E5 (any engine type besides ISE and Engine 5), the line card forwards the packet to the tunnel server card. The tunnel server card determines if the packet is bound to an Engine 2 (or earlier engine) or an ISE/E5 customer-facing line card.
  - If the packet is bound to an Engine 2 (or earlier engine) customer-facing line card, the TSC completes packet decapsulation and sends the Layer 2 packet to the customer-facing interface.
  - If the packet is bound to an ISE/E5 customer-facing line card, the TSC sends the packet to the line card for further decapsulation.
- If the backbone-facing line card is ISE/E5, the line card determines if the packet is bound to an Engine 2 (or earlier engine) or an ISE/E5 customer-facing line card.
  - If the packet is bound to an Engine 2 (or earlier engine) customer-facing line card, the packet is sent to the tunnel server card for further decapsulation. Afterward, the decapsulated Layer 2 packet is sent to the Engine 2 (or earlier engine) customer-facing interface.
  - If the packet is bound to an ISE/E5 customer-facing line card, the packet is sent to the ISE/E5 line card for decapsulation.

Note If no tunnel server card is installed, L2TPv3 decapsulation is not supported in the following conditions:
- The customer-facing line card is Engine 2 or an earlier engine line card.
- The customer-facing line card is ISE/E5 and the backbone-facing line card is non-ISE/5.
In these cases, packets received on the backbone-facing interface are dropped. The following warning message is logged: L2TPv3 decapsulation packet dropped.

Cisco 12000 Series Line Cards—General Restrictions

- IS-IS protocol packet fragmentation is supported only for dynamic L2TPv3 sessions.
- Hairpinning is not supported for local-to-local switching. The start and end of an L2TPv3 session must terminate on different routers linked by an IP or MPLS backbone.
- The L2TPv3 feature set is supported as follows. If a tunnel server card is:
  - Installed, and only Engine 2 or older customer-facing line cards are used, normal L2TPv3 tunnel sessions are supported with the L2TPv3 feature set described in L2TPv3 Features, page 32.
  - Is not installed and ISE/E5 backbone-facing and ISE/E5 customer-facing line cards are used, native L2TPv3 tunnel sessions are supported with the native L2TPv3 feature set described in Table 4.
  - Installed and a combination of Engine 2 or older and ISE/E5 line cards is used as customer-facing line cards, a mixed L2TPv3 tunnel session is supported with the native L2TPv3 feature set described in Table 4.
  - Installed and a ISE/E5 customer-facing and Engine 2 or older backbone-facing line cards are used, a mixed L2TPv3 tunnel session is supported with the native L2TPv3 feature set described in L2TPv3 Encapsulation and L2TPv3 Decapsulation.
- Engine 4 and Engine 4 Plus (E4+) line cards are not supported as the customer-facing line cards in an L2TPv3 tunnel session. However, Engine 4 and Engine 4+ line cards may be used to provide other services in a Layer 2 VPN.

- In a native L2TPv3 tunnel session configured on ISE/E5 interfaces, 802.1q (VLAN) is supported as an L2TPv3 payload starting in Cisco IOS Release 12.0(31)S.

Engine 2 and Earlier Engine-Specific Restrictions

- A dedicated 1-port OC-48c/STM-16c POS/SDH tunnel server card is required for L2TPv3 to function. The server card does not run Engine 2 features.
- TSC-based L2TPv3 tunnel sessions are supported only if a tunnel server card is configured.

  To configure the server card, you must enter the **ip unnumbered** command and configure the IP address on the PoS interface of the server card before you configure hardware modules. Then enter the **hw-module slot** *slot-number* **mode server** command.

  This initial configuration makes the server card IP-aware for backbones requiring an Address Resolution Protocol (ARP) to be generated by the line card. The backbone types that require this configuration are Ethernet and Spatial Reuse Protocol (SRP).

  This configuration is also a requirement for session keepalives. The interface port of the server card is automatically set to loopback internal and no keepalives when the **hw-module slot** *slot-number* **mode server** command is configured.

Note Starting in Cisco IOS Release 12.0(30)S, you must first remove all L2TPv3 xconnect attachment circuits on all Engine-2 or earlier engine customer-facing line cards before you enter the **no hw-module slot** *slot-number* **mode server** command to unconfigure a tunnel server card.

- On the tunnel server card:
  - The IP local interface must be a local loopback interface. Configuring any other interface as the IP local interface results in nonoperational sessions.
  - The IP local interface must be dedicated for the use of L2TPv3 sessions. This interface must not be shared by any other routing or tunneling protocols.
  - The maximum performance of 2.5 million packets per second (pps) is achieved only if you use transmit buffer management (TBM) ASIC ID 60F1. Other ASIC ID versions can cause the performance to be reduced by half. To determine the ASIC value of the line card, use the **execute-on slot** *slot-number* **show controller frfab bma reg | include asic** command, where *slot-number* is the slot number of the server card.
- Cover the optics of the tunnel server card because of possible interference or noise causing cyclic redundancy check (CRC) errors on the line card. These errors are caused by a framer problem in the line card.
- The aggregate performance is bound by the server card limit of 2.5 million pps.
- Because of a framer problem, the server card interfaces accounting in (packets out) are not accurate.
- Only features found in the Vanilla uCode bundle are supported on Engine 2 line cards that are associated with an L2TPv3 session and on a different interface, DLCI, or VLAN of the same line card.
- When you configure an Engine 2 feature, which is not in the Vanilla uCode bundle on an Engine 2 line card, on an interface bound to an L2TPv3 tunnel session, the Vanilla uCode is swapped out. As a result, all traffic through the L2TPv3 session stops on the Engine 2 line card. In this case, you must restore the Vanilla uCode bundle on the line card, and rebind the attachment circuit to the L2TPv3 session as described in Configuring the Xconnect Attachment Circuit, page 62.

- Configuring output access control lists (ACLs) on any line card swaps out the running Engine 2 line card Vanilla uCode bundle in favor of the ACL uCode bundle. This configuration causes all traffic through the L2TPv3 session to stop on those Engine 2 line cards. If output ACLs are essential on the router, we advise you to originate all L2TPv3 sessions on Engine 0 line cards. Output ACLs do not swap out the server card uCode bundle because of the higher priority.
- Engine 2 line cards do not support Frame Relay switching and Frame Relay L2TPv3 DLCI session on the same line card.
- On Engine 2 line cards, the input Frame Relay permanent virtual circuit (PVC) counters are not updated.
- If the 8-port Fast Ethernet (Engine 1) line card is connected to a hub or switch when L2TPv3 is configured on the ingress side of one or more of its ports, duplicate packets are generated, causing the router to be flooded with packets. This restriction results from the requirement that CAM filtering is disabled when L2TPv3 is used.
- On the 3-port Gigabyte Ethernet (Engine 2) line card, performance degradation can occur if IP packets coming from a port are sent to the slow path for forwarding. This performance degradation occurs if both the following conditions are met:
  - The port has at least one 802.1q subinterface that is in an L2TPv3 session.
  - The IP packet comes from the port interface itself (not 802.1q encapsulated) or from an 802.1q subinterface that is under the port interface but has no L2TPv3 session bound to it.

Edge Line Card-Specific Restrictions

The following restrictions apply to L2TPv3 sessions configured on IP Services Engine (ISE) and Engine 5 edge line cards:

- Native L2TPv3 sessions are supported only if the feature mode is configured on a customer-facing ISE/E5 line card.

  To configure the feature mode, enter the **hw-module slot** *slot-number* **np mode feature** command. You cannot unconfigure the feature mode on a customer-facing ISE/E5 line card until all L2TPv3 xconnect attachment circuits on the line card are removed.

  A backbone-facing ISE/E5 line card can operate in any mode and no special feature mode configuration is required.
- Starting in Cisco IOS Release 12.0(31)S, 802.1q (VLAN) is supported as an L2TPv3 payload in a native L2TPv3 tunnel session configured on ISE/E5 interfaces.
- Native L2TPv3 tunnel sessions on customer-facing ISE/E5 line cards can coexist with tunnel sessions that use a tunnel server card.
- L2TPv3 encapsulation on a customer-facing ISE/E5 line card does not support the L2TPv3 Layer 2 Fragmentation feature.

  This means that if you enter the **ip pmtu** command to enable the discovery of a path maximum transmission unit (PMTU) for L2TPv3 traffic, and a customer IP packet exceeds the PMTU, IP fragmentation is not performed on the IP packet before L2TPv3 encapsulation. These packets are dropped. For more information, see L2TPv3 Layer 2 Fragmentation, page 34.

Table 2 and Table 3 show the ISE and E5 interfaces that are supported in a native L2TPv3 tunnel on:

- Customer-facing line cards (ingress encapsulation and egress decapsulation)
- Backbone-facing line cards (ingress decapsulation and egress encapsulation)

Table 2 ISE Interfaces Supported in a Native L2TPv3 Tunnel Session

ISE Line Card	Native L2TPv3 Session on Customer-Facing Interface	Native L2TPv3 Session on Backbone-Facing Interface
4-port OC-3 POS ISE	Supported	Supported
8-port OC-3 POS ISE	Supported	Supported
16-port OC-3 POS ISE	Supported	Supported
4-port OC-12 POS ISE	Supported	Supported
1-port OC-48 POS ISE	Supported	Supported
1-port channelized OC-12 (DS1) ISE	Supported	Not supported
2.5G ISE SPA Interface Processor[1]: • 2-port T3/E3 serial SPA • 4-port T3/E3 serial SPA • 2-port channelized T3 to DS0 SPA • 4-port channelized T3 to DS0 SPA	Supported	Not supported
1-port channelized OC-48 POS ISE	Not supported	Not supported
4-port OC-3 ATM ISE	Supported	Supported
4-port OC-12 ATM ISE	Supported	Supported
4-port Gigabit Ethernet ISE [2]	Supported	Supported

1. For more information about the shared port adapters (SPAs) and SPA interface platforms (SIPs) supported on Cisco 12000 series routers, refer to the *Cisco 12000 Series Router SIP and SPA Hardware Installation Guide*.
2. The 4-port Gigabit Ethernet ISE line card supports VLAN membership (port-based and VLAN-based) in a native L2TPv3 tunnel session on customer-facing and backbone-facing interfaces. See 802.1q (VLAN) for more information.

Table 3 Engine 5 Interfaces Supported in a Native L2TPv3 Tunnel Session

Engine 5 SPA	Native L2TPv3 Session on Customer-facing Interface	Native L2TPv3 Session on Backbone-facing Interface
1-port channelized STM-1/OC-3 to DS0	Supported	Not supported
8-port channelized T1/E1	Supported	Not supported
1-port 10-Gigabit Ethernet	Supported	Supported
5-port Gigabit Ethernet	Supported	Supported
10-port Gigabit Ethernet	Not supported	Supported
8-port Fast Ethernet	Supported	Supported
4-port OC-3/STM4 POS	Supported	Not supported
8-port OC-3/STM4 POS	Supported	Not supported
2-port OC-12/STM4 POS	Supported	Not supported
4-port OC-12/STM4 POS	Supported	Not supported
8-port OC-12/STM4 POS	Supported	Not supported

Table 3 Engine 5 Interfaces Supported in a Native L2TPv3 Tunnel Session (Continued)

Engine 5 SPA	Native L2TPv3 Session on Customer-facing Interface	Native L2TPv3 Session on Backbone-facing Interface
2-port OC-48/STM16 POS/RPR	Not supported	Supported
1-port OC192/STM64 POS/RPR	Not supported	Supported

Table 4 describes the L2TPv3 features supported in a native L2TPv3 tunnel session and the customer-facing ISE/E5 line cards that support each feature. Note that although native L2TPv3 sessions do not support L2TPv3 Layer 2 (IP packet) fragmentation and slow-path switching features, ATM (as a transport type) and QoS features (traffic policing and shaping) across all media types are supported.

Table 4 L2TPv3 Features Supported in a Native L2TPv3 Session

Native L2TPv3 Feature	ISE Line Cards (Customer-facing) Supported	E5 Line Cards (Customer-facing) Supported
Native L2TPv3 tunneling (fast-path) Supports the same L2TPv3 features that are supported by server card-based L2TPv3 tunneling, except that L2TPv3 Layer 2 (IP packet) fragmentation is not supported. For more information, see the "L2TPv3 Features" section.	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 4-port OC-3 ATM ISE 4-port OC-12 ATM ISE 4-port Gigabit Ethernet ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port T3/E3 Serial - 4-port T3/E3 Serial - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 8-port fast Ethernet - 5-port Gigabit Ethernet - 10-port Gigabit Ethernet - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS
L2TP class and pseudowire class configuration You can create an L2TP template of L2TP control channel parameters that can be inherited by different pseudowire classes configured on a PE router. You can also configure a pseudowire template of L2TPv3 session-level parameters that can be used to configure the transport Layer 2 traffic over an xconnect attachment circuit. For more information, see the sections "Configuring L2TP Control Channel Parameters" and "Configuring the L2TPv3 Pseudowire."	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 4-port OC-3 ATM ISE 4-port OC-12 ATM ISE 4-port Gigabit Ethernet ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port T3/E3 Serial - 4-port T3/E3 Serial - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 8-port Fast Ethernet - 5-port Gigabit Ethernet - 10-port Gigabit Ethernet - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS

Table 4 L2TPv3 Features Supported in a Native L2TPv3 Session (Continued)

Native L2TPv3 Feature	ISE Line Cards (Customer-facing) Supported	E5 Line Cards (Customer-facing) Supported
L2TPv3 tunnel marking and traffic policing on the following types of ingress interfaces, when bound to a native L2TPv3 tunnel session: - 802.1q (VLAN) - ATM - Channelized - Ethernet - Frame Relay DLCIs The following conform, exceed, and violate values for the *action* argument are supported for the **police** command when QoS policies are configured on an ISE/E5 ingress interface bound to a native L2TPv3 tunnel. The **set** commands can also be used to set the IP precedence or DSCP value in the tunnel header of a L2TPv3 tunneled packet on an ingress interface. **conform-action** *actions*: **set-prec-tunnel** **set-dscp-tunnel** **transmit** **exceed-action** *actions*: **drop** **set-clp** (ATM only) **set-dscp-tunnel** **set-dscp-tunnel** and **set-clp** (ATM only) **set-dscp-tunnel** and **set-frde** (Frame Relay only) **set-frde** (Frame Relay only) **set-prec-tunnel** **set-prec-tunnel** and **set-clp** (ATM only) **set-prec-tunnel** and **set-frde** (Frame Relay only) **transmit** **violate-action** *actions*: **drop** See *"QoS: Tunnel Marking for L2TPv3 Tunnels"* for information about how to use the L2TPv3 tunnel marking and traffic policing features on Engine 2 (and earlier engine) interfaces bound to a TSC-based L2TPv3 tunnel session.	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 4-port OC-3 ATM ISE 4-port OC-12 ATM ISE 4-port Gigabit Ethernet ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port T3/E3 serial - 4-port T3/E3 serial - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 8-port Fast Ethernet - 5-port Gigabit Ethernet - 10-port Gigabit Ethernet - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS

Table 4 L2TPv3 Features Supported in a Native L2TPv3 Session (Continued)

Native L2TPv3 Feature	ISE Line Cards (Customer-facing) Supported	E5 Line Cards (Customer-facing) Supported
Frame Relay DLCI-to-DLCI tunneling Frame Relay DLCIs are connected to create an end-to-end Frame Relay PVC. Traffic arriving on a DLCI on one interface is forwarded across an L2TPv3 tunnel to another DLCI on the other interface. For more information, see "DLCI-to-DLCI Switching" in the "Frame Relay" section.	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port T3/E3 serial - 4-port T3/E3 serial - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS - 2-port OC-48/STM16 POS/RPR
ATM single cell and packed cell relay: VC mode Each VC is mapped to a single L2TPv3 tunnel session. The following ATM cell relay modes are supported: • ATM cells arriving at an ATM interface with the specified VPI and VCI are encapsulated into a single L2TP packet (single cell relay). • ATM cells arriving at an ingress ATM interface are packed into L2TPv3 data packets and transported to the egress ATM interface (packed cell relay). For more information, see the "ATM" section.	4-port OC-3 ATM ISE 4-port OC-12 ATM ISE	Not supported
ATM single cell and packed cell relay: VP mode ATM cells arriving into a predefined PVP on the ATM interface are transported to a predefined PVP on the egress ATM interface. The following ATM cell relay modes are supported: • A single ATM cell is encapsulated into each L2TPv3 data packet (single cell relay). • Multiple ATM cells are packed into a single L2TPv3 data packet (packed cell relay). For more information, see the "ATM" section.	4-port OC-3 ATM ISE 4-port OC-12 ATM ISE	Not supported

Table 4 L2TPv3 Features Supported in a Native L2TPv3 Session (Continued)

Native L2TPv3 Feature	ISE Line Cards (Customer-facing) Supported	E5 Line Cards (Customer-facing) Supported
ATM single cell relay and packed cell relay: Port mode ATM cells arriving at an ingress ATM interface are encapsulated into L2TPv3 data packets and transported to the egress ATM interface.The following ATM cell relay modes are supported: • A single ATM cell is encapsulated into each L2TPv3 data packet (single cell relay). • Multiple ATM cells are packed into a single L2TPv3 data packet (packed cell relay). For more information, see the "ATM" section.	4-port OC-3 ATM ISE 4-port OC-12 ATM ISE	Not supported
ATM AAL5 PVC tunneling The ATM AAL5 payload of an AAL5 PVC is mapped to a single L2TPv3 session. For more information, see "ATM AAL5" in the "ATM" section.	4-port OC-3 ATM ISE 4-port OC-12 ATM ISE	Not supported
OAM emulation mode for ATM AAL5 OAM local emulation mode for ATM AAL5 payloads is supported. Instead of being passed through the pseudowire, OAM cells are terminated and handled locally. On the L2TPv3-based pseudowire, the CE device sends an SLI message across the pseudowire to notify the peer PE node about the defect, rather than tearing down the session. For more information, see "ATM AAL5 over L2TPv3: OAM Local Emulation Mode" in the "ATM" section.	4-port OC-3 ATM ISE 4-port OC-12 ATM ISE	Not supported
OAM transparent mode for ATM AAL5 OAM transparent mode for ATM AAL5 payloads is supported. The PE routers pass OAM cells transparently across the L2TPv3 tunnel. For more information, see "ATM AAL5 over L2TPv3: OAM Transparent Mode" in the "ATM" section.	4-port OC-3 ATM ISE 4-port OC-12 ATM ISE	Not supported

Table 4 L2TPv3 Features Supported in a Native L2TPv3 Session (Continued)

Native L2TPv3 Feature	ISE Line Cards (Customer-facing) Supported	E5 Line Cards (Customer-facing) Supported
Ethernet port-to-port tunneling Ethernet frames are tunneled through an L2TP pseudowire. For more information, see the "Ethernet" section.	4-port Gigabit Ethernet ISE	Engine 5 SPAs: - 8-port Fast Ethernet - 5-port Gigabit Ethernet - 10-port Gigabit Ethernet
VLAN-to-VLAN tunneling The following types of VLAN membership are supported in an L2TPv3 tunnel: • Port-based, in which undated Ethernet frames are received • VLAN-based, in which tagged Ethernet frames are received For more information, see the "802.1q (VLAN)" section.	4-port Gigabit Ethernet ISE	Engine 5 SPAs: - 8-port Fast Ethernet - 5-port Gigabit Ethernet - 10-port Gigabit Ethernet
Dual rate, 3-Color Marker for traffic policing on Frame Relay DLCIs of ingress interfaces, when bound to a native L2TPv3 tunnel session[1] The dual rate, 3-Color Marker in color-aware and color-blind modes, as defined in RFC 2698 for traffic policing, is supported on ingress ISE interfaces to classify packets. For more information, refer to "QoS: Color-Aware Policer."	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 4-port Gigabit Ethernet ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port T3/E3 serial - 4-port T3/E3 serial - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS - 2-port OC-48/STM16 POS/RPR

Table 4 L2TPv3 Features Supported in a Native L2TPv3 Session (Continued)

Native L2TPv3 Feature	ISE Line Cards (Customer-facing) Supported	E5 Line Cards (Customer-facing) Supported
Traffic shaping on ATM and Frame Relay egress interfaces based on class map configuration is supported. Traffic shaping is supported on ATM egress interfaces for the following service categories: • Lowest priority: UBR (unspecified bit rate) • Second priority: VBR-nrt (variable bit rate nonreal-time) • Highest priority: VBR-rt (VBR real time) • Highest priority: CBR (constant bit rate) [2] For more information, see "QoSTraffic Shaping on ATM Line Cards for the Cisco 12000 Series."	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 4-port OC-3 ATM ISE 4-port OC-12 ATM ISE 4-port Gigabit Ethernet ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port clear channel T3/E3 - 4-port clear channel T3/E3 - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS - 2-port OC-48/STM16 POS/RPR
Layer 2 Virtual Private Network (L2VPN) interworking L2VPN interworking allows attachment circuits using different Layer 2 encapsulation types to be connected over an L2TPv3 pseudowire. On an ISE interface configured for L2TPv3 tunneling, the following Layer 2 encapsulations are supported: ATM AAL5 Ethernet 802.1q (VLAN) Frame Relay DLCI On an Engine 5 interface configured for L2TPv3 tunneling, the following Layer 2 encapsulations are supported: Ethernet 802.1q (VLAN) Frame Relay DLCI	4-port OC-3 POS ISE 8-port OC-3 POS ISE 16-port OC-3 POS ISE 4-port OC-12 POS ISE 1-port OC-48 POS ISE 4-port OC-3 ATM ISE 4-port OC-12 ATM ISE 4-port Gigabit Ethernet ISE 1-port channelized OC-12 (DS1) ISE ISE SPAs: - 2-port T3/E3 serial - 4-port T3/E3 serial - 2-port channelized T3 to DS0 - 4-port channelized T3 to DS0	Engine 5 SPAs: - 1-port channelized STM-1c/OC-3c to DS0 - 8-port channelized T1/E1 - 8-port Fast Ethernet - 8-port 10/100 Ethernet - 1-port 10-Gigabit Ethernet - 2-port Gigabit Ethernet - 5-port Gigabit Ethernet - 10-port Gigabit Ethernet - 4-port OC-3/STM4 POS - 8-port OC-3/STM4 POS - 2-port OC-12/STM4 POS - 4-port OC-12/STM4 POS - 8-port OC-12/STM4 POS - 2-port OC-48/STM16 POS/RPR - 1-port OC192/STM64 POS/RPR

1. Although the dual-rate, 3-Color Marker policer is not supported on ATM ISE/E5 interfaces, the ATM Forum Traffic Management Version 4.1-compliant Generic Cell Rate Algorithm (GCRA) policer is supported. The GCRA policer uses rate, peak rate, delay tolerance, and ATM maximum burst size, and supports the following options:
   - **set-dscp-tunnel**
   - **set-dscp-tunnel** and **set-clp-transmit**
   - **set-prec-tunnel**
   - **set-prec-tunnel** and **set-clp-transmit**
2. Note that VBR-rt and CBR share the same high priority shaping. ATM traffic shaping restricts traffic to the maximum rate configured on an ATM VC or PVP with due priority among the respective service categories.
   You can configure queue limits for an ATM VC or PVP. The queue limits are dual thresholds in which two different thresholds can be configured for CLP=1 cells and CLP0+1 cells. The CLP1 threshold must be lower than the queue limit threshold so that CLP=1 cells are dropped earlier than CLP=0 cells when packets start to fill the queue.

Frame Relay-Specific Restrictions

- Frame Relay per-DLCI forwarding and port-to-port trunking are mutually exclusive. L2TPv3 does not support the use of both on the same interface at the same time.
- The **xconnect** command is not supported on Frame Relay interfaces directly. For Frame Relay, xconnect is applied under the **connect** command specifying the DLCI to be used.
- Changing the encapsulation type on any interface removes any existing **xconnect** command applied to that interface.
- To use DCE or a Network-to-Network Interface (NNI) on a Frame Relay port, you must configure the **frame-relay switching** command.
- The configuration of an L2TPv3 session on a Multilink Frame Relay (MLFR) bundle interface is supported only on Cisco 12000 series 2-port channelized OC-3/STM-1 (DS1/E1) and 6-port Channelized T3 (T1) line cards. (For more information, see Binding L2TPv3 Sessions to Multilink Frame Relay Interfaces, page 43.)
- Frame Relay policing is nondistributed on the Cisco 7500 series. By configuring Frame Relay policing, you cause traffic on the affected PVCs to be sent to the RSP for processing.
- Frame Relay support is for 10-bit DLCI addresses. Frame Relay Extended Addressing is not supported.
- Multipoint DLCI is not supported.
- The keepalive is automatically disabled on interfaces that have an xconnect applied to them, except for Frame Relay encapsulation, which is a requirement for LMI.
- Static L2TPv3 sessions do not support Frame Relay LMI interworking.

VLAN-Specific Restrictions

- A PE router is responsible only for static VLAN membership entries that are manually configured on the router. Dynamic VLAN membership entries, entry aging, and membership discovery are not supported.
- Implicit tagging for VLAN membership operating on the other layers (such as at Layer 2, membership by MAC address or protocol type, at Layer 3, or membership by IP subnet) 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.

ATM VP Mode Single Cell Relay over L2TPv3 Restrictions

- The ATM VP Mode Single Cell Relay over L2TPv3 feature is supported only on the Cisco 7200 and Cisco 7500 series routers with ATM Deluxe PA-A3 interfaces.
- After the ATM VP Mode Single Cell Relay feature is configured for a virtual path connection (VPC), no other permanent virtual circuits (PVCs) are allowed for the same virtual path identifier (VPI).

ATM AAL5 SDU over L2TPv3 and Single Cell Relay VC Mode over L2TPv3 Restrictions

- The ATM AAL5 OAM Emulation over L2TPv3 feature and the ATM Single Cell Relay VC Mode over L2TPv3 feature are supported only on the Cisco 7200, Cisco 7301, Cisco 7304 NSE-100, Cisco 7304 NPE-G100, and Cisco 7500 series routers with ATM Deluxe PA-A3 interfaces.
- Sequencing is supported only for ATM adaptation layer 5 (AAL5) service data unit (SDU) frames or ATM cell relay packets. Sequencing of Operation, Administration, and Maintenance (OAM) cells is not supported.
- Sequencing is supported in CEF mode. If sequencing is enabled with dCEF, all L2TP packets that require sequence number processing are sent to the RSP module.
- L2TPv3 manual mode configuration does not support ATM alarm signaling over the pseudowire.
- The Cisco 7200 series and the Cisco 7500 series ATM driver cannot forward Resource Management (RM) OAM cells over the packet-switched network (PSN) for available bit rate (ABR) ToS. The RM cells are locally terminated.

ATM Port Mode Cell Relay over L2TPv3 Restrictions

- Port mode and virtual path (VP) or VC mode cell relay are mutually exclusive. After the ATM interface is configured for cell relay, no permanent virtual path (PVP) or PVC commands are allowed on that interface.
- ATM port mode cell relay is supported only on the PA-A3-T3, PA-A3-E3, and PA-A3-OC-3 ATM port adapters.
- ATM port mode cell relay is not supported on the PA-A3-8T1IMA and PA-A3-8E1IMA port adapters.

ATM Cell Packing over L2TPv3 Restrictions

- The ATM Cell Packing over L2TPv3 feature is supported only on PA-A3 ATM interfaces on Cisco 7200 and Cisco 7500 routers. Cell packing cannot be configured on other platforms or interface cards.
- A minimum of 2 and a maximum of 28 ATM cells can be packed into an L2TPv3 data packet.

Protocol Demultiplexing for L2TPv3 Restrictions

- IPv6 protocol demultiplexing is supported for Ethernet and terminated DLCI Frame Relay interfaces, PPP traffic, and HDLC traffic.
- Frame Relay demultiplexing is supported for point-to-point or multipoint.
- FRF.12 end-to-end fragmentation is supported on the Cisco 7500 and Cisco 12000 series routers only between the CE and the PE routers.
- FRF.9 hardware payload compression is supported on the Cisco 7200 series and Cisco 7500 series routers only between the CE and the PE routers.
- FRF.9 software payload compression is supported on the Cisco 7500 series routers only between the CE and the PE routers.

- FRF.9 process switched payload compression is not supported.
- IETF encapsulation must be used with FRF.9.
- FRF.16 is supported only between the CE and the PE routers.
- HDLC restrictions for protocol demultiplexing:
  - IP must be enabled on the interface if you want to configure protocol demultiplexing using the **xconnect** command.
  - IPv6 cannot be enabled on the interface at the same time as the **xconnect** command (with or without protocol demultiplexing).
  - Payload compression is not supported.
- Cisco 12000 series router restrictions for protocol demultiplexing:
  - If a Cisco 12000 series router is acting as the PE with IPv6 protocol demultiplexing using PPP, the remote PE must also be a Cisco 12000 series router.
  - IPv6 protocol demultiplexing for Ethernet encapsulation on Engine-5 line cards is only supported with Version-2 Ethernet SPAs. It is not supported with Version-1 Ethernet SPAs.
  - IPv6 protocol demultiplexing is not supported on the SIP-400 Engine-3 line card.
- IPv6 protocol demultiplexing with PPP encapsulation must be configured in the following order to ensure a working tunnel session:
  1. Configure the IP address on the interface.
  2. Enter the encapsulation PPP command.
  3. Enter the PPP **ipv6cp id proxy** <*ipv6 address*> command.
  4. Enter the **xconnect** command with the **match protocol ipv6** command.

  If this configuration order is not followed, the tunnel session cannot operate until you issue a **shut/no shut** command on the protocol demultiplexing interface or do an OIR.

L2TPv3 Control Message Hashing Restrictions

- L2TPv3 control channel authentication configured with the **digest** command requires bidirectional configuration on the peer routers, and a shared secret must be configured on the communicating nodes.
- See Table 8 for a compatibility matrix of all the L2TPv3 authentication methods available in Cisco IOS Release 12.0(29)S and Cisco IOS Release 12.2(27)SBC and later releases.

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, 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.

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:

Frame Relay Interface (Non-ISE/E5)

- On the Cisco 7500 series with distributed CEF (dCEF), in a QoS policy applied to a Frame Relay interface configured for L2TPv3, only the MQC commands **match fr-dlci** in class-map configuration mode and **bandwidth** in policy-map configuration mode are supported. (See Configuring QoS for L2TPv3 on the Cisco 7500 Series: Example, page 101.)
- On the Cisco 12000 series, a QoS policy is supported in TSC-based L2TPv3 tunnel sessions on the Frame Relay interfaces of a 2-port channelized OC-3/STM-1 (DS1/E1) or 6-port channelized T3 (T1) line card with the following restrictions:
  - The **police** command is supported as follows:
    - Only the **transmit** option for the *action* keyword is supported with the **conform-action** command.
    - Only the **set-frde-transmit** option for the *action* keyword is supported with the **exceed-action** command.
    - Only the **drop** option for the *action* keyword is supported with the **violate-action** command.
    - Backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN) configuration are not supported.
    - The type of service (ToS) byte must be configured in IP headers of tunneled Frame Relay packets when you configure the L2TPv3 pseudowire (see Configuring the L2TPv3 Pseudowire, page 58).
    - All standard restrictions for configuring QoS on Cisco 12000 series line cards apply to configuring QoS for L2TPv3 on Cisco 12000 series 2-port Channelized OC-3/STM-1 (DS1/E1) or 6-port Channelized T3 line cards.
  - On the ingress side of a Cisco 12000 series Frame Relay interface configured for TSC-based L2TPv3 tunneling:
    - Weighted random early detection (WRED) and modified deficit round robin (MDRR) configurations are not supported.
  - On the egress side of a Cisco 12000 series Frame Relay interface configured for TSC-based L2TPv3 tunneling:
    - MDRR is the only queueing strategy supported.
    - WRED is the only packet drop strategy supported.
    - MDRR is supported only in the following modes:

      —With both a low latency (priority) queue and class-default queue configured. (The low latency queue is supported only in combination with the class-default queue, and cannot be configured with normal distributed round robin (DRR) queues.)

      —Without a low latency queue configured. (In this case, only six queues are supported, including the class-default queue.)

- Egress queueing is determined according to the IP precedence values configured for classes of L2TPv3 Frame Relay traffic using the **match ip precedence** command, instead of on a per-DLCI basis.

For an example, see Configuring QoS on a Frame Relay Interface in a TSC-Based L2TPv3 Tunnel Session, page 101.

Edge Engine (ISE/E5) Interface

On the Cisco 12000 series, a QoS policy is supported in native L2TPv3 tunnel sessions on ISE/E5 interfaces (see Table 2 and Table 3 on page 16 for a list of supported line cards) with the following restrictions:

- On a Frame Relay or ATM ISE/E5 interface, traffic policing supports only the following conform, exceed, and violate values for the *action* argument of the **police** command:

  conform-action *actions*:
  set-prec-tunnel
  set-dscp-tunnel
  transmit

  exceed-action *actions*:
  drop
  set-clp (ATM only)
  set-dscp-tunnel
  set-dscp-tunnel and **set-clp** (ATM only)
  set-dscp-tunnel and **set-frde** (Frame Relay only)
  set-frde (Frame Relay only)
  set-prec-tunnel
  set-prec-tunnel and **set-clp** (ATM only)
  set-prec-tunnel and **set-frde** (Frame Relay only)
  transmit

  violate-action *actions*:
  drop

- On a Frame Relay ISE/E5 interface:
  - FECN and BECN configuration are not supported.
  - Marking the Frame Relay discard eligible (DE) bit using a MQC **set** command is not supported. To set (mark) the DE bit, use the **police exceed-action** *actions* command in policy-map configuration mode.
  - Configuring Tofab MDRR or WRED using legacy QoS (not MQC) commands is supported and is based on the tunnel precedence value.
  - Egress queueing on a Packet-over-SONET ISE/E5 interface is class-based when configured using MQC.
  - Egress queueing on a per-DLCI basis is not supported.

- On an ATM ISE/E5 interface:
  - Traffic shaping is supported on ATM egress interfaces for the following service categories:
    - Lowest priority: UBR (unspecified bit rate)
      Second priority: VBR-nrt (variable bit rate nonreal-time)
      Highest priority: VBR-rt (VBR real time)
      Highest priority: CBR (constant bit rate)
    - Note that VBR-rt and CBR share the same high-priority shaping. ATM traffic shaping restricts traffic to the maximum rate configured on an ATM VC or PVP with due priority among the respective service categories.
    - You can configure queue limits for an ATM VC or PVP. The queue limits are dual thresholds in which two different thresholds can be configured for CLP=1 cells and CLP0+1 cells. The CLP1 threshold must be lower than the queue limit threshold so that CLP=1 cells are dropped earlier than CLP=0 cells when packets start to fill the queue.
  - Although the dual-rate, 3-Color Marker policer is not supported on ATM ISE/E5 interfaces (as on Frame Relay ISE/E5 interfaces), the ATM Forum Traffic Management Version 4.1-compliant Generic Cell Rate Algorithm (GCRA) policer is supported. The GCRA policer uses rate, peak rate, delay tolerance, and ATM maximum burst size, and supports the following actions:

    set-dscp-tunnel
    set-dscp-tunnel and set-clp-transmit
    set-prec-tunnel
    set-prec-tunnel and set-clp-transmit

Protocol Demultiplexing Interface

- 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:

  ```
  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 Tunnel Protocol Version 3

To configure the Layer 2 Tunnel Protocol Version 3 feature, you must understand the following concepts:

- Migration from UTI to L2TPv3, page 29
- L2TPv3 Operation, page 29
- Benefits of Using L2TPv3, page 31
- L2TPv3 Header Description, page 31
- L2TPv3 Features, page 32
- L2TPv3 and UTI Feature Comparison, page 39
- Supported L2TPv3 Payloads, page 41

Migration from UTI to L2TPv3

UTI is a Cisco proprietary protocol that offers a simple high-speed transparent Layer 2-to-Layer 2 service over an IP backbone. The UTI protocol lacks the signaling capability and standards support necessary for large-scale commercial service. To begin to answer the need for a standard way to provide large-scale VPN connectivity over an IP core network, limited migration from UTI to L2TPv3 was introduced in Cisco IOS Release 12.0(21)S. The L2TPv3 feature in Cisco IOS Release 12.0(23)S introduced a more robust version of L2TPv3 to replace UTI.

As described in the section "L2TPv3 Header Description," the UTI data header is identical to the L2TPv3 header but with no sequence numbers and an 8-byte cookie. By manually configuring an L2TPv3 session using an 8-byte cookie (see the section "Manually Configuring L2TPv3 Session Parameters") and by setting the IP protocol number of outgoing data packets to 120 (as described in the section "Configuring the L2TPv3 Pseudowire"), you can ensure that a PE running L2TPv3 may interoperate with a peer PE running UTI. However, because UTI does not define a signaling plane, dynamically established L2TPv3 sessions cannot interoperate with UTI.

When a customer upgrades from a pre-L2TPv3 Cisco IOS release to a post-L2TPv3 release, an internal UTI-to-xconnect command-line interface (CLI) migration utility will automatically convert the UTI commands to xconnect and pseudowire class configuration commands without the need for any user intervention. After the CLI migration, the UTI commands that were replaced will not be available. The old-style UTI CLI is hidden from the user.

Note The UTI keepalive feature will *not* be migrated. The UTI keepalive feature will no longer be supported in post-L2TPv3 releases. You should convert to using dynamic L2TPv3 sessions to preserve the functionality provided by the UTI keepalive.

L2TPv3 Operation

L2TPv3 provides similar and enhanced services to replace the current UTI implementation, including the following features:

- Xconnect for Layer 2 tunneling through a pseudowire over an IP network
- Layer 2 VPNs for PE-to-PE router service using xconnect that supports Ethernet, 802.1q (VLAN), Frame Relay, HDLC, and PPP Layer 2 circuits, including both static (UTI-like) and dynamic (using the new L2TPv3 signaling) forwarded sessions

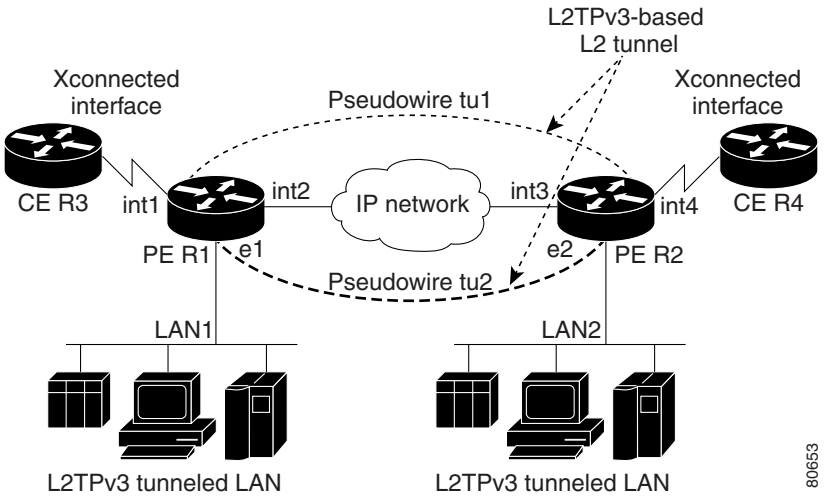
The initial Cisco IOS Release 12.0(23)S features supported only the following features:

- 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 User Datagram Protocol (UDP)
- Point-to-point sessions, not point-to-multipoint or multipoint-to-point sessions
- Sessions between the same Layer 2 protocols; for example, Ethernet-to-Ethernet, VLAN-to-VLAN, but not VLAN-to-Ethernet or Frame Relay

The attachment circuit is the physical interface or subinterface attached to the pseudowire.

Figure 1 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 routers of the IP network treat the traffic as any other IP traffic and need not know anything about the customer networks.

Figure 1 L2TPv3 Operation—Example



In Figure 1, the PE routers R1 and R2 provide L2TPv3 services. The R1 and R2 routers communicate with each other using a pseudowire over the IP backbone network through a path comprising the interfaces int1 and int2, the IP network, and interfaces int3 and int4.

In this example, the CE routers R3 and R4 communicate through a pair of xconnect Ethernet or 802.1q VLAN interfaces using an L2TPv3 session. The L2TPv3 session tu1 is a pseudowire configured between interface int1 on R1 and interface int4 on R2. Any packet arriving on interface int1 on R1 is encapsulated and sent through the pseudowire control channel (tu1) to R2. R2 decapsulates the packet and sends it on interface int4 to R4. When R4 needs to send a packet to R3, the packet follows the same path in reverse.

Note the following features regarding L2TPv3 operation:

- 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 are 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.

Benefits of Using L2TPv3

L2TPv3 Simplifies Deployment of VPNs

L2TPv3 is an industry-standard Layer 2 tunneling protocol that ensures interoperability among vendors, increasing customer flexibility and service availability.

L2TPv3 Does Not Require MPLS

With L2TPv3 service providers need not deploy MPLS in the core IP backbone to set up VPNs using L2TPv3 over the IP backbone, resulting in operational savings and increased revenue.

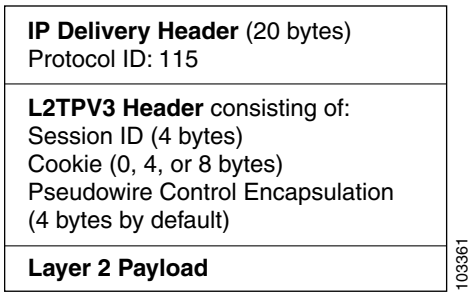
L2TPv3 Supports Layer 2 Tunneling over IP for Any Payload

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.

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 Figure 2.

Figure 2 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 section "How to Configure Layer 2 Tunnel Protocol Version 3" for more information on the CLI commands for L2TPv3.

Session ID

The L2TPv3 session ID is similar to the UTI session ID, and 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 that is similar to the UTI control channel key field. The control channel cookie field, however, 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 manually configured for static sessions, or dynamically determined 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 section "Sequencing") and to distinguish AAL5 data and OAM cells for AAL5 SDU mode over L2TPv3. 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 Features

L2TPv3 provides xconnect support for Ethernet, 802.1q (VLAN), Frame Relay, HDLC, and PPP, using the sessions described in the following sections:

- Static L2TPv3 Sessions (nonnegotiated, PVC-like forwarded sessions)
- Dynamic L2TPv3 Sessions (negotiated, forwarded sessions using the L2TPv3 control plane for session negotiation)

L2TPv3 also includes support for the features described in the following sections:

- Sequencing
- Local Switching
- Distributed Switching
- L2TPv3 Layer 2 Fragmentation
- L2TPv3 Type of Service Marking
- Keepalive
- MTU Handling
- L2TPv3 Control Message Hashing
- L2TPv3 Control Message Rate Limiting
- L2TPv3 Digest Secret Graceful Switchover
- Manual Clearing of L2TPv3 Tunnels
- L2TPv3 Tunnel Management
- L2TPv3 Protocol Demultiplexing
- Color Aware Policer on Ethernet over L2TPv3
- Site of Origin for Border Gateway Protocol VPNs

Static L2TPv3 Sessions

Typically, the L2TP control plane is responsible for negotiating session parameters (such as the session ID or the cookie) to set up the session. However, some IP networks require sessions to be configured so that no signaling is required for session establishment. Therefore, you can set up static L2TPv3 sessions for a PE router by configuring fixed values for the fields in the L2TP data header. A static L2TPv3 session allows the PE to tunnel Layer 2 traffic as soon as the attachment circuit to which the session is bound comes up.

Note In an L2TPv3 static session, you can still run the L2TP control channel to perform peer authentication and dead-peer detection. If the L2TP control channel cannot be established or is torn down because of a hello failure, the static session is also torn down.

When you use a static L2TPv3 session, you cannot perform circuit interworking, 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: the payload type, virtual circuit (VC) ID, and so on.

Multiple L2TP sessions (one for each forwarded Layer 2 circuit) can exist between a pair of PEs, and can be maintained by a single control channel. Session IDs and cookies are dynamically generated and exchanged as part of a dynamic session setup. Information such as sequencing configuration is also exchanged. Circuit state changes (UP/DOWN) are conveyed using the set link info (SLI) message.

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 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 that receives this AV pair (or that is manually configured to send sequenced packets) uses the Layer 2-specific pseudowire control encapsulation defined in L2TPv3.

You can configure L2TP to only drop out-of-order packets; you cannot configure L2TP to deliver the packets out-of-order. No reordering mechanism is available.

Cisco IOS Release 12.0(28)S and Cisco IOS Release 12.2(25)S introduced support for L2TPv3 distributed sequencing on the Cisco 7500 series routers only.

Local Switching

Local switching (from one port to another port in the same router) is supported for both static and dynamic sessions. You must configure separate IP addresses for each xconnect statement.

See the section "Configuration Examples for Layer 2 Tunnel Protocol Version 3" for an example of how to configure local port switching.

Distributed Switching

Distributed CEF switching is supported for L2TP on the Cisco 7500 series routers.

Note For the Cisco 7500 series, sequencing is supported, but all L2TP packets that require sequence number processing are sent to the RSP.

L2TPv3 Layer 2 Fragmentation

Because the reassembly of fragmented packets is computationally expensive, it is desirable to avoid fragmentation issues in the service provider network. The easiest way to avoid fragmentation issues is to configure the CE routers with an path maximum transmission unit (MTU) value that is smaller than the pseudowire path MTU. However, in scenarios where this is not an option, fragmentation issues must be considered. L2TP initially supported only the following options for packet fragmentation when a packet is determined to exceed the L2TP path MTU:

- Unconditionally drop the packet
- Fragment the packet after L2TP/IP encapsulation
- Drop the packet and send an Internet Control Message Protocol (ICMP) unreachable message back to the CE router

The L2TPv3 Layer 2 Fragmentation feature introduces the ability to allow IP traffic from the CE router to be fragmented before the data enters the pseudowire, forcing the computationally expensive reassembly to occur in the CE network rather than in the service-provider network. The number of fragments that must be generated is determined based on the discovered pseudowire path MTU.

To enable the discovery of the path MTU for Layer 2 traffic, enter the **ip pmtu** command in a pseudowire class configuration (see "Configuring the L2TPv3 Pseudowire" section on page 58). On the PE router, the original Layer 2 header is then copied to each of the generated fragments, the L2TP/IP encapsulation is added, and the frames are forwarded through the L2TPv3 pseudowire.

Because the Don't Fragment (DF) bit in the Layer 2 encapsulation header is copied from the inner IP header to the encapsulation header, fragmentation of IP packets is performed on any packets received from the CE network that have a DF bit set to 0 and that exceed the L2TP path MTU in size. To prevent the reassembly of fragmented packets on the decapsulation router, you can enter the **ip dfbit set** command in the pseudowire class configuration to enable the DF bit in the outer Layer 2 header.

L2TPv3 Type of Service Marking

When Layer 2 traffic is tunneled across an IP network, information contained in the ToS bits may be transferred to the L2TP-encapsulated IP packets in one of the following ways:

- If the tunneled Layer 2 frames encapsulate IP packets themselves, it may be desirable to simply copy the ToS bytes of the inner IP packets to the outer IP packet headers. This action is known as "ToS byte reflection."
- Static ToS byte configuration. You specify the ToS byte value used by all packets sent across the pseudowire.

See the section "Configuring a Negotiated L2TPv3 Session for Local HDLC Switching: Example" for more information about how to configure ToS information.

Keepalive

The keepalive mechanism for L2TPv3 extends only to the endpoints of the tunneling protocol. L2TP has a reliable control message delivery mechanism that serves as the basis for the keepalive mechanism. The keepalive mechanism consists of an exchange of L2TP hello messages.

If a keepalive mechanism is required, the control plane is used, although it may not be used to bring up sessions. You can manually configure sessions.

In the case of static L2TPv3 sessions, a control channel between the two L2TP peers is negotiated through the exchange of start control channel request (SCCRQ), start control channel replay (SCCRP), and start control channel connected (SCCCN) control messages. The control channel is responsible only for maintaining the keepalive mechanism through the exchange of hello messages.

The interval between hello messages is configurable per control channel. If one peer detects that the other has gone down through the keepalive mechanism, it sends a StopCCN control message and then notifies all of the pseudowires to the peer about the event. This notification results in the teardown of both manually configured and dynamic sessions.

MTU Handling

It is important that you configure an MTU appropriate for a 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 discovery. When you enable this feature, the following processing is performed:
  - ICMP unreachable messages sent back to the L2TPv3 router are deciphered and the tunnel MTU is updated accordingly. To receive ICMP unreachable messages for fragmentation errors, the DF bit in the tunnel header is set according to the DF bit value received from the CE, or 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 Challenge Handshake Authentication Protocol (CHAP)-like authentication system inherited from L2TPv2, which uses the Challenge and Challenge Response AV pairs in the SCCRQ, SCCRP, and SCCCN 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, and guard against control message spoofing and replay attacks that would otherwise be trivial to mount against the network.

The L2TPv3 Control Message Hashing feature incorporates an optional authentication or integrity check for all control messages. The new authentication method uses a computed one-way hash over the header and body of the L2TP control message, a pre-configured shared secret that must be defined on communicating L2TP nodes, and 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 router, control messages are sent with the message digest calculated without the shared secret or Nonce AV pairs, and are verified by the remote PE router. If verification fails, the remote PE router drops the control message.

L2TPv3 Control Message Rate Limiting

The L2TPv3 Control Message Rate Limiting feature was introduced to counter the possibility of a denial-of-service attack on a router 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 the control plane resources of the PE router.

On distributed platforms, most control packet filtering occurs at the line card level, and the CPU of the RP is minimally impacted even in a worst-case denial-of-service attack scenario. This feature has minimal impact on the shared bus or switching fabric, which are typically the bottleneck of a router.

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 routers. In Cisco IOS releases earlier than Release 12.0(30)S, changing this password requires removing 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 routers, which are often at different physical locations. It is difficult for all peer PE routers be updated with the new password simultaneously to minimize interruptions in L2TPv3 services.

Cisco IOS Release 12.0(30)S introduces the L2TPv3 Digest Secret Graceful Switchover feature. This feature allows the password used to authenticate L2TPv3 control channel messages to be changed without tearing down 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 continues to use the old password until it is removed from the configuration. This allows authentication to continue normally with peer PE routers that have not yet been updated to use the new password. After all peer PE routers are configured with the new password, the old password can be removed from the configuration.

Manual Clearing of L2TPv3 Tunnels

Cisco IOS Release 12.0(30)S introduces the ability to clear L2TPv3 tunnels manually. In Cisco IOS releases earlier than Release 12.0(30)S, no provision was made to manually clear a specific L2TPv3 tunnel at will. This functionality provides users more control over an L2TPv3 network.

L2TPv3 Tunnel Management

New and enhanced commands have been introduced to facilitate managing xconnect configurations and diagnosing problems with xconnect configurations. No specific configuration tasks are associated with these commands.

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.

New and enhanced commands were introduced in the following releases:

- Syslog, SNMP Trap, and show Command Enhancements for L2TPv3 in Cisco IOS Release 12.0(31)S, Cisco IOS Release 12.2(27)SBC.
- Control Message Statistics and Conditional Debugging Command Enhancements

Syslog, SNMP Trap, and show Command Enhancements for L2TPv3 in Cisco IOS Release 12.0(31)S, Cisco IOS Release 12.2(27)SBC.

Cisco IOS Release 12.0(31)S and Cisco IOS Release 12.2(27)SBC introduce new or enhanced commands for managing and diagnosing problems with xconnect configurations:

- **debug vpdn**—The output of this command includes authentication failure messages.
- **show l2tun session**—The **hostname** keyword option allows the peer hostname to be displayed in the output.
- **show l2tun tunnel**—The **authentication** keyword option allows the display of global information about L2TP control channel authentication attribute-value pairs (AV pairs).
- **show xconnect**—Displays xconnect-specific information, providing a sortable single point of reference for information about all xconnect configurations.
- **snmp-server enable traps l2tun pseudowire status**—Enables the sending of Simple Network Management Protocol (SNMP) notifications when a pseudowire changes state.
- **xconnect logging pseudowire status**—Enables syslog reporting of pseudowire status events.

Control Message Statistics and Conditional Debugging Command Enhancements

This feature introduces new commands and modifies existing commands for managing control message statistics and conditionally filtering xconnect debug messages.

For this feature, the following commands were introduced:

- **clear l2tun counters**—Clears session counters for Layer 2 tunnels.
- **clear l2tun counters tunnel l2tp**—Clears global or per-tunnel control message statistics.
- **debug condition xconnect**—Allows the conditional filtering of debug messages related to xconnect configurations (allows pseudowire conditional debugging)
- **monitor l2tun counters tunnel l2tp**—Enables or disables the collection of per-tunnel control message statistics.
- **show l2tun counters tunnel l2tp**—Displays global or per-tunnel control message statistics.

For this feature, the following command was modified:

- **show l2tun tunnel**—The **authentication** keyword was removed. The statistics previously displayed by the **show l2tun tunnel authentication** command are now displayed by the **show l2tun counters tunnel l2tp authentication** command.

L2TPv3 Protocol Demultiplexing

The 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. IPv6 traffic is transparently tunneled to the IPv6 network using L2TPv3 pseudowires without affecting the configuration of the CE routers. IPv4 traffic is routed as usual within the IPv4 network, maintaining the existing performance and reliability of the IPv4 network.

The IPv4 PE routers must be configured to demultiplex incoming IPv6 traffic from IPv4 traffic. The PE routers facing the IPv6 network do not require demultiplexing configuration. The configuration of the IPv6 network is beyond the scope of this document. For more information on configuring an IPv6 network, refer to the *Cisco IOS IPv6 Configuration Guide*.

Color Aware Policer on Ethernet over L2TPv3

The QoS: Color-Aware Policer was introduced in Cisco IOS Release 12.0(29)S. Cisco IOS Release 12.0(33)S provides support for the Color Aware Policer on Engine-3/Engine-5 line cards for Ethernet over L2TPv3.

The Color-Aware Policer enables a “color-aware” method of traffic policing. This feature allows you to police traffic according to the color classification of a packet. The packet color classification is based on packet matching criteria defined for two user-specified traffic classes—the conform-color class and the exceed-color class. These two traffic classes are created using the conform-color command and the metering rates are defined using the police command.

Site of Origin for Border Gateway Protocol VPNs

Site of Origin (SoO) for Border Gateway Protocol Virtual Private Networks (BGP-VPNs) is supported in Cisco IOS Release 12.0(33)S. Site of Origin (SoO) is a concept in a distributed VPN architecture that prevents routing loops in a site which is multi-homed to the VPN backbone and uses AS-OVERRIDE. The mechanism works by applying the SoO tag at the VPN entry point, the provider's edge (PE) equipment. When SoO is enabled, the PE only forwards prefixes to the customer premises equipment (CPE) when the SoO tag of the prefix does not match the SoO tag configured for the CPE.

Each site should be assigned a unique ID value, which is used as the second half of the SoO tag. These ID values used can be repeated for different customers, but not for the same customer. A “site” is considered SoO enabled if it has two or more CPEs that are connected to different PEs and includes at least one non-PE link between them.

SoO is a BGP extended community attribute used to identify when a prefix that originated from a customer site is re-advertised back into that site from a backdoor link. The following format can be used to address the SoO extended community:

<Customer-AS>:<Site-ID>

SoO can now be configured either using inbound route-maps or using the per-neighbor **neighbor soo** command. The SoO value set through the **neighbor soo** command should override the legacy inbound route-map settings when both are configured at the same time.

L2TPv3 and UTI Feature Comparison

Table 5 compares L2TPv3 and UTI feature support for the Cisco 7200 and Cisco 7500 series routers.

Table 5 Comparison of L2TPv3 and UTI Feature Support

Feature	L2TPv3	UTI
Maximum number of sessions	Cisco 7200 and Cisco 7500 series:3000	Cisco 7200 and Cisco 7500 series: 1000
Tunnel cookie length	0-, 4-, or 8-byte cookies are supported for the Cisco 7200 series and the Cisco 7500 series routers.	8 bytes
Static sessions	Supported in Cisco IOS Release 12.0(21)S.	Supported
Dynamic sessions	Supported in Cisco IOS Release 12.0(23)S.	Not supported
Static ToS	Supported in Cisco IOS Release 12.0(23)S.	Supported
MQC ToS	Supported in Cisco IOS Release 12.0(27)S.	Supported
Inner IP ToS mapping	Supported on the Cisco 7200 series routers and Cisco 7500 series routers.	Not supported
802.1p mapping	Not supported.	Not supported
Keepalive	Supported in Cisco IOS Release 12.0(23)S.	Not supported
Path MTU discovery	Supported on the Cisco 7200 series and Cisco 7500 series routers.	Not supported
ICMP unreachable	Supported on the Cisco 7200 series and Cisco 7500 series routers.	Not supported
VLAN rewrite	Supported on the Cisco 7200 series and Cisco 7500 series routers in Cisco IOS Release 12.0(23)S.	Supported
VLAN and non-VLAN translation	To be supported in a future release.	Not supported
Port trunking	Supported in Cisco IOS Release 12.0(23)S.	Supported
IS-IS packet fragmentation through an L2TPv3 session	Supported on the Cisco 7200 series and Cisco 7500 series routers, and on the Cisco 10720 Internet router in Cisco IOS Release 12.0(24)S.	Not supported
L2TPv3 Layer 2 (IP packet) fragmentation through an L2TPv3 session	Supported on the Cisco 7200 series and Cisco 7500 series routers in Cisco IOS Release 12.0(24)S. Supported on the Cisco 10720 Internet router in Cisco IOS Release 12.0(32)SY.	Not supported
Payload sequence number checking	Supported on the Cisco 7500 series in Cisco IOS Release 12.0(28)S.	Not supported
MIB support	VPDN MIB for the pseudowire IfTable MIB for the attachment circuit.	IfTable MIB for the session interface.

Supported L2TPv3 Payloads

L2TPv3 supports the following Layer 2 payloads that can be included in L2TPv3 packets tunneled over the pseudowire:

- Frame Relay
- Ethernet
- 802.1q (VLAN)
- HDLC
- PPP
- ATM
- IPv6 Protocol Demultiplexing

Note Each L2TPv3 tunneled packet includes the entire Layer 2 frame of the payloads described in this section. If sequencing is required (see the section "Sequencing"), a Layer 2-specific sublayer (see the section "Pseudowire Control Encapsulation") is included in the L2TPv3 header to provide the Sequence Number field.

Frame Relay

L2TPv3 supports the Frame Relay functionality described in the following sections:

- Port-to-Port Trunking
- DLCI-to-DLCI Switching
- PVC Status Signaling
- Sequencing
- ToS Marking
- CIR Guarantees
- Binding L2TPv3 Sessions to Multilink Frame Relay Interfaces

Port-to-Port Trunking

Port-to-port trunking is where two CE Frame Relay interfaces are connected as by a leased line (UTI raw mode). All traffic arriving on one interface is forwarded transparently across the pseudowire to the other interface.

For example, in Figure 1, if the two CE routers are connected by a virtual leased line, the PE routers transparently transport all packets between CE R3 and CE R4 over a pseudowire. PE R1 and PE R2 do not examine or change the DLCIs, and do not participate in the LMI protocol. The two CE routers are LMI peers. There is nothing Frame Relay-specific about this service as far as the PE routers are concerned. The CE routers should be able to use any encapsulation based on HDLC framing without needing to change the provider configuration.

DLCI-to-DLCI Switching

Frame Relay DLCI-to-DLCI switching is where individual Frame Relay DLCIs are connected to create an end-to-end Frame Relay PVC. Traffic arriving on a DLCI on one interface is forwarded across the pseudowire to another DLCI on the other interface.

For example, in Figure 1, CE R3 and PE R1 are Frame Relay LMI peers; CE R4 and PE R2 are also LMI peers. You can use a different type of LMI between CE R3 and PE R1 compared to what you use between CE R4 and PE R2.

The CE devices may be a Frame Relay switch or end-user device. Each Frame Relay PVC is composed of multiple segments. The DLCI value is local to each segment and is changed as traffic is switched from segment to segment. Note that, in Figure 1, two Frame Relay PVC segments are connected by a pseudowire. Frame Relay header flags (FECN, BECN, C/R, DE) are preserved across the pseudowire.

PVC Status Signaling

PVC status signaling is propagated toward Frame Relay end users by the LMI protocol. You can configure the LMI to operate in any of the following modes:

- UNI DTE mode—PVC status is not reported, only received.
- UNI DCE mode—PVC status is reported but not received.
- NNI mode—PVC status is reported and received independently.

L2TPv3 supports all three modes.

The PVC status should be reported as ACTIVE only if the PVC is available from the reporting device to the Frame Relay end-user device. All interfaces, line protocols, and pseudowires must be operational between the reporting device and the Frame Relay end-user device.

Note that any keepalive functions on the session are independent of Frame Relay, but any state changes that are detected are fed into the PVC status reporting. For example, the L2TP control channel uses hello packets as a keepalive function. If the L2TPv3 keepalive fails, all L2TPv3 sessions are torn down. Loss of the session is notified to Frame Relay, which can then report PVCs INACTIVE to the CE devices.

For example, in Figure 1, CE R3 reports ACTIVE to PE R1 only if the PVC is available within CE R3. When CE R3 is a switch, it reports all the way to the user device in the customer network.

PE R1 reports ACTIVE to CE R3 only if the PVC is available within PE R1 and all the way to the end-user device (through PE R2 and CE R3) in the other customer VPN site.

The ACTIVE state is propagated hop-by-hop, independently in each direction, from one end of the Frame Relay network to the other end.

Sequencing

Frame Relay provides an ordered service in which packets sent to the Frame Relay network by one end-user device are delivered in order to the other end-user device. When switching is occurring over the pseudowire, packet ordering must be able to be preserved with a very high probability to closely emulate a traditional Frame Relay service. If the CE router is not using a protocol that can detect misordering itself, configuring sequence number processing may be important. For example, if the Layer 3 protocol is IP and Frame Relay is therefore used only for encapsulation, sequencing is not required. To detect misordering, you can configure sequence number processing separately for transmission or reception. For more information about how to configure sequencing, see the section "Configuring a Negotiated L2TPv3 Session for Local HDLC Switching: Example."

ToS Marking

The ToS bytes in the IP header can be statically configured or reflected from the internal IP header. The Frame Relay discard eligible (DE) bit does not influence the ToS bytes.

CIR Guarantees

To provide committed information rate (CIR) guarantees, you can configure a queueing policy that provides bandwidth to each DLCI to the interface facing the customer network on the egress PE.

Note CIR guarantees are supported only on the Cisco 7500 series with dCEF. This support requires that the core has sufficient bandwidth to handle all CE traffic and that the congestion occurs only at the egress PE.

Binding L2TPv3 Sessions to Multilink Frame Relay Interfaces

The configuration of an L2TPv3 session on a Multilink Frame Relay (MLFR) bundle interface is supported only on Cisco 12000 series 2-port channelized OC-3/STM-1 (DS1/E1) and 6-port channelized T3 (T1) line cards.

The Multilink Frame Relay feature introduces functionality based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16). This feature provides a cost-effective way to increase bandwidth for particular applications by enabling multiple serial links to be aggregated into a single bundle of bandwidth.

For an example of how to configure L2TPv3 tunneling on a multilink Frame Relay bundle interface, see Configuring MLFR for L2TPv3 on the Cisco 12000 Series: Example, page 108.

For information about how configure and use the MLFR feature, refer to the *Multilink Frame Relay (FRF.16)* publication.

Ethernet

An Ethernet frame arriving at a PE router 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 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 router. All frames are tunneled through the L2TP pseudowire.

802.1q (VLAN)

L2TPv3 supports VLAN membership 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 802.1q VLAN packets, the Ethernet interface must be configured in promiscuous mode to capture all traffic received on the Ethernet segment attached to the router. All frames are tunneled through the L2TP pseudowire.

HDLC

L2TPv3 encapsulates an HDLC frame arriving at a PE in its entirety (including the Address, Control, and Protocol fields, but not the Flag fields and the frame check sequence) with an L2TP data header.

PPP

PEs that support L2TPv3 forward PPP traffic using a “transparent pass-through” model, in which the PEs play no role in the negotiation and maintenance of the PPP link. L2TPv3 encapsulates a PPP frame arriving at a PE in its entirety (including the HDLC Address and Control fields) with an L2TP data header.

ATM

L2TPv3 can connect two isolated ATM clouds over a packet-switched network (PSN) while maintaining an end-to-end ATM Service Level Agreement (SLA). The ATM Single Cell Relay features forward one ATM cell per packet. The ATM Cell Packing over L2TPv3 features allows multiple ATM frames to be packed into a single L2TPv3 data packet. All packets are transparently forwarded over the L2TPv3 pseudowire.

Note VPI or VPI/VCI rewrite is not supported for any ATM transport mode. Both pairs of PE to CE peer routers must be configured with matching VPI or VCI values except in OAM local emulation mode. For example, if PE1 and CE1 are connected by PVC 10/100, PE2 and CE2 should also be connected by PVC 10/100.

Table 6 shows the releases that introduced support for the ATM cell relay features.

Table 6 Release Support for the ATM Cell Relay Features

Transport Type	Single Cell Relay	Packed Cell Relay
VC mode	12.0(28)S, 12.2(25)S	12.0(29)S

Table 6 Release Support for the ATM Cell Relay Features

Transport Type	Single Cell Relay	Packed Cell Relay
VP mode	12.0(25)S, 12.2(25)S	12.0(29)S
Port mode	12.0(29)S, 12.2(25)S4	12.0(29)S

ATM Single Cell Relay VC Mode over L2TPv3

The ATM Single Cell Relay VC mode over L2TPv3 feature maps one VC to a single L2TPv3 session. All ATM cells arriving at an ATM interface with the specified VPI and VCI are encapsulated into a single L2TP packet. Each ATM cell will have a 4-byte ATM cell header without Header Error Control Checksum (HEC) and a 48-byte ATM cell payload.

The ATM Single Cell Relay VC mode feature can be used to carry any type of AAL traffic over the pseudowire. It will not distinguish OAM cells from User data cells. In this mode, Performance and Security OAM cells are also transported over the pseudowire.

ATM VP Mode Single Cell Relay over L2TPv3

The ATM VP Mode Single Cell Relay over L2TPv3 feature allows cells coming into a predefined PVP on the ATM interface to be transported over an L2TPv3 pseudowire to a predefined PVP on the egress ATM interface. A single ATM cell is encapsulated into each L2TPv3 data packet.

ATM Port Mode Cell Relay over L2TPv3

The ATM Port Mode Cell Relay over L2TPv3 feature packs ATM cells arriving at an ingress ATM interface into L2TPv3 data packets and transports them to the egress ATM interface. A single ATM cell is encapsulated into each L2TPv3 data packet.

ATM Cell Packing over L2TPv3

The ATM Cell Packing over L2TPv3 feature enhances throughput and uses bandwidth more efficiently than the ATM cell relay features. Instead of a single ATM cell being packed into each L2TPv3 data packet, multiple ATM cells can be packed into a single L2TPv3 data packet. ATM cell packing is supported for Port mode, VP mode, and VC mode. Cell packing must be configured on the PE devices. No configuration is required on the CE devices.

ATM AAL5 over L2TPv3

The ATM AAL5 over L2TPv3 feature maps the AAL5 payload of an AAL5 PVC to a single L2TPv3 session. This service will transport OAM and RM cells, but does not attempt to maintain the relative order of these cells with respect to the cells that comprise the AAL5 common part convergence sublayer protocol data unit (CPCS-PDU). OAM cells that arrive during the reassembly of a single AAL5 CPCS-PDU are sent immediately over the pseudowire, followed by the AAL5 payload without the AAL5 pad and trailer bytes.

VC Class Provisioning for L2TPv3

Beginning in Cisco IOS Release 12.0(30)S, ATM AAL5 encapsulation over L2TPv3 can be configured in VC class configuration mode in addition to ATM VC configuration mode. The ability to configure ATM encapsulation parameters in VC class configuration mode provides greater control and flexibility for AAL5 encapsulation configurations.

OAM Transparent Mode

In OAM transparent mode, the PEs will pass the following OAM cells transparently across the pseudowire:

- F5 segment and end-to-end Fault Management (FM) OAM cells
- RM OAM cells, except Performance Management (PM) and Security OAM cells

Note The Cisco 7200 and the Cisco 7500 ATM driver cannot forward RM cells over the PSN for ABR ToS. The RM cells are locally terminated.

VPI or VPI/VCI rewrite is not supported for any ATM transport mode. Both pairs of PE to CE peer routers must be configured with matching VPI and VCI values except in OAM local emulation mode. For example, if PE1 and CE1 are connected by PVC 10/100, PE2 and CE2 should also be connected by PVC 10/100.

OAM Local Emulation Mode

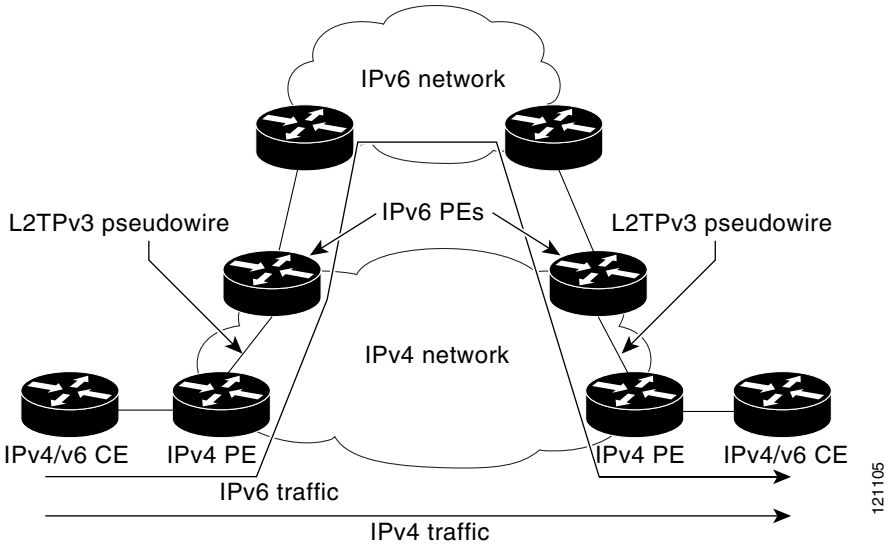
In OAM Local Emulation mode, OAM cells are not passed through the pseudowire. All F5 OAM cells are terminated and handled locally. On the L2TPv3-based pseudowire, the CE device sends an SLI message across the pseudowire to notify the peer PE node about the defect, rather than tearing down the session. The defect can occur at any point in the link between the local CE and the PE. OAM management can also be enabled on the PE node using existing OAM management configurations.

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 transparently tunneled to the IPv6 network using L2TPv3 pseudowires without affecting the configuration of the CE routers. IPv4 traffic is routed as usual within the IPv4 network, maintaining the existing performance and reliability of the IPv4 network.

Figure 3 shows a network deployment that offloads IPv6 traffic from the IPv4 network to a specialized IPv6 network. The PE routers 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 routers must be configured to demultiplex incoming IPv6 traffic from IPv4 traffic. The PE routers facing the IPv6 network do not require demultiplexing configuration.

Figure 3 Protocol Demultiplexing of IPv6 Traffic from IPv4 Traffic



IPv6 protocol demultiplexing is supported only for Ethernet and Frame Relay traffic on Cisco 7500 series routers beginning in Cisco IOS Release 12.0(29)S and Cisco IOS Release 12.2(27)SBC. Cisco IOS Release 12.0(33)S adds IPv6 protocol demultiplexing support for PPP and HDLC traffic to the Cisco 7500 series routers. Cisco IOS Release 12.0(33)S adds IPv6 protocol demultiplexing support for all transports on the Cisco 12000 series routers. Protocol demultiplexing requires supporting the combination of an IP address and an **xconnect** command configuration on the IPv4 PE interface. This combination of configurations is not allowed without enabling protocol demultiplexing, with the exception of switched Frame Relay PVCs. 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 enabled, the IP address cannot be removed. To change or remove the configured IP address, the **xconnect** command configuration must first be disabled.

Table 7 shows the valid combinations of configurations.

Table 7 Valid Configuration Scenarios

Scenario	IP Address	xconnect Configuration	Protocol Demultiplexing Configuration
Routing	Yes	No	—
L2VPN	No	Yes	No
IPv6 Protocol Demultiplexing	Yes	Yes	Yes

How to Configure Layer 2 Tunnel Protocol Version 3

This section contains the following procedures:

- Configuring L2TP Control Channel Parameters, page 48 (optional)
- Configuring the L2TPv3 Pseudowire, page 58 (required)

- Configuring the Xconnect Attachment Circuit, page 62 (required)
- Manually Configuring L2TPv3 Session Parameters, page 64 (required)
- Configuring the Xconnect Attachment Circuit for ATM VP Mode Single Cell Relay over L2TPv3, page 66 (optional)
- Configuring the Xconnect Attachment Circuit for ATM Single Cell Relay VC Mode over L2TPv3, page 67 (optional)
- Configuring the Xconnect Attachment Circuit for ATM Port Mode Cell Relay over L2TPv3, page 68 (optional)
- Configuring the Xconnect Attachment Circuit for ATM Cell Packing over L2TPv3, page 69 (optional)
- Configuring the Xconnect Attachment Circuit for ATM AAL5 SDU Mode over L2TPv3, page 74 (optional)
- Configuring OAM Local Emulation for ATM AAL5 over L2TPv3, page 78 (optional)
- Configuring Protocol Demultiplexing for L2TPv3, page 82 (optional)
- Manually Clearing L2TPv3 Tunnels, page 88 (optional)

Configuring L2TP 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 router at each end of the control channel. Configuring L2TP control channel parameters is optional. However, the L2TP class must be configured before it is with associated a pseudowire class (see the section "Configuring the L2TPv3 Pseudowire").

The three main groups of L2TP control channel parameters that you can configure in an L2TP class are described in the following sections:

- Configuring L2TP Control Channel Timing Parameters
- Configuring L2TPv3 Control Channel Authentication Parameters
- Configuring L2TP Control Channel Maintenance 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 L2TP class control channel parameters can be applied to a connection between any pair of IP addresses.

Configuring L2TP Control Channel Timing Parameters

The following L2TP control channel timing parameters can be configured in L2TP class configuration mode:

- Packet size of the receive window used for the control channel
- Retransmission parameters used for control messages
- Timeout parameters used for the control channel

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, the default values are applied.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l2tp-class** [*l2tp-class-name*]
4. **receive-window** *size*
5. **retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*}
6. **timeout setup** *seconds*

DETAILED STEPS

	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	**l2tp-class** [*l2tp-class-name*] **Example:** `Router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, if you want to configure multiple L2TP classes you must specify a unique *l2tp-class-name* for each one.
Step 4	**receive-window** *size* **Example:** `Router(config-l2tp-class)# receive-window 30`	(Optional) Configures the number of packets that can be received by the remote peer before backoff queueing occurs. • The valid values range from 1 to the upper limit the peer has for receiving packets. The default value is the upper limit.

	Command or Action	Purpose
Step 5	**retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*} **Example:** `Router(config-l2tp-class)# retransmit retries 10`	(Optional) Configures parameters that affect the retransmission of control packets. • **initial retries**—specifies how many SCCRQs are re-sent 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 router does not respond. 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 6	**timeout setup** *seconds* **Example:** `Router(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.

Configuring L2TPv3 Control Channel Authentication Parameters

Two methods of control channel message authentication are available beginning in Cisco IOS Release 12.0(29)S and Cisco IOS Release 12.2(27)SBC. 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 methods of authentication to ensure interoperability with peers that support only one of these methods of authentication, but this configuration will yield control of which authentication method is used to the peer PE router. Enabling both methods of authentication should be considered an interim solution to solve backward-compatibility issues during software upgrades.

The principal difference between the L2TPv3 Control Message Hashing feature and CHAP-style L2TP control channel authentication is that, instead of computing the hash over selected contents of a received control message, the L2TPv3 Control Message Hashing feature uses the entire message in the hash. In addition, instead of including the hash digest in only the SCCRP and SCCCN messages, it includes it in all messages.

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.

Table 8 shows a compatibility matrix for the different L2TPv3 authentication methods. PE1 is running Cisco IOS 12.0(29)S, and the different possible authentication configurations for PE1 are shown in the first column. Each remaining column represents PE2 running software with different available authentication options, and the intersections indicate the different compatible configuration options for PE2. If any PE1/PE2 authentication configuration poses ambiguity on which method of authentication is 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 8 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
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, such as Cisco IOS Release 12.0(29)S or Cisco IOS Release 12.2(27)SBC.

Perform one or both of the following tasks to configure authentication parameters for the L2TPv3 control channel messages:

- Configuring Authentication for the L2TP Control Channel, page 52 (optional)
- Configuring L2TPv3 Control Message Hashing, page 53 (optional)

If you choose to configure authentication using the L2TPv3 Control Message Hashing feature, you may perform the following optional task:

- Configuring L2TPv3 Digest Secret Graceful Switchover, page 55 (optional)

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, the 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*

DETAILED STEPS

	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	**l2tp-class** [*l2tp-class-name*] **Example:** `Router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, if you want to configure multiple L2TP classes you must specify a unique *l2tp-class-name* for each one.
Step 4	**authentication** **Example:** `Router(config-l2tp-class)# authentication`	(Optional) Enables authentication for the control channel between PE routers.

	Command or Action	Purpose
Step 5	**password** [**0** | **7**] *password* **Example:** `Router(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 routers.
Step 6	**hostname** *name* **Example:** `Router(config-l2tp-class)# hostname yb2`	(Optional) Specifies a hostname used to identify the router during L2TP control channel authentication. • If you do not use this command, the default hostname of the router is used.

Configuring L2TPv3 Control Message Hashing

The L2TPv3 Control Message Hashing feature introduced in Cisco IOS Release 12.0(29)S and Cisco IOS Release 12.2(27)SBC is a new authentication system that is more secure than the CHAP-style L2TP control channel method of authentication. L2TPv3 Control Message Hashing incorporates an optional authentication or integrity check for all control messages. This per-message authentication is designed to guard against control message spoofing and replay attacks that would otherwise be trivial to mount against the network.

Enabling the L2TPv3Control Message Hashing feature will impact performance during control channel and session establishment because additional digest calculation of the full message content is required for each sent and received control message. This is an expected trade-off for the additional security afforded by this feature. In addition, network congestion may occur if the receive window size is too small. If the L2TPv3 Control Message Hashing feature is enabled, message digest validation must be enabled. Message digest validation deactivates the data path received sequence number update and restricts the minimum local receive window size to 35.

You 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 routers. Control message integrity check is unidirectional, and requires configuration on only one of the peers.

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**

DETAILED STEPS

	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	**l2tp-class** [*l2tp-class-name*] **Example:** `Router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, if you want 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:** `Router(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 routers. The value entered for the *password* argument must be in the format that matches the input format specified by the [**0** | **7**] keyword option. • **hash** {**md5** | **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**.

	Command or Action	Purpose
Step 5	**digest** check **Example:** `Router(config-l2tp-class)# digest check`	(Optional) Enables the validation of the message digest in received control messages. • 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:** `Router(config-l2tp-class)# hidden`	(Optional) Enables AV pair hiding when sending control messages to an L2TPv3 peer. • AV pair hiding is disabled by default. • In Cisco IOS Release 12.0(29)S and Cisco IOS Release 12.2(27)SBC, 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.

Configuring L2TPv3 Digest Secret Graceful Switchover

L2TPv3 control channel authentication occurs using a password that is configured on all participating peer PE routers. The L2TPv3 Digest Secret Graceful Switchover feature allows a transition from an old control channel authentication password to a new control channel authentication password without disrupting established L2TPv3 tunnels. This feature was introduced in Cisco IOS Release 12.0(30)S.

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.

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.

Prerequisites

Before performing this task, you must enable control channel authentication as documented in the task "Configuring L2TPv3 Control Message Hashing."

Restrictions

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

	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	**l2tp-class** [*l2tp-class-name*] **Example:** `Router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, if you want 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:** `Router(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:** `Router(config-l2tp-class)# end`	Ends your configuration session by exiting to privileged EXEC mode.

	Command or Action	Purpose
Step 6	**show l2tun tunnel all** **Example:** `Router# show l2tun tunnel all`	(Optional) Displays the current state of Layer 2 tunnels and information about configured tunnels, including local and remote Layer 2 Tunneling Protocol (L2TP) hostnames, aggregate packet counts, and control channel information. • 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, that tunnel can be manually cleared and a defect report should be filed with the Cisco Technical Assistance Center (TAC). To manually clear an L2TPv3 tunnel, perform the task "Manually Clearing L2TPv3 Tunnels." **Note** Issue this command to determine if any tunnels are not using the new password for control channel authentication. The output displayed for each tunnel in the specified L2TP class should show that two secrets are configured.
Step 7	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 8	**l2tp-class** [*l2tp-class-name*] **Example:** `Router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, if you want 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:** `Router(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 routers have been updated with the new password.
Step 10	**end** **Example:** `Router(config-l2tp-class)# end`	Ends your configuration session by exiting to privileged EXEC mode.
Step 11	**show l2tun tunnel all** **Example:** `Router# show l2tun tunnel all`	(Optional) Displays the current state of Layer 2 tunnels and information about configured tunnels, including local and remote Layer 2 Tunneling Protocol (L2TP) hostnames, aggregate packet counts, and control channel information. • Tunnels should no longer be using the old control channel authentication password. If a tunnel does not update to show that only one secret is configured after several minutes have passed, that tunnel can be manually cleared and a defect report should be filed with TAC. To manually clear an L2TPv3 tunnel, perform the task "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*

DETAILED STEPS

	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	**l2tp-class** [*l2tp-class-name*] **Example:** `Router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode. • The *l2tp-class-name* argument is optional. However, if you want to configure multiple L2TP classes you must specify a unique *l2tp-class-name* for each one.
Step 4	**hello** *interval* **Example:** `Router(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.

Configuring the 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, fragmentation, payload-specific options, and IP properties. The setting that determines if signaling is used to set up the pseudowire is also included.

For simple L2TPv3 signaling configurations on most platforms, pseudowire class configuration is optional. However, specifying a source IP address to configure a loopback interface is highly recommended. If you do not configure a loopback interface, the router will choose the best available local address, which could be any IP address configured on a core-facing interface. This configuration could prevent a control channel from being established. On the Cisco 12000 series Internet routers, specifying a source IP address is mandatory, and you should configure a loopback interface that is dedicated for the use of L2TPv3 sessions exclusively. If you do not configure other pseudowire class configuration commands, the default values are used.

Once you specify the **encapsulation l2tpv3** command, you cannot remove it using the **no encapsulation l2tpv3** command. Nor can you change the command's setting using the **encapsulation mpls** command. Those 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 with the **no pseudowire-class** command. To change the type of encapsulation, remove the pseudowire with the **no pseudowire-class** command and re-establish the pseudowire and specify the new encapsulation type.

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** | **uti** | *protocol-number*}
12. **sequencing** {**transmit** | **receive** | **both**}

DETAILED STEPS

	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	**pseudowire-class** [*pw-class-name*] **Example:** `Router(config)# pseudowire-class etherpw`	Enters pseudowire class configuration mode and optionally specifies the name of the L2TP pseudowire class.
Step 4	**encapsulation l2tpv3** **Example:** `Router(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:** `Router(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**. (The **protocol none** configuration is necessary when configuring interoperability with a remote peer that runs UTI.)
Step 6	**ip local interface** *interface-name* **Example:** `Router(config-pw)# ip local interface e0/0`	Specifies the PE router 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 pseudowire classes configured between a pair of PE routers. **Note** This command must be configured for pseudowire-class configurations using L2TPv3 as the data encapsulation method.

	Command or Action	Purpose
Step 7	**ip pmtu** **Example:** `Router(config-pw)# ip pmtu`	(Optional) Enables the discovery of the path MTU for tunneled traffic. • 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. • This command must be enabled in the pseudowire class configuration for fragmentation of IP packets before the data enters the pseudowire to occur. **Note** For fragmentation of IP packets before the data enters the pseudowire, Cisco recommends that you also enter the **ip dfbit set** command in the pseudowire class configuration. This allows the PMTU to be obtained more rapidly.
Step 8	**ip tos** {**value** *value* | **reflect**} **Example:** `Router(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:** `Router(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 to be performed on the peer PE router. This command is disabled by default. **Note** On the Cisco 10720 Internet router and Cisco 12000 series Internet routers, the DF bit is set on by default. The **no ip dfbit set** command is not supported.
Step 10	**ip ttl** *value* **Example:** `Router(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.

	Command or Action	Purpose
Step 11	**ip protocol** {**l2tp** | **uti** | *protocol-number*} **Example:** `Router(config-pw)# ip protocol uti`	(Optional) Configures the IP protocol to be used for tunneling packets. • For backward compatibility with UTI, enter **uti** or **120**, the UTI protocol number. The default IP protocol value is **l2tp** or **115**, the L2TP protocol number.
Step 12	**sequencing** {**transmit** | **receive** | **both**} **Example:** `Router(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. • **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.

Configuring the Xconnect Attachment Circuit

This configuration procedure binds an Ethernet, 802.1q VLAN, or Frame Relay attachment circuit to an L2TPv3 pseudowire for xconnect service. The virtual circuit identifier that you configure creates the binding between a pseudowire configured on a PE router and an attachment circuit in a CE device. The virtual circuit identifier configured on the PE router at one end of the L2TPv3 control channel must also be configured on the peer PE router at the other end.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}]

DETAILED STEPS

	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	**interface** *type slot/port* **Example:** `Router(config)# interface ethernet 0/0`	Specifies the interface by type (for example, Ethernet) and slot and port number, and enters interface configuration mode.
Step 4	**xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}] **Example:** `Router(config-if)# xconnect 10.0.3.201 123 pw-class vlan-xconnect`	Specifies the IP address of the peer PE router and the 32-bit virtual circuit identifier shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • 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 router in xconnect configuration mode for 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 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 section "Configuring the Xconnect Attachment Circuit" 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. You may configure both options. **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.

Manually Configuring L2TPv3 Session Parameters

When you bind an attachment circuit to an L2TPv3 pseudowire for xconnect service using the **xconnect l2tpv3 manual** command (see the section "Configuring the Xconnect Attachment Circuit") because you do not want signaling, you must then 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*

DETAILED STEPS

	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* **Example:** `Router(config)# interface ethernet 0/0`	Specifies the interface by type (for example, Ethernet) and 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* **Example:** `Router(config-if)# xconnect 10.0.3.201 123 encapsulation l2tpv3 manual pw-class vlan-xconnect`	Specifies the IP address of the peer PE router and the 32-bit virtual circuit identifier shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • The **encapsulation l2tpv3 manual** parameter specifies that L2TPv3 is to be used as the pseudowire tunneling method, and enters xconnect configuration mode. • 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.

	Command or Action	Purpose
Step 5	**l2tp id** *local-session-id remote-session-id* **Example:** `Router(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 router. • This command is required to complete the attachment circuit configuration and for a static L2TPv3 session configuration.
Step 6	**l2tp cookie local** *size low-value* [*high-value*] **Example:** `Router(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:** `Router(config-if-xconn)# l2tp cookie remote 4 12345`	(Optional) Specifies the value that the router 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:** `Router(config-if-xconn)# l2tp hello l2tp-defaults`	(Optional) Specifies the L2TP class name to use (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 channel is to be used to provide keepalive support through an exchange of L2TP hello messages. By default, no hello messages are sent.

Configuring the Xconnect Attachment Circuit for ATM VP Mode Single Cell Relay over L2TPv3

The ATM VP Mode Single Cell Relay over L2TPv3 feature allows cells coming into a predefined PVP on the ATM interface to be transported over an L2TPv3 pseudowire to a predefined PVP on the egress ATM interface. This task binds a PVP to an L2TPv3 pseudowire for xconnect service.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **atm pvp** *vpi* [**l2transport**]
5. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**atm pvp** *vpi* [**l2transport**] **Example:** `Router(config-if)# atm pvp 5 l2transport`	Specifies that the PVP is dedicated to transporting ATM cells. • The **l2transport** keyword indicates that the PVP is for cell relay. After you enter this command, the router enters l2transport PVP configuration mode. This configuration mode is for Layer 2 transport only; it is not for terminated PVPs.
Step 5	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-if-atm-l2trans-pvp)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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.

Configuring the Xconnect Attachment Circuit for ATM Single Cell Relay VC Mode over L2TPv3

The ATM Single Cell Relay VC Mode over L2TPv3 feature maps one VCC to a single L2TPv3 session. All ATM cells arriving at an ATM interface with the specified VPI and VCI are encapsulated into a single L2TP packet.

The ATM Single Cell Relay VC mode feature can be used to carry any type of AAL traffic over the pseudowire. It will not distinguish OAM cells from User data cells. In this mode, PM and Security OAM cells are also transported over the pseudowire.

Perform this task to enable the ATM Single Cell Relay VC Mode over L2TPv3 feature.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **pvc** [*name*] *vpi/vci* **l2transport**
5. **encapsulation aal0**
6. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**pvc** [*name*] *vpi/vci* **l2transport** **Example:** `Router(config-if)# pvc 5/500 l2transport`	Creates or assigns a name to an ATM PVC, specifies the encapsulation type on an ATM PVC, and enters ATM VC configuration mode. • The **l2transport** keyword indicates that the PVC is for Layer 2 switched connections. After you enter this command, the router enters ATM VC configuration mode.

	Command or Action	Purpose
Step 5	**encapsulation aal0** **Example:** `Router(config-atm-vc)# encapsulation aal0`	Specifies ATM AAL0 encapsulation for the PVC.
Step 6	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-atm-vc)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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.

Configuring the Xconnect Attachment Circuit for ATM Port Mode Cell Relay over L2TPv3

The ATM Port Mode Cell Relay feature packs ATM cells arriving at an ingress ATM interface into L2TPv3 data packets and transports them to the egress ATM interface. A single ATM cell is encapsulated into each L2TPv3 data packet.

Perform this task to enable the ATM Port Mode Cell Relay over L2TPv3 feature.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-if)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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.

Configuring the Xconnect Attachment Circuit for ATM Cell Packing over L2TPv3

The ATM Cell Packing over L2TPv3 feature allows multiple ATM frames to be packed into a single L2TPv3 data packet. ATM cell packing can be configured for Port mode, VP mode, and VC mode. Perform one of the following tasks to configure the ATM Cell Packing over L2TPv3 feature:

- Configuring Port Mode ATM Cell Packing over L2TPv3, page 70
- Configuring VP Mode ATM Cell Packing over L2TPv3, page 71
- Configuring VC Mode ATM Cell Packing over L2TPv3, page 73

Configuring Port Mode ATM Cell Packing over L2TPv3

Perform this task to configure port mode ATM cell packing over L2TPv3.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **atm mcpt-timers** [*timeout-value-1 timeout-value-2 timeout-value-3*]
5. **cell packing** [*cells*] [**mcpt-timer** *timer*]
6. **xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}]

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**atm mcpt-timers** [*timeout-value-1 timeout-value-2 timeout-value-3*] **Example:** `Router(config-if)# atm mcpt-timers 10 100 1000`	(Optional) Sets up the cell-packing timers, which specify how long the PE router can wait for cells to be packed into an L2TPv3 packet.

	Command or Action	Purpose
Step 5	**cell-packing** [*cells*] [**mcpt-timer** *timer*] **Example:** `Router(config-if)# cell-packing 10 mcpt-timer 2`	Enables the packing of multiple ATM cells into each L2TPv3 data packet. • *cells*—(Optional) The number of cells to be packed into an L2TPv3 data packet. The default number of ATM cells to be packed is the maximum transmission unit (MTU) of the interface divided by 52. • **mcpt-timer** *timer*—(Optional) Specifies which maximum cell packing timeout (MCPT) timer to use. The MCPT timers are set using the **mcpt-timers** command. The default value is 1.
Step 6	**xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}] **Example:** `Router(config-if)# xconnect 10.0.3.201 888 encapsulation l2tpv3`	Binds an attachment circuit to a Layer 2 pseudowire and enters xconnect configuration mode.

Configuring VP Mode ATM Cell Packing over L2TPv3

Perform this task to configure VP mode ATM cell packing over L2TPv3.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **atm mcpt-timers** [*timeout-value-1 timeout-value-2 timeout-value-3*]
5. **atm pvp** *vpi* [*peak-rate*] [l2transport]
6. **cell packing** [*cells*] [**mcpt-timer** *timer*]
7. **xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}]

DETAILED STEPS

	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	**interface** *type slot/port* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**atm mcpt-timers** [*timeout-value-1 timeout-value-2 timeout-value-3*] **Example:** `Router(config-if)# atm mcpt-timers 10 100 1000`	(Optional) Sets up the cell-packing timers, which specify how long the PE router can wait for cells to be packed into an L2TPv3 packet.
Step 5	**atm pvp** *vpi* [*peak-rate*] [**l2transport**] **Example:** `Router(config-if)# atm pvp 10 l2transport`	Create a PVP used to multiplex (or bundle) one or more VCs.
Step 6	**cell-packing** [*cells*] [**mcpt-timer** *timer*] **Example:** `Router(config-if)# cell-packing 10 mcpt-timer 2`	Enables the packing of multiple ATM cells into each L2TPv3 data packet. • *cells*—(Optional) The number of cells to be packed into an L2TPv3 data packet. The default number of ATM cells to be packed is the MTU of the interface divided by 52. • **mcpt-timer** *timer*—(Optional) Specifies which MCPT timer to use. The MCPT timers are set using the **mcpt-timers** command. The default value is 1.
Step 7	**xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}] **Example:** `Router(config-if)# xconnect 10.0.3.201 888 encapsulation l2tpv3`	Binds an attachment circuit to a Layer 2 pseudowire and enters xconnect configuration mode.

Configuring VC Mode ATM Cell Packing over L2TPv3

Perform this task to configure VC mode ATM cell packing over L2TPv3.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **atm mcpt-timers** [*timeout-value-1 timeout-value-2 timeout-value-3*]
5. **pvc** [*name*] *vpi/vci* [**ces** | **ilmi** | **qsaal** | **smds** | **l2transport**]
6. **encapsulation aal0**
7. **cell packing** [*cells*] [**mcpt-timer** *timer*]
8. **xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}]

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**atm mcpt-timers** [*timeout-value-1 timeout-value-2 timeout-value-3*] **Example:** `Router(config-if)# atm mcpt-timers 10 100 1000`	(Optional) Sets up the cell-packing timers, which specify how long the PE router can wait for cells to be packed into an L2TPv3 packet.
Step 5	**pvc** [*name*] *vpi/vci* [**ces** | **ilmi** | **qsaal** | **smds** | **l2transport**] **Example:** `Router(config-if)# pvc 1/32 l2transport`	Creates or assigns a name to an ATM PVC, specifies the encapsulation type on an ATM PVC, and enters ATM VC configuration mode.

	Command or Action	Purpose
Step 6	**encapsulation aal0** **Example:** `Router(config-if-atm-vc)# encapsulation aal0`	Specifies ATM AAL0 encapsulation for the PVC.
Step 7	**cell-packing** [*cells*] [**mcpt-timer** *timer*] **Example:** `Router(config-if-atm-vc)# cell-packing 10 mcpt-timer 2`	Enables the packing of multiple ATM cells into each L2TPv3 data packet. • *cells*—(Optional) The number of cells to be packed into an L2TPv3 data packet. The default number of ATM cells to be packed is the MTU of the interface divided by 52. • **mcpt-timer** *timer*—(Optional) Specifies which timer to use. The mcpt timers are set using the **mcpt-timers** command. The default value is 1.
Step 8	**xconnect** *peer-ip-address vcid pseudowire-parameters* [**sequencing** {**transmit** | **receive** | **both**}] **Example:** `Router(config-if-atm-vc)# xconnect 10.0.3.201 888 encapsulation l2tpv3`	Binds an attachment circuit to a Layer 2 pseudowire and enters xconnect configuration mode.

Configuring the Xconnect Attachment Circuit for ATM AAL5 SDU Mode over L2TPv3

The ATM AAL5 SDU Mode feature maps the AAL5 payload of an AAL5 PVC to a single L2TPv3 session. This service will transport OAM and RM cells, but does not attempt to maintain the relative order of these cells with respect to the cells that comprise the AAL5 CPCS-PDU. OAM cells that arrive during the reassembly of a single AAL5 CPCS-PDU are sent immediately over the pseudowire, followed by the AAL5 SDU payload.

Beginning in Cisco IOS Release 12.0(30)S, you may choose to configure the ATM AAL5 SDU Mode feature in ATM VC configuration mode or in VC class configuration mode.

To enable the ATM AAL5 SDU Mode feature, perform one of the following tasks:

- Configuring ATM AAL5 SDU Mode over L2TPv3 in ATM VC Configuration Mode, page 75
- Configuring ATM AAL5 SDU Mode over L2TPv3 in VC Class Configuration Mode, page 76

Configuring ATM AAL5 SDU Mode over L2TPv3 in ATM VC Configuration Mode

Perform this task to bind a PVC to an L2TPv3 pseudowire for ATM AAL5 SDU mode xconnect service.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **pvc** [*name*] *vpi/vci* [**l2transport**]
5. **encapsulation aal5**
6. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**pvc** [*name*] *vpi/vci* [**l2transport**] **Example:** `Router(config-if)# pvc 5/500 l2transport`	Creates or assigns a name to an ATM permanent virtual circuit (PVC), specifies the encapsulation type on an ATM PVC, and enters ATM VC configuration mode. • The **l2transport** keyword indicates that the PVC is for Layer 2 switched connections. After you enter this command, the router enters ATM VC configuration mode.

	Command or Action	Purpose
Step 5	**encapsulation aal5** **Example:** `Router(config-atm-vc)# encapsulation aal5`	Specifies ATM AAL5 encapsulation for the PVC.
Step 6	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-atm-vc)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **pw-class** *pw-class-name*—The pseudowire class configuration from which the data encapsulation type (L2TPv3) is taken. The **pw-class** keyword 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.

Configuring ATM AAL5 SDU Mode over L2TPv3 in VC Class Configuration Mode

You can create a VC class that specifies AAL5 encapsulation and then attach the VC class to an interface, subinterface, or PVC. Perform this task to create a VC class configured for AAL5 encapsulation and attach the VC class to an interface.

Restrictions

This task requires Cisco IOS Release 12.0(30)S or a later release.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vc-class atm** *vc-class-name*
4. **encapsulation aal5**
5. **end**
6. **interface** *type slot/port*
7. **class-int** *vc-class-name*
8. **pvc** [*name*] *vpi/vci* **l2transport**
9. **xconnect** *peer-router-id vcid* **encapsulation l2tpv3**

DETAILED STEPS

	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	**vc-class atm** *name* **Example:** `Router(config)# vc-class atm aal5class`	Creates a VC class and enters VC class configuration mode.
Step 4	**encapsulation aal5** **Example:** `Router(config-vc-class)# encapsulation aal5`	Specifies ATM AAL5 encapsulation for the PVC.
Step 5	**end** **Example:** `Router(config-vc-class)# end`	Ends your configuration session by exiting to privileged EXEC mode.
Step 6	**interface** *type slot*/*port* **Example:** `Router(config)# interface atm 1/0`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 7	**class-int** *vc-class-name* **Example:** `Router(config-if)# class-int aal5class`	Applies a VC class on an the ATM main interface or subinterface. **Note** You can also apply a VC class to a PVC.
Step 8	**pvc** [*name*] *vpi*/*vci* **l2transport** **Example:** `Router(config-if)# pvc 1/200 l2transport`	Creates or assigns a name to an ATM permanent virtual circuit (PVC), specifies the encapsulation type on an ATM PVC, and enters ATM VC configuration mode. • The **l2transport** keyword indicates that the PVC is for Layer 2 switched connections. After you enter this command, the router enters ATM VC configuration mode.
Step 9	**xconnect** *peer-router-id vcid* **encapsulation l2tpv3** **Example:** `Router(config-if-atm-l2trans-pvc)# xconnect 10.13.13.13 100 encapsulation l2tpv3`	Binds the attachment circuit to a pseudowire VC.

Configuring OAM Local Emulation for ATM AAL5 over L2TPv3

If a PE router does not support the transport of OAM cells across an L2TPv3 session, you can use OAM cell emulation to locally terminate or loopback the OAM cells. You configure OAM cell emulation on both PE routers. You use the **oam-ac emulation-enable** command on both PE routers to enable OAM cell emulation.

After you enable OAM cell emulation on a router, you can configure and manage the ATM VC in the same manner as you would a terminated VC. A VC that has been configured with OAM cell emulation can send loopback cells at configured intervals toward the local CE router. The endpoint can be either of the following:

- End-to-end loopback, which sends OAM cells to the local CE router.
- Segment loopback, which responds to OAM cells to a device along the path between the PE and CE routers.

The OAM cells have the following information cells:

- Alarm indication signal (AIS)
- Remote defect indication (RDI)

These cells identify and report defects along a VC. When a physical link or interface failure occurs, intermediate nodes insert OAM AIS cells into all the downstream devices affected by the failure. When a router receives an AIS cell, it marks the ATM VC as down and sends an RDI cell to let the remote end know about the failure.

Beginning in Cisco IOS Release 12.0(30)S, you may choose to configure the OAM Local Emulation for ATM AAL5 over L2TPv3 feature in ATM VC configuration mode or in VC class configuration mode.

To enable the OAM Local Emulation for ATM AAL5 over L2TPv3 feature, perform one of the following tasks:

- Configuring OAM Local Emulation for ATM AAL5 over L2TPv3 in ATM VC Configuration Mode, page 78
- Configuring OAM Local Emulation for ATM AAL5 over L2TPv3 in VC Class Configuration Mode, page 80

Configuring OAM Local Emulation for ATM AAL5 over L2TPv3 in ATM VC Configuration Mode

Perform this task to enable the OAM Local Emulation for ATM AAL5 over L2TPv3 feature in ATM VC configuration mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **pvc** [*name*] *vpi/vci* [**l2transport**]
5. **encapsulation aal5**
6. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*
7. **oam-ac emulation-enable** [*ais-rate*]
8. **oam-pvc manage** [*frequency*]

DETAILED STEPS

	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* **Example:** `Router(config)# interface ATM 4/1`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**pvc** [*name*] *vpi/vci* [**l2transport**] **Example:** `Router(config-if)# pvc 5/500 l2transport`	Creates or assigns a name to an ATM PVC, specifies the encapsulation type on an ATM PVC, and enters ATM VC configuration mode. • The **l2transport** keyword indicates that the PVC is for Layer 2 switched connections. After you enter this command, the router enters ATM VC configuration mode.
Step 5	**encapsulation aal5** **Example:** `Router(config-atm-vc)# encapsulation aal5`	Specifies ATM AAL5 encapsulation for the PVC.
Step 6	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-atm-vc)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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.

	Command or Action	Purpose
Step 7	**oam-ac emulation-enable** [*ais-rate*] **Example:** `Router(config-atm-vc)# oam-ac emulation-enable 30`	Enables OAM cell emulation on AAL5 over L2TPv3. • The **oam-ac emulation-enable** command lets you specify the rate at which AIS cells are sent. The default is one cell every second. The range is 0 to 60 seconds.
Step 8	**oam-pvc manage** [*frequency*] **Example:** `Router(config-atm-vc)# oam-pvc manage`	(Optional) Enables the PVC to generate end-to-end OAM loopback cells that verify connectivity on the virtual circuit. • The optional *frequency* argument is the interval between transmission of loopback cells and ranges from 0 to 600 seconds. The default value is 10 seconds. **Note** You can configure the **oam-pvc manage command only after you issue the oam-ac emulation-enable command.**

Configuring OAM Local Emulation for ATM AAL5 over L2TPv3 in VC Class Configuration Mode

This task configures OAM Cell Emulation as part of a VC class. After a VC class is configured, you can apply the VC class to an interface, a subinterface, or a VC.

When you apply a VC class to an interface, the settings in the VC class apply to all the VCs on that interface unless you specify otherwise at a lower level, such as the subinterface or VC level. For example, if you create a VC class that specifies OAM cell emulation and sets the AIS cell rate to 30 seconds and apply that VC class to an interface, every VC on that interface will use the AIS cell rate of 30 seconds. If you then enable OAM cell emulation on a single PVC and set the AIS cell rate to 15 seconds, the 15 second AIS cell rate configured at the PVC level will take precedence over the 30 second AIS cell rate configured at the interface level.

Perform this task to create a VC class configured for OAM emulation and to attach the VC class to an interface.

Restrictions

This task requires Cisco IOS Release 12.0(30)S or a later release.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vc-class atm** *name*
4. **encapsulation** ***layer-type***
5. **oam-ac emulation-enable** [*ais-rate*]
6. **oam-pvc manage** [*frequency*]
7. **end**
8. **interface** *type slot/port*
9. **class-int** *vc-class-name*
10. **pvc** [*name*] *vpi/vci* **l2transport**
11. **xconnect** *peer-router-id vcid* **encapsulation l2tpv3**

DETAILED STEPS

	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	**vc-class atm** *name* **Example:** `Router(config)# vc-class atm oamclass`	Creates a VC class and enters vc-class configuration mode.
Step 4	**encapsulation** *layer-type* **Example:** `Router(config-vc-class)# encapsulation aal5`	Configures the ATM adaptation layer (AAL) and encapsulation type.
Step 5	**oam-ac emulation-enable** [*ais-rate*] **Example:** `Router(config-vc-class)# oam-ac emulation-enable 30`	Enables OAM cell emulation for AAL5 over L2TPv3. • The *ais-rate* variable lets you specify the rate at which AIS cells are sent. The default is one cell every second. The range is 0 to 60 seconds.
Step 6	**oam-pvc manage** [*frequency*] **Example:** `Router(config-vc-class)# oam-pvc manage`	(Optional) Enables the PVC to generate end-to-end OAM loopback cells that verify connectivity on the virtual circuit. • The optional *frequency* argument is the interval between transmission of loopback cells and ranges from 0 to 600 seconds. The default value is 10 seconds. **Note** You can configure the **oam-pvc manage** command only after you issue the **oam-ac emulation-enable** command.
Step 7	**end** **Example:** `Router(config-vc-class)# end`	Ends your configuration session by exiting to privileged EXEC mode.
Step 8	**interface** *type slot*/*port* **Example:** `Router(config)# interface atm1/0`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.

Step 9	**class-int** *vc-class-name* **Example:** `Router(config-if)# class-int oamclass`	Applies a VC class on an the ATM main interface or subinterface. **Note** You can also apply a VC class to a PVC.
Step 10	**pvc** [*name*] *vpi*/*vci* **l2transport** **Example:** `Router(config-if)# pvc 1/200 l2transport`	Creates or assigns a name to an ATM PVC, specifies the encapsulation type on an ATM PVC, and enters ATM VC configuration mode. • The **l2transport** keyword indicates that the PVC is for Layer 2 switched connections. After you enter this command, the router enters ATM VC configuration mode.
Step 11	**xconnect** *peer-router-id vcid* **encapsulation l2tpv3** **Example:** `Router(config-if-atm-l2trans-pvc)# xconnect 10.13.13.13 100 encapsulation l2tpv3`	Binds the attachment circuit to a pseudowire VC.

Configuring Protocol Demultiplexing for L2TPv3

The 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. IPv6 traffic is transparently tunneled to the IPv6 network using L2TPv3 pseudowires without affecting the configuration of the CE routers. IPv4 traffic is routed as usual within the IPv4 network, maintaining the existing performance and reliability of the IPv4 network.

The IPv4 PE routers must be configured to demultiplex incoming IPv6 traffic from IPv4 traffic. The PE routers facing the IPv6 network do not require demultiplexing configuration. The configuration of the IPv6 network is beyond the scope of this document. For more information on configuring an IPv6 network, refer to the *Cisco IOS IPv6 Configuration Guide*.

Perform one of the following tasks on the customer-facing IPv4 PE routers to enable IPv6 protocol demultiplexing:

- Configuring Protocol Demultiplexing for Ethernet Interfaces, page 83
- Configuring Protocol Demultiplexing for Frame Relay Interfaces, page 84
- Configuring Protocol Demultiplexing for PPP Interfaces, page 85
- Configuring Protocol Demultiplexing for HDLC Interfaces, page 87

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**

DETAILED STEPS

	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* **Example:** `Router(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:** `Router(config-if)# ip address 172.16.128.4`	Sets a primary or secondary IP address for an interface.

	Command or Action	Purpose
Step 5	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-if)# xconnect 10.0.3.201 888 pw-class demux`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel and enters xconnect configuration mode. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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:** `Router(config-if-xconn)# match protocol ipv6`	Enables protocol demultiplexing of IPv6 traffic.

Configuring Protocol Demultiplexing for Frame Relay Interfaces

Perform this task to configure the Protocol Demultiplexing feature on a Frame Relay interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port-adapter.subinterface-number* [**multipoint** | **point-to-point**]
4. **ip address** *ip-address mask* [**secondary**]
5. **frame-relay interface-dlci** *dlci* [**ietf** | **cisco**] [**voice-cir** *cir*] [**ppp** *virtual-template-name*]
6. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*
7. **match protocol ipv6**

DETAILED STEPS

	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	**interface** *type slot/port-adapter.subinterface-number* [**multipoint** | **point-to-point**] **Example:** `Router(config)# interface serial 1/1.2 multipoint`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 172.16.128.4`	Sets a primary or secondary IP address for an interface.
Step 5	**frame-relay interface-dlci** *dlci* [**ietf** | **cisco**] [**voice-cir** *cir*] [**ppp** *virtual-template-name*] **Example:** `Router(config-if)# frame-relay interface-dlci 100`	Assigns a DLCI to a specified Frame Relay subinterface on the router or access server, assigns a specific PVC to a DLCI, or applies a virtual template configuration for a PPP session and enters Frame Relay DLCI interface configuration mode.
Step 6	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-fr-dlci)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel and enters xconnect configuration mode. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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 7	**match protocol ipv6** **Example:** `Router(config-if-xconn)# match protocol ipv6`	Enables protocol demultiplexing of IPv6 traffic.

Configuring Protocol Demultiplexing for PPP Interfaces

Perform this task to configure the Protocol Demultiplexing feature on a Point-to-Point Protocol (PPP) interface.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **interface** *type slot/port*
4. **ip address** *ip-address mask* [**secondary**]
5. **encapsulation** *physical-interface*
6. **ppp** *interface-address*
7. **xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name*
8. **match protocol ipv6**

DETAILED STEPS

	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* **Example:** `Router(config)# interface serial 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:** `Router(config-if)# ip address 192.167.1.1 255.255.255.252`	Sets a primary or secondary IP address for an interface.
Step 5	**encapsulation** *physical-interface* **Example:** `Router(config-if)# encapsulation ppp`	Specifies PPP encapsulation for IPv6.
Step 6	**ppp** *interface-address* **Example:** `Router(config-if)# ppp ipv6cp id proxy A8BB:CCFF:FE00:7000`	

	Command or Action	Purpose
Step 7	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-if)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel and enters xconnect configuration mode. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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 8	**match protocol ipv6** **Example:** `Router(config-if-xconn)# match protocol ipv6`	Enables protocol demultiplexing of IPv6 traffic.

Configuring Protocol Demultiplexing for HDLC Interfaces

Perform this task to configure the Protocol Demultiplexing feature on a High-Level Data Link Control (HDLC) 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**

DETAILED STEPS

	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	**interface** *type slot/port* **Example:** `Router(config)# interface serial 0/0`	Specifies the interface by type, slot, and port number, and enters interface configuration mode.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 172.16.128.4 255.255.255.252`	Sets a primary or secondary IP address for an interface.
Step 5	**xconnect** *peer-ip-address vcid* **pw-class** *pw-class-name* **Example:** `Router(config-if)# xconnect 10.0.3.201 888 pw-class atm-xconnect`	Specifies the IP address of the peer PE router and the 32-bit VCI shared between the PE at each end of the control channel and enters xconnect configuration mode. • The peer router ID (IP address) and virtual circuit ID must be a unique combination on the router. • **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:** `Router(config-if-xconn)# match protocol ipv6`	Enables protocol demultiplexing of IPv6 traffic.

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* | **tunid** *tunnel-id* | **local ip** *ip-address* | **remote ip** *ip-address* | **all**}

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear l2tun** {**l2tp-class** *l2tp-class-name* | **tunid** *tunnel-id* | **local ip** *ip-address* | **remote ip** *ip-address* | **all**} **Example:** `Router# clear l2tun tunid 56789`	(Optional) Clears the specified L2TPv3 tunnel. • **l2tp-class** *l2tp-class-name*—All L2TPv3 tunnels with the specified L2TP class name are torn down. • **tunid** *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 Tunnel Protocol Version 3

This section provides the following configuration examples:

- Configuring a Static L2TPv3 Session for an Xconnect Ethernet Interface: Example, page 90
- Configuring a Negotiated L2TPv3 Session for an Xconnect VLAN Subinterface: Example, page 91
- Configuring a Negotiated L2TPv3 Session for Local HDLC Switching: Example, page 91
- Verifying an L2TPv3 Session: Example, page 91
- Verifying an L2TP Control Channel: Example, page 92
- Configuring L2TPv3 Control Channel Authentication: Examples, page 93
- Configuring L2TPv3 Digest Secret Graceful Switchover: Example, page 93
- Verifying L2TPv3 Digest Secret Graceful Switchover: Example, page 93
- Configuring Frame Relay DLCI-to-DLCI Switching: Example, page 100
- Configuring ATM VP Mode Single Cell Relay over L2TPv3: Example, page 94
- Verifying ATM VP Mode Single Cell Relay over L2TPv3 Configuration: Example, page 94
- Configuring ATM Single Cell Relay VC Mode over L2TPv3: Example, page 95
- Verifying ATM Single Cell Relay VC Mode over L2TPv3: Example, page 95
- Configuring ATM Port Mode Cell Relay over L2TPv3: Example, page 96
- Configuring ATM Cell Packing over L2TPv3: Examples, page 96
- Configuring ATM AAL5 SDU Mode over L2TPv3: Examples, page 96
- Verifying ATM AAL5 SDU Mode over L2TPv3 Configuration: Examples, page 97
- Configuring OAM Local Emulation for ATM AAL5 over L2TPv3: Examples, page 97
- Verifying OAM Local Emulation for ATM AAL5 over L2TPv3 Configuration: Examples, page 99

- Configuring Protocol Demultiplexing for L2TPv3: Examples, page 99
- Manually Clearing an L2TPv3 Tunnel: Example, page 100
- Configuring Frame Relay DLCI-to-DLCI Switching: Example, page 100
- Configuring Frame Relay Trunking: Example, page 100
- Configuring QoS for L2TPv3 on the Cisco 7500 Series: Example, page 101
- Configuring QoS for L2TPv3 on the Cisco 12000 Series: Examples, page 101
- Configuring a QoS Policy for Committed Information Rate Guarantees: Example, page 107
- Setting the Frame Relay DE Bit Configuration: Example, page 107
- Matching the Frame Relay DE Bit Configuration: Example, page 107
- Configuring MLFR for L2TPv3 on the Cisco 12000 Series: Example, page 108
- Configuring an MQC for Committed Information Rate Guarantees: Example, page 109

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

Configuring a Static L2TPv3 Session for an Xconnect Ethernet Interface: Example

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 router 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
```

Configuring a Negotiated L2TPv3 Session for an Xconnect VLAN Subinterface: Example

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
```

Configuring a Negotiated L2TPv3 Session for Local HDLC Switching: Example

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
```

Verifying an L2TPv3 Session: Example

To display detailed information about current L2TPv3 sessions on a router, use the **show l2tun session all** command:

```
Router# show l2tunnel session all
```

```
Session Information Total tunnels 0 sessions 1

Session id 111 is up, tunnel id 0
Call serial number is 0
Remote tunnel name is
  Internet address is 10.0.0.1
  Session is manually signalled
  Session state is established, time since change 00:06:05
    0 Packets sent, 0 received
    0 Bytes sent, 0 received
    Receive packets dropped:
      out-of-order:             0
      total:                    0
    Send packets dropped:
      exceeded session MTU:     0
      total:                    0
  Session vcid is 123
  Session Layer 2 circuit, type is ATM VPC CELL, name is ATM3/0/0:1000007
  Circuit state is UP
    Remote session id is 222, remote tunnel id 0
  DF bit off, ToS reflect disabled, ToS value 0, TTL value 255
  Session cookie information:
    local cookie, size 8 bytes, value 00 00 00 00 00 00 00 64
    remote cookie, size 8 bytes, value 00 00 00 00 00 00 00 C8
  SSS switching enabled
  Sequencing is off
```

Verifying an L2TP Control Channel: Example

To display detailed information the L2TP control channels that are set up to other L2TP-enabled devices for all L2TP sessions on the router, use the **show l2tun tunnel all** command. The L2TP control channel is used to negotiate capabilities, monitor the health of the peer PE router, and set up various components of an L2TPv3 session.

```
Router# show l2tun tunnel all

Tunnel id 26515 is up, remote id is 41814, 1 active sessions
  Tunnel state is established, time since change 03:11:50
  Tunnel transport is IP (115)
  Remote tunnel name is tun1
    Internet Address 172.18.184.142, port 0
  Local tunnel name is Router
    Internet Address 172.18.184.116, port 0
  Tunnel domain is
  VPDN group for tunnel is
  0 packets sent, 0 received
  0 bytes sent, 0 received
  Control Ns 11507, Nr 11506
  Local RWS 2048 (default), Remote RWS 800
  Tunnel PMTU checking disabled
  Retransmission time 1, max 1 seconds
  Unsent queuesize 0, max 0
  Resend queuesize 1, max 1
  Total resends 0, ZLB ACKs sent 11505
  Current nosession queue check 0 of 5
  Retransmit time distribution: 0 0 0 0 0 0 0 0 0 0
  Sessions disconnected due to lack of resources 0
```

Configuring L2TPv3 Control Channel Authentication: Examples

The following example configures CHAP-style authentication of the L2TPv3 control channel:

```
l2tp-class class0
 authentication
 password cisco
```

The following example configures control channel authentication using the L2TPv3 Control Message Hashing feature:

```
l2tp-class class1
 digest secret cisco hash sha
 hidden
```

The following example configures control channel integrity checking and disables validation of the message digest using the L2TPv3 Control Message Hashing feature:

```
l2tp-class class2
 digest hash sha
 no digest check
```

The following example disables validation of the message digest using the L2TPv3 Control Message Hashing feature:

```
l2tp-class class3
 no digest check
```

Configuring L2TPv3 Digest Secret Graceful Switchover: Example

The following example uses 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.

```
Router(config)# l2tp-class class1
Router(config-l2tp-class)# digest secret cisco2 hash sha
!
! Verify that all peer PE routers have been updated to use the new password before
! removing the old password.
!
Router(config-l2tp-class)# no digest secret cisco hash sha
```

Verifying L2TPv3 Digest Secret Graceful Switchover: Example

The following **show l2tun tunnel all** command output shows information about the L2TPv3 Digest Secret Graceful Switchover feature:

```
Router# 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
```

Configuring a Pseudowire Class for Fragmentation of IP Packets: Example

The following is a sample configuration of a pseudowire class that will allow IP traffic generated from the CE router to be fragmented before entering the pseudowire:

```
pseudowire class class1
 encapsulation l2tpv3
 ip local interface Loopback0
 ip pmtu
 ip dfbit set
```

Configuring ATM VP Mode Single Cell Relay over L2TPv3: Example

The following configuration binds a PVP to an xconnect attachment circuit to forward ATM cells over an established L2TPv3 pseudowire:

```
pw-class atm-xconnect
 encapsulation l2tpv3

interface ATM 4/1
 atm pvp 5 l2transport
 xconnect 10.0.3.201 888 pw-class atm-xconnect
```

Verifying ATM VP Mode Single Cell Relay over L2TPv3 Configuration: Example

To verify the configuration of a PVP, use the **show atm vp** command in privileged EXEC mode:

```
Router# show atm vp 5

ATM4/1/0  VPI: 5, Cell-Relay, PeakRate: 155000, CesRate: 0, DataVCs: 0,
CesVCs: 0, Status: ACTIVE

  VCD    VCI Type     InPkts  OutPkts  AAL/Encap     Status
    8      3 PVC           0        0  F4 OAM        ACTIVE
    9      4 PVC           0        0  F4 OAM        ACTIVE

TotalInPkts: 0, TotalOutPkts: 0, TotalInFast: 0, TotalOutFast: 0,
TotalBroadcasts: 0
```

Configuring ATM Single Cell Relay VC Mode over L2TPv3: Example

The following example shows how to configure the ATM Single Cell Relay VC Mode over L2TPv3 feature:

```
pw-class atm-xconnect
 encapsulation l2tpv3

interface ATM 4/1
 pvc 5/500 l2transport
  encapsulation aal0
  xconnect 10.0.3.201 888 pw-class atm-xconnect
```

Verifying ATM Single Cell Relay VC Mode over L2TPv3: Example

The following **show atm vc** command output displays information about VCC cell relay configuration:

```
Router# show atm vc

VCD/                                             Peak   Avg/Min    Burst
Interface   Name  VPI   VCI   Type     Encaps     Kbps    Kbps      Cells         Sts
2/0         4     9     901    PVC      AAL0     149760   N/A                     UP
```

The following **show l2tun session** command output displays information about VCC cell relay configuration:

```
Router# show l2tun session all

Session Information Total tunnels 1 sessions 2
Session id 41883 is up, tunnel id 18252
Call serial number is 3211600003
Remote tunnel name is khur-l2tp
  Internet address is 10.0.0.2
  Session is L2TP signalled
  Session state is established, time since change 00:00:38
    8 Packets sent, 8 received
    416 Bytes sent, 416 received
    Receive packets dropped:
      out-of-order:             0
      total:                    0
    Send packets dropped:
      exceeded session MTU:     0
      total:                    0
  Session vcid is 124
  Session Layer 2 circuit, type is ATM VCC CELL, name is ATM2/0:9/901
  Circuit state is UP
    Remote session id is 38005, remote tunnel id 52436
  DF bit off, ToS reflect disabled, ToS value 0, TTL value 255
  No session cookie information available
  FS cached header information:
    encap size = 24 bytes
    00000000 00000000 00000000 00000000
    00000000 00000000
  Sequencing is off
```

Configuring ATM Port Mode Cell Relay over L2TPv3: Example

The following example shows how to configure the ATM Port Mode Cell Relay over L2TPv3 feature:

```
pw-class atm-xconnect
 encapsulation l2tpv3

interface atm 4/1
 xconnect 10.0.3.201 888 pw-class atm-xconnect
```

Configuring ATM Cell Packing over L2TPv3: Examples

The following examples show how to configure the ATM Cell Packing over L2TPv3 feature for Port mode, VP mode, and VC mode:

Port Mode

```
interface atm 4/1
 atm mcpt-timers 10 100 1000
 cell-packing 10 mcpt-timer 2
 xconnect 10.0.3.201 888 encapsulation l2tpv3
```

VP Mode

```
interface atm 4/1
 atm mcpt-timers 10 100 1000
 atm pvp 10 l2transport
 cell-packing 10 mcpt-timer 2
 xconnect 10.0.3.201 888 encapsulation l2tpv3
```

VC Mode

```
interface atm 4/1
 atm mcpt-timers 10 100 1000
 pvc 1/32 l2transport
  encapsulation aal0
  cell-packing 10 mcpt-timer 2
  xconnect 10.0.3.201 888 encapsulation l2tpv3
```

Configuring ATM AAL5 SDU Mode over L2TPv3: Examples

Configuring ATM AAL5 SDU Mode over L2TPv3 in ATM VC Configuration Mode

The following configuration binds a PVC to an xconnect attachment circuit to forward ATM cells over an established L2TPv3 pseudowire:

```
pw-class atm-xconnect
 encapsulation l2tpv3

interface atm 4/1
 pvc 5/500 l2transport
  encapsulation aal5
  xconnect 10.0.3.201 888 pw-class atm-xconnect
```

Configuring ATM AAL5 SDU Mode over L2TPv3 in VC-Class Configuration Mode

The following example configures ATM AAL5 over L2TPv3 in VC class configuration mode. The VC class is then applied to an interface.

```
vc-class atm aal5class
```

```
 encapsulation aal5
!
interface atm 1/0
 class-int aal5class
 pvc 1/200 l2transport
  xconnect 10.13.13.13 100 encapsulation l2tpv3
```

Verifying ATM AAL5 SDU Mode over L2TPv3 Configuration: Examples

Verifying ATM AAL5 over MPLS in ATM VC Configuration Mode

To verify the configuration of a PVC, use the **show atm vc** command in privileged EXEC mode:

```
Router# show atm vc

VCD/                                             Peak   Avg/Min    Burst
Interface   Name  VPI   VCI   Type     Encaps     Kbps    Kbps      Cells         Sts
2/0         pvc    9    900    PVC      AAL5      2400    200                     UP
2/0         4      9    901    PVC      AAL5     149760   N/A                     UP
```

The following **show l2tun session** command output displays information about ATM VC mode configurations:

```
Router# show l2tun session brief

Session Information Total tunnels 1 sessions 2
LocID      TunID      Peer-address    State        Username, Intf/
                                     sess/cir        Vcid, Circuit
41875      18252      10.0.0.2         est,UP       124, AT2/0:9/901
111           0       10.0.0.2         est,UP       123, AT2/0:9/900
```

Verifying ATM AAL5 over MPLS in VC Class Configuration Mode

To verify that ATM AAL5 over L2TPv3 is configured as part of a VC class, issue the **show atm class-links** command. The command output shows the type of encapsulation and that the VC class was applied to an interface.

```
Router# show atm class links 1/100

Displaying vc-class inheritance for ATM1/0.0, vc 1/100:
no broadcast - Not configured - using default
encapsulation aal5 - VC-class configured on main interface
.
.
.
```

Configuring OAM Local Emulation for ATM AAL5 over L2TPv3: Examples

Configuring OAM Cell Emulation for ATM AAL5 over L2TPv3 in ATM VC Configuration Mode

The following configuration binds a PVC to an xconnect attachment circuit to forward ATM AAL5 frames over an established L2TPv3 pseudowire, enables OAM local emulation, and specifies that AIS cells are sent every 30 seconds:

```
pw-class atm-xconnect
 encapsulation l2tpv3

interface ATM 4/1
 pvc 5/500 l2transport
```

```
  encapsulation aal5
  xconnect 10.0.3.201 888 pw-class atm-xconnect
  oam-ac emulation-enable 30
```

Configuring OAM Cell Emulation for ATM AAL5 over L2TPv3 in VC Class Configuration Mode

The following example configures OAM cell emulation for ATM AAL5 over L2TPv3 in VC class configuration mode. The VC class is then applied to an interface.

```
vc-class atm oamclass
 encapsulation aal5
 oam-ac emulation-enable 30
 oam-pvc manage
!
interface atm1/0
 class-int oamclass
 pvc 1/200 l2transport
  xconnect 10.13.13.13 100 encapsulation l2tpv3
```

The following example configures OAM cell emulation for ATM AAL5 over L2TPv3 in VC class configuration mode. The VC class is then applied to a PVC.

```
vc-class atm oamclass
 encapsulation aal5
 oam-ac emulation-enable 30
 oam-pvc manage
!
interface atm1/0
 pvc 1/200 l2transport
  class-vc oamclass
  xconnect 10.13.13.13 100 encapsulation l2tpv3
```

The following example configures OAM cell emulation for ATM AAL5 over L2TPv3 in VC class configuration mode. The OAM cell emulation AIS rate is set to 30 for the VC class. The VC class is then applied to an interface. One PVC is configured with OAM cell emulation at an AIS rate of 10. That PVC uses the AIS rate of 10 instead of 30.

```
vc-class atm oamclass
 encapsulation aal5
 oam-ac emulation-enable 30
 oam-pvc manage
!
interface atm1/0
 class-int oamclass
 pvc 1/200 l2transport
  oam-ac emulation-enable 10
  xconnect 10.13.13.13 100 encapsulation l2tpv3
```

Verifying OAM Local Emulation for ATM AAL5 over L2TPv3 Configuration: Examples

The following **show atm pvc** command output shows that OAM cell emulation is enabled and working on the ATM PVC:

```
Router# show atm pvc 5/500

ATM4/1/0.200: VCD: 6, VPI: 5, VCI: 500
UBR, PeakRate: 1
AAL5-LLC/SNAP, etype:0x0, Flags: 0x34000C20, VCmode: 0x0
OAM Cell Emulation: enabled, F5 End2end AIS Xmit frequency: 1 second(s)
OAM frequency: 0 second(s), OAM retry frequency: 1 second(s)
OAM up retry count: 3, OAM down retry count: 5
OAM Loopback status: OAM Disabled
OAM VC state: Not ManagedVerified
ILMI VC state: Not Managed
InPkts: 564, OutPkts: 560, InBytes: 19792, OutBytes: 19680
InPRoc: 0, OutPRoc: 0
InFast: 4, OutFast: 0, InAS: 560, OutAS: 560
InPktDrops: 0, OutPktDrops: 0
CrcErrors: 0, SarTimeOuts: 0, OverSizedSDUs: 0
Out CLP=1 Pkts: 0
OAM cells received: 26
F5 InEndloop: 0, F5 InSegloop: 0, F5 InAIS: 0, F5 InRDI: 26
OAM cells sent: 77
F5 OutEndloop: 0, F5 OutSegloop: 0, F5 OutAIS: 77, F5 OutRDI: 0
OAM cell drops: 0
Status: UP
```

Configuring Protocol Demultiplexing for L2TPv3: Examples

The following examples show how to configure the Protocol Demultiplexing feature on the IPv4 PE routers. The PE routers facing the IPv6 network do not require demultiplexing configuration.

Ethernet Interface

```
interface ethernet 0/1
 ip address 172.16.128.4
 xconnect 10.0.3.201 888 pw-class demux
  match protocol ipv6
```

Frame Relay Interface

```
interface serial 1/1.1 multipoint
 ip address 172.16.128.4
 frame-relay interface-dlci 100
  xconnect 10.0.3.201 888 pw-class atm-xconnect
   match protocol ipv6
```

PPP Interface

```
interface serial 0/0
 ip address 192.167.1.1 2555.2555.2555.252
 encapsulation ppp
 ppp ipv6cp id proxy A8BB:CCFF:FE00:7000
 xconnect 75.0.0.1 1 pw-class 12tp
  match protocol ipv6
```

HDLC Interface

```
interface serial 0/0
 ip address 192.168.1.2 2555.2555.2555.252
 xconnect 75.0.0.1 1 pw-class l2tp
  match protocol ipv6
```

Manually Clearing an L2TPv3 Tunnel: Example

The following example demonstrates how to manually clear a specific L2TPv3 tunnel using the tunnel ID:

clear l2tun tunid 65432

Configuring Frame Relay DLCI-to-DLCI Switching: Example

The following is a sample configuration for switching a Frame Relay DLCI over a pseudowire:

```
pseudowire-class fr-xconnect
 encapsulation l2tpv3
 protocol l2tpv3
 ip local interface Loopback0
 sequencing both
!
interface Serial0/0
 encapsulation frame-relay
 frame-relay intf-type dce
!
connect one Serial0/0 100 l2transport
 xconnect 10.0.3.201 555 pw-class fr-xconnect
!
connect two Serial0/0 200 l2transport
 xconnect 10.0.3.201 666 pw-class fr-xconnect
```

Configuring Frame Relay Trunking: Example

The following is a sample configuration for setting up a trunk connection for an entire serial interface over a pseudowire. All incoming packets are switched to the pseudowire regardless of content.

Note that when you configure trunking for a serial interface, the trunk connection does not require an encapsulation method. You do not, therefore, need to enter the **encapsulation frame-relay** command. Reconfiguring the default encapsulation removes all xconnect configuration settings from the interface.

```
interface Serial0/0
 xconnect 10.0.3.201 555 pw-class serial-xconnect
```

Configuring QoS for L2TPv3 on the Cisco 7500 Series: Example

The following example shows the MQC commands used on a Cisco 7500 series router to configure a CIR guarantee of 256 kbps on DLCI 100 and 512 kbps for DLCI 200 on the egress side of a Frame Relay interface that is also configured for L2TPv3 tunneling:

```
ip cef distributed
 class-map dlci100
 match fr-dlci 100
 class-map dlci200
 match fr-dlci 200
!
policy-map dlci
 class dlci100
 bandwidth 256
 class dlci200
 bandwidth 512
!
interface Serial0/0
 encapsulation frame-relay
 frame-relay interface-type dce
 service-policy output dlci
!
connect one Serial0/0 100 l2transport
 xconnect 10.0.3.201 555 encapsulation l2tpv3 pw-class mqc
!
connect two Serial0/0 200 l2transport
 xconnect 10.0.3.201 666 encapsulation l2tpv3 pw-class mqc
```

Configuring QoS for L2TPv3 on the Cisco 12000 Series: Examples

This section contains the following examples for configuring QoS for L2TPv3 on the Cisco 12000 series:

- Configuring QoS on a Frame Relay Interface in a TSC-Based L2TPv3 Tunnel Session, page 101
- Configuring Traffic Policing on an ISE/E5 Interface in a Native L2TPv3 Tunnel Session, page 103
- Configuring Tunnel Marking in a Native L2TPv3 Tunnel Session, page 105
- Configuring Traffic Shaping in a Native L2TPv3 Tunnel Session, page 105

Configuring QoS on a Frame Relay Interface in a TSC-Based L2TPv3 Tunnel Session

To apply a QoS policy for L2TPv3 to a Frame Relay interface on a Cisco 12000 series 2-port Channelized OC-3/STM-1 (DS1/E1) or 6-port Channelized T3 line card in a tunnel server card-based L2TPv3 tunnel session, you must:

- Use the **map-class frame-relay** *class-name* command in global configuration mode to apply a QoS policy to a Frame Relay class of traffic.
- Use the **frame-relay interface-dcli** *dcli-number* **switched** command (in interface configuration mode) to enter Frame Relay DLCI interface configuration mode and then the **class** command to configure a QoS policy for a Frame Relay class of traffic on the specified DLCI. You must enter a separate series of these configuration commands to configure QoS for each Frame Relay DLCI on the interface.

As shown in the following example, when you configure QoS for L2TPv3 on the ingress side of a Cisco 12000 series Frame Relay interface, you may also configure the value of the ToS byte used in IP headers of tunneled packets when you configure the L2TPv3 pseudowire (see the section "Configuring the L2TPv3 Pseudowire").

The following example shows the MQC commands and ToS byte configuration used on a Cisco 12000 series router to apply a QoS policy for DLCI 100 on the ingress side of a Frame Relay interface configured for server card-based L2TPv3 tunneling:

```
policy-map frtp-policy
 class class-default
 police cir 8000 bc 6000 pir 32000 be 4000 conform-action transmit exceed-action
set-frde-transmit violate-action drop
!
map-class frame-relay fr-map
 service-policy input frtp-policy
!
interface Serial0/1/1:0
 encapsulation frame-relay
 frame-relay interface-dlci 100 switched
   class fr-map
 connect frol2tp1 Serial0/1/1:0 100 l2transport
   xconnect 10.0.3.201 666 encapsulation l2tpv3 pw-class aaa
!
pseudowire-class aaa
encapsulation l2tpv3
 ip tos value 96
```

To apply a QoS policy for L2TPv3 to the egress side of a Frame Relay interface on a Cisco 12000 series 2-port Channelized OC-3/STM-1 (DS1/E1) or 6-port Channelized T3 line card, you must:

- Use the **match ip precedence** command in class-map configuration mode to configure the IP precedence value used to determine the egress queue for each L2TPv3 packet with a Frame Relay payload.
- Use the **random-detect** command in policy-map class configuration mode to enable a WRED drop policy for a Frame Relay traffic class that has a bandwidth guarantee. Use the **random-detect precedence** command to configure the WRED and MDRR parameters for particular IP precedence values.

The next example shows the MQC commands used on a Cisco 12000 series Internet router to apply a QoS policy with WRED/MDRR settings for specified IP precedence values to DLCI 100 on the egress side of a Frame Relay interface configured for a server card-based L2TPv3 tunnel session:

```
class-map match-all d2
 match ip precedence 2
class-map match-all d3
 match ip precedence 3
!
policy-map o
 class d2
   bandwidth percent 10
   random-detect
   random-detect precedence 1 200 packets 500 packets 1
 class d3
   bandwidth percent 10
   random-detect
   random-detect precedence 1 1 packets 2 packets 1
!
map-class frame-relay fr-map
 service-policy output o
!
interface Serial0/1/1:0
```

```
 encapsulation frame-relay
 frame-relay interface-dlci 100 switched
 class fr-map
connect frol2tp1 Serial0/1/1:0 100 l2transport
 xconnect 10.0.3.201 666 encapsulation l2tpv3 pw-class aaa
```

Configuring Traffic Policing on an ISE/E5 Interface in a Native L2TPv3 Tunnel Session

Starting in Cisco IOS Release 12.0(30)S, QoS traffic policing is supported on the following types of Edge Engine (ISE/E5) ingress interfaces bound to a native L2TPv3 tunnel session:

- ATM
- Frame Relay DLCIs

QoS traffic shaping in a native L2TPv3 tunnel session is supported on ATM ISE/E5 egress interfaces for the following service categories:

- UBR (unspecified bit rate)
- VBR-nrt (variable bit rate nonreal-time)

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface and to partition a network into multiple priority levels or classes of service (CoS). The dual rate, 3-Color Marker in color-aware and color-blind modes, as defined in RFC 2698 for traffic policing, is supported on ingress ISE/E5 interfaces to classify packets.

The **police** command configures traffic policing using two rates, the committed information rate (CIR) and the peak information rate (PIR). The following conform, exceed, and violate values for the *actions* argument are supported with the **police** command in policy-map configuration mode on an ISE/E5 interface bound to an L2TPv3 tunnel session:

- **conform-action** *actions*: Actions taken on packets that conform to the CIR and PIR.
  - **set-prec-tunnel**: Sets the IP precedence value in the tunnel header of a packet encapsulated for native L2TPv3 tunneling.
  - **set-dscp-tunnel**: Sets the IP differentiated services code point (DSCP) value in the tunnel header of a packet encapsulated for native L2TPv3 tunneling.
  - **transmit**: Sends the packet with no alteration.
- **exceed-action** *actions*: Actions taken on packets that conform to the CIR but not the PIR.
  - **drop**: Drops the packet.
  - **set-clp** (ATM only): Sets the Cell Loss Priority (CLP) bit from 0 to 1 in an ATM cell encapsulated for native L2TPv3 tunneling.
  - **set-dscp-tunnel**: Sets the DSCP value in the tunnel header of a packet encapsulated for native L2TPv3 tunneling.
  - **set-dscp-tunnel** and **set-clp** (ATM only): Sets the DSCP value in the tunnel header and the CLP bit in an ATM cell encapsulated for native L2TPv3 tunneling.
  - **set-dscp-tunnel** and **set-frde** (Frame Relay only): Sets the DSCP value in the tunnel header and discard eligible (DE) bit in a Frame Relay packet encapsulated for native L2TPv3 tunneling.
  - **set-frde** (Frame Relay only): Sets the DE bit in a Frame Relay packet encapsulated for native L2TPv3 tunneling.
  - **set-prec-tunnel** and **set-clp** (ATM only): Sets the precedence value in the tunnel header and the CLP bit in an ATM cell encapsulated for native L2TPv3 tunneling.

  - **set-prec-tunnel** and **set-frde** (Frame Relay only): Sets the precedence value in the tunnel header and the Frame Relay DE bit in a Frame Relay packet encapsulated for native L2TPv3 tunneling.
  - **transmit**: Sends the packet with no alteration.
- **violate-action** *actions*: Actions taken on packets that exceed the PIR.
  - **drop**: Drops the packet.

You can configure these conform, exceed, and violate values for the *actions* argument of the **police** command in policy-map configuration mode on an ATM or Frame Relay ISE/E5 interface at the same time you use the **ip tos** command to configure the value of the ToS byte in IP headers of tunneled packets in a pseudowire class configuration applied to the interface (see the sections "Configuring the L2TPv3 Pseudowire" and "Manually Configuring L2TPv3 Session Parameters").

However, the values you configure with the **police** command on an ISE/E5 interface for native L2TPv3 tunneling take precedence over any IP ToS configuration. This means that the traffic policing you configure always rewrites the IP header of the tunnel packet and overwrites the values set by an **ip tos** command. The priority of enforcement is as follows when you use these commands simultaneously:

1. **set-prec-tunnel** or **set-dscp-tunnel** (QoS policing in native L2TPv3 tunnel)
2. **ip tos reflect**
3. **ip tos tos-value**

Note This behavior is designed. We recommend that you configure only native L2TPv3 tunnel sessions and reconfigure any ISE/E5 interfaces configured with the **ip tos** command to use the QoS policy configured for native L2TPv3 traffic policing.

The following example shows how to configure traffic policing using the dual rate, 3-Color Marker on an ISE/E5 Frame Relay interface in a native L2TPv3 tunnel session.

Note This example shows how to use the **police** command in conjunction with the **conform-color** command to specify the policing actions to be taken on packets in the conform-color class and the exceed-color class. This is called a color-aware method of policing and is described in "QoS: Color-Aware Policer." However, you can also configure color-blind traffic policing on an ISE/E5 Frame Relay interface in a native L2TPv3 tunnel session, using only the **police** command without the **conform-color** command.

```
class-map match-any match-not-frde
 match not fr-de
!
class-map match-any match-frde
 match fr-de
!
policy-map 2R3C_CA
 class class-default
  police cir 16000 bc 4470 pir 32000 be 4470
  conform-color match-not-frde exceed-color match-frde
  conform-action set-prec-tunnel-transmit 2
  exceed-action set-prec-tunnel-transmit 3
  exceed-action set-frde-transmit
  violate-action drop
```

The following example shows how to configure a QoS policy for traffic on the egress side of an ISE/E5 Frame Relay interface configured for a native L2TPv3 tunnel session.

Note that the sample output policy configured for a TSC-based L2TPv3 tunnel session in the section "Configuring QoS on a Frame Relay Interface in a TSC-Based L2TPv3 Tunnel Session" is not supported on a Frame Relay ISE/E5 interface. QoS policies on per-DLCI output traffic are not supported on ISE/E5 interfaces configured for a native L2TPv3 tunnel.

```
policy-map o
 class d2
  bandwidth percent 10
  random-detect precedence 1 200 packets 500 packets 1
 class d3
  bandwidth percent 10
  random-detect precedence 1 1 packets 2 packets 1
!
interface Serial0/1/1:0
 encapsulation frame-relay
 frame-relay interface-dlci 100 switched
  class fr-map
 service output o
```

Configuring Tunnel Marking in a Native L2TPv3 Tunnel Session

The QoS: Tunnel Marking for L2TPv3 Tunnels feature allows you to set (mark) either the IP precedence value or the differentiated services code point (DSCP) in the header of an L2TPv3 tunneled packet, using the **set-prec-tunnel** or **set-dscp-tunnel** command without configuring QoS traffic policing. Tunnel marking simplifies administrative overhead previously required to control customer bandwidth by allowing you to mark the L2TPv3 tunnel header on an ingress ISE/E5 interface.

The following example shows how to configure tunnel marking using MQC **set** commands for the default traffic class and a traffic class that matches a specified Frame Relay DE bit value:

```
class-map match-any match-frde
 match fr-de
policy-map set_prec_tun
 class match-frde
  set ip precedence tunnel 1
 class class-default
  set ip precedence tunnel 2
!
map-class frame-relay fr_100
 service-policy input set_prec_tun
```

L2TPv3 Customer-Facing ISE/E5 Interface

```
interface POS0/0
 frame-relay interface-dlci 100 switched
  class fr_100
```

Configuring Traffic Shaping in a Native L2TPv3 Tunnel Session

The following example shows how to configure traffic shaping on a Frame Relay ISE/E5 egress interface bound to a native L2TPv3 tunnel session. You can configure traffic shaping on a Frame Relay main egress interface by classifying traffic with different class maps.

Note You cannot configure per-DLCI shaping using the method shown in this example to configure traffic shaping.

To configure class-based shaping, configure the **match qos-group** and **random-detect discard-class** values according to the incoming IP precedence and DSCP values from packets received on the backbone-facing ingress interface. Use these values to define traffic classes on the customer-facing egress interface.

```
class-map match-any match_prec1
 match ip precedence 1
class-map match-any match_prec2
 match ip precedence 2
class-map match-any match_prec3
 match ip precedence 3
!
class-map match-all match_qos3
 match qos-group 3
!
class-map match-any match_qos12
 match qos-group 1
 match qos-group 2
!
policy-map customer_egress_policy
 class match_qos3
  bandwidth percent 5
  shape average 160000000
 class match_qos12
  shape average 64000000
  random-detect discard-class-based
  random-detect discard-class 1 500 packets 1000 packets
  random-detect discard-class 2 1000 packets 2000 packets
  bandwidth percent 10
 class class-default
  shape average 64000000
  queue-limit 1000 packets
  bandwidth percent 1
!
policy-map backbone_ingress_policy
 class match_prec1
  set qos-group 1
  set discard-class 1
 class match_prec2
  set qos-group 2
  set discard-class 2
 class match_prec3
  set qos-group 3
  set discard-class 3
 class class-default
  set qos-group 5
  set discard-class 5
```

L2TPv3 Customer-Facing ISE/E5 Interface

```
interface POS0/0
 service-policy output customer_egress_policy
 frame-relay interface-dlci 100 switched
  class fr_100
```

L2TPv3 Backbone-Facing ISE/E5 Interface

```
interface POS1/0
 service-policy input backbone_ingress_policy
```

Configuring a QoS Policy for Committed Information Rate Guarantees: Example

The following example shows how to configure a QoS policy that guarantees a CIR of 256 kbps on DLCI 100 and 512 kbps for DLCI 200 on a serial interface at one end of a TSC-based L2TPv3 tunnel session:

```
ip cef distributed
 class-map dlci100
 match fr-dlci 100
 class-map dlci200
 match fr-dlci 200
!
policy-map dlci
 class dlci100
 bandwidth 256
 class dlci200
 bandwidth 512
!
interface Serial 0/0
 encapsulation frame-relay
 frame-relay intf-type dce
 service-policy output dlci
!
connect one Serial 0/0 100 l2transport
 xconnect 10.0.3.201 555 encapsulation l2tpv3 pw-class mqc
!
connect two Serial 0/0 200 l2transport
 xconnect 10.0.3.201 666 encapsulation l2tpv3 pw-class mqc
```

Setting the Frame Relay DE Bit Configuration: Example

The following example shows how to configure the service policy called set-de and attach it to an output serial interface bound to a TSC-based L2TPv3 tunnel session. Note that setting the Frame Relay DE bit is not supported on a Frame Relay ISE/E5 interface bound to a native L2TPv3 tunnel session.

In this example, the class map called data evaluates all packets exiting the interface for an IP precedence value of 1. If the exiting packet has been marked with the IP precedence value of 1, the packet's DE bit is set to 1.

```
class-map data
 match qos-group 1
!
policy-map SET-DE
 class data
  set fr-de
!
interface Serial 0/0/0
 encapsulation frame-relay
 service-policy output SET-DE
!
connect fr-mpls-100 serial 0/0/0 100 l2transport
 xconnect 10.10.10.10 pw-class l2tpv3
```

Matching the Frame Relay DE Bit Configuration: Example

The following example shows how to configure the service policy called match-de and attach it to an interface bound to a TSC-based L2TPv3 tunnel session. In this example, the class map called "data" evaluates all packets entering the interface for a DE bit setting of 1. If the entering packet has been a DE bit value of 1, the packet's IP precedence value is set to 3.

```
class-map data
 match fr-de
!
policy-map MATCH-DE
 class data
  set ip precedence tunnel 3
!
ip routing
ip cef distributed
!
mpls label protocol ldp
interface Loopback0
 ip address 10.20.20.20 255.255.255.255
!
interface Ethernet1/0/0
 ip address 172.16.0.2 255.255.255.0
 tag-switching ip
!
interface Serial4/0/0
 encapsulation frame-relay
service input MATCH-DE
!
connect 100 Serial4/0/0 100 l2transport
 xconnect 10.10.10.10 100 encapsulation l2tpv3
```

The next example shows how to configure the service policy called set_prec_tunnel_from_frde and attach it to a Cisco 12000 series ISE/E5 interface bound to a native L2TPv3 tunnel session. Note that in a native L2TPv3 session, you must attach the service policy to a DLCI (in the example, DCLI 100) instead of to a main interface (as in the preceding example).

```
class-map match-any match-frde
  match fr-de
!
policy-map set_prec_tunnel_from_frde
  class match-frde
   set ip precedence tunnel 6
  class class-default
   set ip precedence tunnel 3
!
map-class frame-relay fr_100
  service-policy input set_prec_tunnel_from_frde
!
interface POS0/0
  description ISE: L2TPv3 Customer-facing interface
  frame-relay interface-dlci 100 switched
    class fr_100
```

Configuring MLFR for L2TPv3 on the Cisco 12000 Series: Example

The following example shows how to configure L2TPv3 tunneling on a multilink Frame Relay bundle interface on a Cisco 12000 series 2-port Channelized OC-3/STM-1 (DS1/E1) or 6-port Channelized T3 line card:

```
frame-relay switching
!
pseudowire-class mfr
 encapsulation l2tpv3
 ip local interface Loopback0
!
interface mfr0
 frame-relay intf-type dce
```

```
!
interface Serial0/0.1/1:11
 encapsulation frame-relay MFR0
!
interface Serial0/0.1/1:12
 encapsulation frame-relay MFR0
!
connect L2TPoMFR MFR0 100 l2transport
 xconnect 10.10.10.10 3 pw-class mfr
```

Configuring an MQC for Committed Information Rate Guarantees: Example

The following is a sample configuration of the MQC to guarantee a CIR of 256 kbps on DLCI 100 and 512 kbps for DLCI 200:

```
ip cef distributed
 class-map dlci100
 match fr-dlci 100
 class-map dlci200
 match fr-dlci 200
!
policy-map dlci
 class dlci100
 bandwidth 256
 class dlci200
 bandwidth 512
!
interface Serial0/0
 encapsulation frame-relay
 frame-relay intf-type dce
 service-policy output dlci
!
connect one Serial0/0 100 l2transport
 xconnect 10.0.3.201 555 encapsulation l2tpv3 pw-class mqc
!
connect two Serial0/0 200 l2transport
 xconnect 10.0.3.201 666 encapsulation l2tpv3 pw-class mqc
```

Additional References

The following sections provide references related to the L2TPv3 feature.

Related Documents

Related Topic	Document Title
L2TPv3	Layer 2 Tunneling Protocol Version 3 Technical Overview
L2VPN interworking	"L2VPN Interworking" chapter in the *Cisco IOS Multiprotocol Label Switching Configuration Guide*
L2VPN pseudowire switching	"L2VPN Pseudowire Switching" chapter in the *Cisco IOS Multiprotocol Label Switching Configuration Guide*
L2VPN pseudowire redundancy	"L2VPN Pseudowire Redundancy" chapter in the *Cisco IOS Wide-Area Networking Configuration Guide*

Related Topic	Document Title
L2TP	• Layer 2 Tunnel Protocol • Layer 2 Tunneling Protocol: A Feature in Cisco IOS Software
Configuring CEF	"Part 1: Cisco Express Forwarding" in the *Cisco IOS IP Switching Configuration Guide*
MTU discovery and packet fragmentation	MTU Tuning for L2TP
Tunnel marking for L2TPv3 tunnels	QoS: Tunnel Marking for L2TPv3 Tunnels
Multilink Frame Relay over L2TPv3/AToM	Multilink Frame Relay over L2TPv3/AToM
Additional VPN commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS Dial Technologies Command Reference*
Additional Frame Relay commands: complete command syntax, command mode, defaults, usage guidelines and examples	*Cisco IOS Wide-Area Networking Command Reference*
UTI	Universal Transport Interface (UTI)
IPv6	*Cisco IOS IPv6 Configuration Guide*
Additional IPv6 commands: complete command syntax, command mode, defaults, usage guidelines and examples	*Cisco IOS IPv6 Command Reference*

Standards

Standard	Title
draft-ietf-l2tpext-l2tp-base-03.txt	Layer Two Tunneling Protocol (Version 3) "L2TPv3"

MIBs

MIB	MIBs Link
• VPDN MIB—MIB support for L2TPv3 is based on the VPDN MIB	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
RFC 2661	*Layer Two Tunneling Protocol "L2TP"*
RFC 1321	*The MD5 Message Digest Algorithm*
RFC 2104	*HMAC-Keyed Hashing for Message Authentication*
RFC 3931	*Layer Two Tunneling Protocol Version 3 "L2TPv3*

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 Layer 2 Tunnel Protocol Version 3

Table 9 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Releases 12.2(1) or 12.0(3)S or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

The following commands were introduced or modified for the Layer 2 Tunnel Protocol Version 3 feature and related features: **atm mcpt-timers**, **atm pvp**, **authentication (L2TP)**, **cell-packing**, **clear l2tun**, **clear l2tun counters**, **clear l2tun counters tunnel l2tp**, **clear l2tun tunnel counters**, **debug acircuit**, **debug atm cell-packing**, **debug condition xconnect**, **debug vpdn**, **debug xconnect**, **digest**, **digest check**, **encapsulation l2tpv3**, **hello**, **hidden**, **hostname (L2TP)**, **ip dfbit set**, **ip local interface**, **ip pmtu**, **ip protocol**, **ip tos (L2TP)**, **ip ttl**, **l2tp cookie local**, **l2tp cookie remote**, **l2tp hello**, **l2tp id**, **l2tp-class**, **match fr-de**, **match protocol (L2TPv3)**, **monitor l2tun counters tunnel l2tp**, **oam-ac emulation-enable**, **password (L2TP)**, **protocol (L2TP)**, **pseudowire-class**, **receive-window**, **retransmit**, **sequencing**, **show atm cell-packing**, **show l2tun**, **show l2tun counters tunnel l2tp**, **show l2tun session**, **show l2tun tunnel**, **show xconnect**, **snmp-server enable traps l2tun pseudowire status**, **snmp-server enable traps l2tun session**, **snmp-server host**, **timeout setup**, **xconnect**, and **xconnect logging pseudowire status**.

Note Table 9 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 9 Feature Information for Layer 2 Tunnel Protocol Version 3

Release	Modification
Cisco IOS Release 12.0	
12.0(21)S	Initial data plane support for L2TPv3 was introduced on the Cisco 7200 series, Cisco 7500 series, Cisco 10720, and Cisco 12000 series platforms.
12.0(23)S	L2TPv3 control plane support was introduced on the Cisco 7200 series, Cisco 7500 series, Cisco 10720, and Cisco 12000 series platforms.
12.0(24)S	L2TPv3 was enhanced to support the Layer 2 Fragmentation feature (fragmentation of IP packets before they enter the pseudowire) on the Cisco 7200 series, Cisco 7500 series, and Cisco 12000 series Internet routers.
12.0(25)S	Support was added for the ATM VP Mode Single Cell Relay over L2TPv3 feature on the Cisco 7200 and Cisco 7500 series routers with ATM Deluxe PA-A3 interfaces. L2TPv3 control plane support was introduced on the Cisco 12000 series 1-port channelized OC-12 (DS3) line card.
12.0(23)S3	L2TPv3 control plane support was introduced on the Cisco 12000 series 1-port channelized OC-12 (DS3) line card.

Table 9 Feature Information for Layer 2 Tunnel Protocol Version 3 (Continued)

12.0(24)S1	L2TPv3 control plane support was introduced on the Cisco 12000 series 1-port channelized OC-12 (DS3) line card.
12.0(27)S	Support was added for the following features to Cisco 12000 series 2-port channelized OC-3/STM-1 (DS1/E1) and 6-port Channelized T3 (T1) line cards: • Binding L2TPv3 sessions to Multilink Frame Relay (MLFR) interfaces • Quality of service (QoS) for Frame Relay attachment circuits
12.0(28)S	Support was added for the following features on the Cisco 7200 series and Cisco 7500 series routers: • ATM AAL5 OAM Emulation over L2TPv3 • ATM Single Cell Relay VC Mode over L2TPv3 • L2TPv3 Distributed Sequencing • L2TPv3 Support for PA-A3-8T1IMA PA and PA-A3-8E1IMA Port Adapters
12.0(29)S	Support was added for the following features: • ATM Cell Packing over L2TPv3 • ATM Port Mode Cell Relay over L2TPv3 • L2TPv3 Control Message Hashing • L2TPv3 Control Message Rate Limiting • Protocol Demultiplexing for L2TPv3
12.0(30)S	Support was added for the following features to Cisco IOS Release 12.0(30)S: • L2TPv3 Digest Secret Graceful Switchover • Manual Clearing of L2TPv3 Tunnels • VC Class Provisioning for L2VPN Support was added for native L2TPv3 tunneling on IP services engine (ISE) line cards on the Cisco 12000 series Internet router.
12.0(31)S	Support was added for the following feature to Cisco IOS Release 12.0(31)S: • Layer 2 VPN (L2VPN): Syslog, SNMP Trap, and show Command Enhancements for AToM and L2TPv3 Support was added for native L2TPv3 tunneling on the following ISE line cards on the Cisco 12000 series Internet router: • 2.5G ISE SPA Interface Processor (SIP): – 2-port T3/E3 serial shared port adapter (SPA) – 4-port T3/E3 serial SPA – 2-port channelized T3 SPA – 4-port channelized T3 Serial SPA • 4-port Gigabit Ethernet ISE

Table 9 Feature Information for Layer 2 Tunnel Protocol Version 3 (Continued)

12.0(31)S2	Support was added for customer-facing IP Services Engine (ISE) interfaces configured for Layer 2 local switching on a Cisco 12000 series Internet router (see *Layer 2 Local Switching*).
12.0(32)SY	Support was added for Engine 5 line cards — shared port adapters (SPAs) and SPA interface processors (SIPs) — on the Cisco 12000 series Internet router, including: • Engine-5 customer-facing interfaces that are configured for local switching (see *Layer 2 Local Switching*). • Engine-5 and ISE (Engine-3) interfaces that are configured for Layer 2 VPN interworking (see *L 2VPN Interworking*). Support was added for the L2TPv3 Layer 2 fragmentation feature on the Cisco 10720 Internet router.
12.0(33)S	Support was added for the following features to Cisco IOS Release 12.0(33)S: • Protocol Demultiplexing for L2TPv3 for PPP traffic • Protocol Demultiplexing for L2TPv3 for HDLC traffic • Protocol Demultiplexing for L2TPv3 on Engine-3/Engine-5 line cards in the Cisco 12000 series platforms • Protocol Demultiplexing for L2TPv3 on Engine-3/Engine-5 line cards in the Cisco 12000 series platforms for PPP, HDLC, Ethernet and Frame-relay encapsulations • Color Aware Policer on Engine-3/Engine-5 line cards for Ethernet over L2TPv3 • Site of Origin for Border Gateway Protocol Virtual Private Networks (BGP-VPNs) • Control Message Statistics and Conditional Debugging Command Enhancements (including L2VPN Pseudowire Conditional Debugging)
Cisco IOS Release 12.2S	
12.2(25)S	Support was added for the following features to Cisco IOS Release 12.2(25)S: • L2TPv3: Layer 2 Tunneling Protocol • ATM AAL5 OAM Emulation over L2TPv3 • ATM Single Cell Relay VC Mode over L2TPv3 • ATM VP Mode Single Cell Relay over L2TPv3 • L2TPv3 Distributed Sequencing • L2TPv3 Layer 2 fragmentation • L2TPv3 Support for PA-A3-8T1IMA PA and PA-A3-8E1IMA Port Adapters

Table 9 Feature Information for Layer 2 Tunnel Protocol Version 3 (Continued)

12.2(25)S4	Support was added for the following features on the Cisco 7304 NPE-G100 and the Cisco 7304 NSE-100: • L2TPv3: Layer 2 Tunneling Protocol • ATM AAL5 OAM Emulation over L2TPv3 • ATM Port Mode Cell Relay over L2TPv3 • ATM Single Cell Relay VC Mode over L2TPv3 • ATM VP Mode Single Cell Relay over L2TPv3 • L2TPv3 Layer 2 fragmentation Support was added for this feature on the Cisco 7304 NPE-G100 only: • L2TPv3 Distributed Sequencing
Cisco IOS Release 12.2SB	
12.2(27)SBC	Support was added for the following features: • L2TPv3 Control Message Hashing • L2TPv3 Control Message Rate Limiting • Layer 2 VPN (L2VPN): Syslog, SNMP Trap, and show Command Enhancements for AToM and L2TPv3 • Protocol Demultiplexing for L2TPv3
12.2(28(SB)	Support was added for Control Message Statistics and Conditional Debugging Command Enhancements (including L2VPN Pseudowire Conditional Debugging)
Cisco IOS Release 12.2SR	
12.2(33)SRC	The Layer 2 Tunnel Protocol Version 3 feature was integrated into Cisco IOS Release 12.2(33)SRC and implemented on the Cisco 7600 series SPA Interface Processor-400 (SIP-400) linecard.
Cisco IOS Release 12.3T	
12.3(2)T	The Layer 2 Tunnel Protocol Version 3 feature was integrated into Cisco IOS Release 12.3(2)T and implemented on the Cisco 2600XM series Multiservice platforms, the Cisco 2691 Multiservice routers, the Cisco 3662 Multiservice Access platforms, the Cisco 3725 Modular Access routers, and the Cisco 3745 Modular Access routers.
Cisco IOS Release 12.4T	
12.4(11)T	Support was added for the following features: • L2TPv3 Control Message Hashing • L2TPv3 Control Message Rate Limiting • Protocol Demultiplexing for L2TPv3

Glossary

AV pairs—attribute-value pairs.

BECN—backward explicit congestion notification. Bit set by a Frame Relay network in frames traveling in the opposite direction of frames encountering a congested path. DTE receiving frames with the BECN bit set can request that higher-level protocols take flow control action as appropriate.

CE—customer edge (Frame Relay switch or user device).

CIR—committed information rate. Rate at which a Frame Relay network agrees to transfer information under normal conditions, averaged over a minimum increment of time. CIR, measured in bits per second, is one of the key negotiated tariff metrics.

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.

dCEF—distributed Cisco Express Forwarding.

DLCI—data-link connection identifier. A unique number assigned to a PVC endpoint in a Frame Relay network. Identifies a particular PVC endpoint within an access channel in a Frame Relay network and has local significance only to that channel.

DTE—data terminal equipment. Device at the user end of a user-network interface that serves as a data source, destination, or both.

FECN—forward explicit congestion notification. Bit set by a Frame Relay network to inform DTE receiving the frame that congestion was experienced in the path from source to destination. DTE receiving frames with the FECN bit set can request that higher-level protocols take flow-control action as appropriate.

HDLC—High-Level Data Link Control. A generic link-level communications protocol developed by the International Organization for Standardization (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. OSI link-state hierarchical routing protocol based on DECnet Phase V routing, whereby ISs (routers) exchange routing information based on a single metric to determine network topology.

L2TP—An extension to PPP merging features of two tunneling protocols: Layer 2 Forwarding (L2F) from Cisco Systems and Point-to-Point Tunneling (PPTP) from Microsoft. L2TP is an Internet Engineering Task Force (IETF) standard endorsed by Cisco Systems, and other networking industry leaders.

L2TPv3—Draft version of L2TP that enhances functionality in RFC 2661 (L2TP).

LMI—Local Management Interface.

MPLS—Multiprotocol Label Switching. Switching method that forwards IP traffic using a label. This label instructs the routers and the switches in the network where to forward the packets based on preestablished IP routing information.

MQC—modular quality of service command-line interface.

MTU—maximum transmission unit. Maximum packet size, in bytes, that a particular interface can handle.

NNI—Network-to-Network Interface. ATM Forum standard that defines the interface between two ATM switches that are both located in a private network or are both located in a public network. The UNI standard defines the interface between a public switch and a private one. Also, the standard interface between two Frame Relay switches meeting the same criteria.

PE—Provider edge router providing Frame Relay over L2TPv3 functionality.

PPP—Point-to-Point Protocol. A link-layer encapsulation method for dialup or dedicated circuits. A successor to Serial Line IP (SLIP), PPP provides router-to-router and host-to-network connections over synchronous and asynchronous circuits.

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 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. Network management protocol used almost exclusively in TCP/IP networks. SNMP provides a means to monitor and control network devices, and to 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.

UTI—Universal Transport 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 routers. A VPDN enables users to configure secure networks that take advantage of ISPs that tunnel remote access traffic through the ISP cloud.

WAN—wide-area network. Data communications network that serves users across a broad geographic area and often uses transmission devices provided by common carriers. Frame Relay, SMDS, and X.25 are examples of WANs.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco Nurse Connect, Cisco Pulse, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, and Flip Gift Card are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Fast Step, Follow Me Browsing, FormShare, GainMaker, GigaDrive, HomeLink, iLYNX, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0908R)

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.

© 2001–2009 Cisco Systems, Inc. All rights reserved.

Configuring SMDS

The Switched Multimegabit Data Service (SMDS) is a WAN service offered by a variety of service providers. This chapter describes the configuration tasks for the SMDS packet-switched software.

For further general information about SMDS, see the chapter "Wide-Area Networking Overview" at the beginning of this book.

For a complete description of the commands mentioned in this chapter, refer to the chapter "SMDS Commands" in the Cisco IOS Wide-Area Networking Command Reference. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the section "Identifying Supported Platforms" in the chapter "Using Cisco IOS Software."

SMDS Hardware Requirements

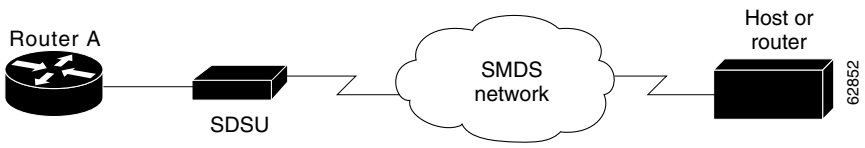
You need the following hardware, equipment, and special software to configure SMDS:

- CSC-MCI or CSC-SCI serial interface controller card, or a HSSI interface on chassis-based systems, or the serial port on a router

  To operate on CSC-SCI or CSC-MCI cards, SMDS requires that the appropriate microcode version be installed. Version numbers are 1.2 (or later) for CSC-SCI and 1.7 (or later) for CSC-MCI.
- EIA/TIA-449 or V.35 applique
- SMDS data service unit (SDSU) device

Figure 1 illustrates the connections among the components.

Figure 1 Typical SMDS Configuration



Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

SMDS Addresses

All addresses for SMDS service are assigned by the service provider and can be assigned to individuals and groups.

You must enter addresses in the Cisco SMDS configuration software using an E prefix for multicast addresses and a C prefix for unicast addresses.

Cisco IOS software expects the addresses to be entered in E.164 format, which is 64 bits (15-digit addressing). The first 4 bits are the address type, and the remaining 60 bits are the address. If the first 4 bits are 1100 (0xC), the address is a unicast SMDS address, which is the address of an individual SMDS host. If the first 4 bits are 1110 (0xE), the address is a multicast SMDS address, which is used to broadcast a packet to multiple end points. The 60 bits of the address are in binary-coded decimal (BCD) format. Each 4 bits of the address field presents a single telephone number digit, allowing for up to 15 digits. At a minimum, you must specify at least 11 digits (44 bits). Unused bits at the end of this field are filled with ones.

Note The **arp smds** command supports 48-bit addresses only (C or E followed by 11 digits). The addresses must be entered in dotted notation—for example, C141.5556.1414.

An example of a 15-digit E.164 address follows:

```
C14155561313FFFF
```

Note Older versions of Cisco IOS software supported 48-bit SMDS addresses. If, when using the current version of the software, you write the configuration to NVRAM, the full 64-bit SMDS address is written. Older versions of the software will no longer be able to read the new SMDS configuration from NVRAM. However, the current version of the software can read previous versions of the configuration in NVRAM.

The addresses can be entered with periods in a manner similar to Ethernet-style notation, or simply as a string of digits.

The following is an example of an individual address entered in Ethernet-style notation:

```
C141.5555.1212.FFFF
```

The following is an example of a group address:

```
E180.0999.9999.FFFF
```

SMDS Configuration Task List

Before you can begin the configuration tasks, you must have already obtained your SMDS addresses from your service provider. You need the following two types of addresses:

- The group address for broadcasts
- The SMDS hardware (individual) address for each router that interfaces directly into the SMDS network (that is, customer premises equipment)

You must perform basic steps to enable SMDS. In addition, you can customize SMDS for your particular network needs and monitor SMDS connections. Perform the tasks in the following sections:

- Enabling SMDS on the Interface

- Customizing Your SMDS Network
- Monitoring the SMDS Connection

See the section "SMDS Configuration Examples," at the end of this chapter, for ideas of how to configure SMDS on your network.

Enabling SMDS on the Interface

Perform the tasks in the following sections to enable SMDS:

- Setting SMDS Encapsulation
- Specifying the SMDS Address
- Establishing Address Mapping
- Mapping a Multicast Address to an SMDS Address
- Enabling ARP
- Enabling Broadcast ARP Messages
- Enabling Dynamic Address Mapping for IPX over SMDS

Setting SMDS Encapsulation

To set SMDS encapsulation at the interface level, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# encapsulation smds`	Enables SMDS on the interface.

For examples of enabling SMDS encapsulation, see the "SMDS Configuration Examples" section later in this chapter.

Specifying the SMDS Address

To specify the SMDS individual address for a particular interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# smds address smds-address`	Enters an individual address provided by the SMDS service provider.

For examples of specifying the SMDS address, see the examples in the section "SMDS Configuration Examples" later in this chapter.

Establishing Address Mapping

Routing tables are configured dynamically when DECnet, extended AppleTalk, IP, IPX, and ISO CLNS routing are configured. However, you can configure static mapping for these protocols, if needed. For other protocols, you must configure a static map between an individual SMDS address and a higher-level protocol address.

To establish address mapping, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# smds static-map` *protocol protocol-address smds-address* `[broadcast]`	Defines static entries for those routers that are SMDS remote peers.

The supported protocols and the keywords to enable them are as follows:

- AppleTalk—**appletalk**
- Banyan VINES—**vines**
- DECnet—**decnet**
- IP—**ip**
- ISO CLNS—**clns**
- Novell IPX—**ipx**
- XNS—**xns**

For examples of establishing address mapping, see the "SMDS Configuration Examples" section later in this chapter.

Mapping a Multicast Address to an SMDS Address

You can map an SMDS group address to a broadcast or multicast address used by a higher-level protocol. If you do so, you need not specify the **broadcast** keyword in the **smds static-map** command, and the Cisco IOS software need not replicate each broadcast address.

To map an SMDS group address to a multicast address, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# smds multicast` *protocol smds-address*	Maps an SMDS group address to a multicast address used by a higher-level protocol.

The protocols supported and the keywords to enable them are as follows. Note that bridging is not a protocol, but the **bridge** keyword is valid for providing a map to a multicast address.

- AppleTalk—**appletalk**
- AppleTalk ARP address—**aarp**
- Banyan VINES—**vines**
- Bridging—**bridge**
- DECnet—**decnet**

- DECnet multicast address for all Level 1 routers—**decnet_router-L1**
- DECnet multicast address for all Level 2 routers—**decnet_router-L2**
- DECnet multicast address for all end systems—**decnet_node**
- IP—**ip**
- ISO CLNS—**clns**
- Multicast address for all CLNS intermediate systems—**clns_is**
- Multicast address for all CLNS end systems—**clns_es**
- Novell IPX—**ipx**
- XNS—**xns**

For examples of mapping to a multicast address, see the "SMDS Configuration Examples" later in this chapter.

Enabling ARP

When you enable the Address Resolution Protocol (ARP), you can choose to enable either a dynamic ARP cache or one built statically. To enable ARP, use one of the following commands in the specified configuration mode:

Command	Purpose
`Router(config-if)# smds enable-arp`	Enables ARP and dynamic address resolution (interface).
`Router(config)# arp` *ip-address smds-address* `smds`	Enables ARP with a static entry for the remote router (global).

An SMDS network can be thought of in much the same way as an X.25 cloud. The premises equipment (in this case Cisco routers) represents the edge of the cloud. The service provider enables communication across the cloud. However, proper configuration is needed for communication to occur. This configuration will differ from one protocol family to another.

One major difference between protocol families is dynamic versus static routing among the routers (called *remote peers*) on the periphery of the cloud. For IP, routing across the SMDS cloud is fully dynamic. No action on the user's part is needed to map higher-level protocol addresses to SMDS addresses. Both IP and ARP can be configured and a dynamic ARP routing table enabled.

Note The **arp smds** command requires 12-digit dotted-notation SMDS addresses—for example, C141.5678.9012.

See the section "Configuring Specific Protocols," later in this chapter, for more information about configuring higher-level protocols.

Enabling Broadcast ARP Messages

When an ARP server is present in the network, you can enable broadcast ARP messages that are sent to all ARP SMDS addresses or to all IP SMDS multicast addresses when ARP addresses are not present.

To enable broadcast ARP messages, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# smds enable-arp`	Enables ARP and dynamic address resolution.
Step 2	`Router(config-if)# smds multicast arp` *smds-address* [*ip-address mask*]	Enables broadcast ARP messages.

For an example of how to enable broadcast ARP messages, see the section "Typical Multiprotocol Configuration Example" later in this chapter.

Enabling Dynamic Address Mapping for IPX over SMDS

To enable dynamic address mapping for IPX on an SMDS interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# smds glean ipx` [*timeout-value*] [**broadcast**]	Enables dynamic address mapping for IPX.

For an example of how to enable dynamic address mapping for IPX over SMDS, see the section "IPX Dynamic Address Mapping Example" later in this chapter.

Customizing Your SMDS Network

Perform the tasks in the following sections to customize your SMDS network:

- Configuring Specific Protocols
- Enabling Transparent Bridging over SMDS
- Configuring SMDS Subinterfaces for Multiple Logical IP Subnetworks
- Reenabling Data Exchange Interface Version 3.2 with Heartbeat Support
- Configuring Pseudobroadcasting
- Enabling Fast Switching

Configuring Specific Protocols

Some protocol families are dynamically routed. For IP and CLNS, routing is fully dynamic, and no action on your part is needed to map higher-level protocol addresses to SMDS addresses. But for the other supported protocols, you must make a static entry for each router to communicate with all other peer routers. The static entries need to be made only for those routers that are SMDS remote peers. Nothing additional needs to be done to assure communication with other nodes behind the peer routers.

For an example of how to configure specific protocols, see the section "Typical Multiprotocol Configuration Example" later in this chapter.

Table 1 lists protocol families and the multicasts that are needed.

Table 1 Protocol Families and Types of Multicasts Needed

Protocol Family	Multicasts Needed
IP	IP
DECnet	DECNET, DECNET_NODE, DECNET_ROUTER-L1, DECNET_ROUTER-L2
CLNS	CLNS, CLNS_ES,CLNS_IS
Novell IPX	IPX
XNS	XNS
AppleTalk	APPLETALK, AARP
Banyan VINES	VINES

Configuring ARP and IP

For both IP and ARP, the multicast address must be configured and ARP must be enabled. ARP multicast is required only for ARP servers; the IP multicast is used for ARP and routing updates.

Configuring DECnet

Static maps must be configured for DECnet. In addition, a separate **smds multicast** command is needed for DECNET, DECNET_NODE, DECNET_ROUTER-L1, and DECNET_ROUTER-L2.

Configuring CLNS

Multicasts must be configured for CLNS_ES and CLNS_IS. No static maps are necessary. End system hello (ESH), intermediate system hello (ISH), and router hello packets are sent to the multicast address, and neighbor entries are created automatically.

Configuring IPX

For Novell IPX, the multicast address must be configured. A static map entry can be made for each remote peer, or you can use the **smds glean** command to dynamically map addresses. Static map entries override any dynamic map entries.

Routing Information Protocol (RIP) routing packets, Service Advertisement Protocol (SAP) packets, NetBIOS Name Lookups, directed broadcasts, and traffic to the helper addresses (if that helper address is a broadcast address) are sent to the SMDS IPX multicast address.

Configuring XNS

For XNS, the multicast address must be configured, and a static map entry must be made for each remote peer. Only RIP, directed broadcasts, and helper traffic are sent to the XNS multicast address.

Configuring AppleTalk

The SMDS cloud must be treated by all AppleTalk routers connected to it as either extended or nonextended. The network types cannot be mixed on the same SMDS cloud. Instead, all AppleTalk routers on an SMDS cloud must agree about the network type: extended or nonextended.

If any router in the SMDS cloud uses Cisco IOS Release 10.3(3) (or earlier), use a nonextended AppleTalk configuration for the SMDS cloud. To use nonextended AppleTalk, use the **appletalk address** command and configure static maps.

If all routers in the SMDS cloud use Cisco IOS Release 10.3(4) (or later), you can use extended AppleTalk to support dynamic AARP for SMDS addresses. To use extended AppleTalk, use the **appletalk cable-range** command.

For information on the **appletalk address** and **appletalk cable-range** commands, refer to the *Cisco IOS AppleTalk and Novell IPX Command Reference*.

For an example of how to configure AppleTalk, see the section "AppleTalk Configuration Examples" later in this chapter.

Configuring Banyan VINES

For Banyan VINES, the multicast address must be configured. Also note that VINES works only with static maps.

Enabling Transparent Bridging over SMDS

You can enable transparent bridging for SMDS encapsulated serial and HSSI interfaces. Cisco's implementation of IEEE 802.6i transparent bridging for SMDS supports 802.3, 802.5, and FDDI frame formats. The router can accept frames with or without frame check sequence (FCS).

Fast-switched transparent bridging is the default and is not configurable. If a packet cannot be fast switched, it will be process switched.

To enable transparent bridging, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Specifies a serial or HSSI interface.
Step 2	`Router(config-if)# encapsulation smds`	Configures SMDS encapsulation on the serial interface.
Step 3	`Router(config-if)# bridge-group bridge-group`	Associates the interface with a bridge group.
Step 4	`Router(config-if)# smds multicast bridge smds-address`	Configures bridging across SMDS.

For more information about bridge groups and the **bridge-group** command, see the "Configuring Transparent Bridging" chapter in the *Cisco IOS Bridging and IBM Networking Configuration Guide*.

Configuring SMDS Subinterfaces for Multiple Logical IP Subnetworks

Multiple logical IP subnetworks are supported as defined by RFC 1209. This RFC explains routing IP over an SMDS cloud where each connection is considered a host on one specific private network, and describes cases where traffic must transit from network to network.

This solution allows a single SMDS interface to be treated as multiple logical IP subnetworks and to support routing of packets from one network to the next without using intervening routers. When multiple logical IP subnetworks are enabled, the router performs routing between the subnetworks using IP addresses on an SMDS interface. Each supported subnetwork has an IP address, a unicast SMDS E.164 address, and a multicast SMDS E.164 address configured on the SMDS interface. Broadcast packets are duplicated and transmitted to all IP networks on the specified SMDS interface and use the associated multicast SMDS address for the network.

Only routers that require knowledge of multiple IP networks need to be configured with multipoint subinterfaces that correspond to different networks.

To configure the Cisco IOS software to have multipoint subinterfaces for multiple logical IP subnetworks, use the following commands beginning in global configuration mode:

	Command	Purpose

Step 1	`Router(config)# interface serial` *interface.subinterface* `multipoint`
	
	`Router(config)# interface serial` *slot/port.subinterface* `multipoint`
	(for Cisco 7000 series routers[1])
		Defines a logical subinterface for each IP network.

Step 2	`Router(config-if)# ip address` *ip-address mask*	Configures the subinterface as an IP network.
Step 3	`Router(config-if)# smds address` *smds-address*	Assigns unicast SMDS E.164 address to the subinterface.
Step 4	`Router(config-if)# smds multicast` *protocol smds-address*	Assigns multicast SMDS E.164 address for each protocol supported on the subinterface.
Step 5	`Router(config-if)# smds enable-arp`	Enables ARP on the subinterface, if required by the protocol.

1. Beginning in Cisco IOS Release 11.3, all commands supported on the Cisco 7500 series are also supported on the Cisco 7000 series.

For an example of how to configure multiple logical IP subnetworks, see the "Multiple Logical IP Subnetworks over SMDS Example" section later in this chapter.

Reenabling Data Exchange Interface Version 3.2 with Heartbeat Support

By default, SMDS provides the Data Exchange Interface (DXI) Version 3.2 *heartbeat* process as specified in the SIG-TS-001/1991 standard. The DXI mechanism encapsulates SMDS packets in a DXI frame before they are transmitted. The heartbeat mechanism automatically generates a heartbeat poll frame every 10 seconds. The Interim Local Management Interface (ILMI) is not supported. See the *Cisco IOS Wide-Area Networking Command Reference* for more information about DXI 3.2.

Note If you are running serial lines back-to-back, disable keepalive on SMDS interfaces. Otherwise, DXI declares the link down.

If you find you must reenable the DXI heartbeat, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# smds dxi`	Enables DXI 3.2.

Configuring Pseudobroadcasting

Some hosts do not support multicast E.164 addresses. This is a problem in IP where frequent broadcast packets are sent because routing updates are generally broadcast. IP and ARP depend on the use of multicast addresses to determine a route to a destination IP address. A mechanism was needed to artificially support the use of broadcast where multicast E.164 addresses do not exist; the result is *pseudobroadcasting*. If a multicast address is not available to a destination, pseudobroadcasting can be enabled to broadcast packets to those destinations using a unicast address.

To configure pseudobroadcasting, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# smds static-map` *protocol protocol-address smds-address* `broadcast`	Configures pseudobroadcasting.

For an example of how to configure pseudobroadcasting, see the section "Pseudobroadcasting Example" later in this chapter.

Enabling Fast Switching

SMDS fast switching of IP, IPX, and AppleTalk packets provides faster packet transfer on serial links with speeds above 56 kbps. Use fast switching if you use high-speed, packet-switched, datagram-based WAN technologies such as Frame Relay offered by service providers.

By default, SMDS fast switching is enabled.

To re-enable fast switching, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface` *type number*	Defines and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation smds`	Sets SMDS encapsulation.
Step 3	`Router(config-if)# ip route-cache`	Enables the interface for IP fast switching.
Step 4	`Router(config-if)# ipx route-cache`	Enables the interface for IPX fast switching.
Step 5	`Router(config-if)# appletalk route-cache`	Enables the interface for AppleTalk fast switching.

Monitoring the SMDS Connection

To monitor the SMDS connection, use one or more of the following commands in EXEC mode:

Command	Purpose
Router# **show arp**	Monitors ARP activity.
Router# **show smds addresses**	Displays the individual addresses and the interface with which they are associated.
Router# **show smds map**	Displays all SMDS addresses that are mapped to higher-level protocol addresses.
Router# **show smds traffic**	Displays packet traffic activity.

SMDS Configuration Examples

The following section provides typical configuration file examples you can use as models for your network configurations:

- Typical Multiprotocol Configuration Example
- Remote Peer on the Same Network Example
- IPX Dynamic Address Mapping Example
- AppleTalk Configuration Examples
- Multiple Logical IP Subnetworks over SMDS Example
- Pseudobroadcasting Example

Typical Multiprotocol Configuration Example

The following example is a typical interface configured for IP, DECnet, ISO CLNS, Novell IPX, XNS, and AppleTalk. DECnet needs to be configured globally and at the interface level.

```
interface serial 4
 ip address 1.1.1.2 255.0.0.0
 decnet cost 4
 appletalk address 92.1
 appletalk zone smds
 clns router igrp FOO
 ipx net 1a
 xns net 17
 encapsulation SMDS
! SMDS configuration follows
 smds address c120.1580.4721
 smds static-map APPLETALK 92.2 c120.1580.4592
 smds static-map APPLETALK 92.3 c120.1580.4593
 smds static-map APPLETALK 92.4 c120.1580.4594
 smds static-map NOVELL 1a.0c00.0102.23ca c120.1580.4792
 smds static-map XNS 17.0c00.0102.23ca c120.1580.4792
 smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
 smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
 smds multicast NOVELL e180.0999.9999
 smds multicast XNS e180.0999.9999
 smds multicast ARP e180.0999.9999
 smds multicast IP e180.0999.9999
 smds multicast APPLETALK e180.0999.9999
 smds multicast AARP e180.0999.9999
 smds multicast CLNS_IS e180.0999.9990
```

```
 smds multicast CLNS_ES e180.0999.9990
 smds multicast DECNET_ROUTER e180.0999.9992
 smds multicast DECNET_NODE e180.0999.9992
 smds multicast DECNET e180.0999.9992
 smds enable-arp
```

Remote Peer on the Same Network Example

The following example illustrates a remote peer on the same SMDS network. DECnet needs to be configured globally and at the interface level.

```
interface serial 0
 ip address 1.1.1.1 255.0.0.0
 decnet cost 4
 appletalk address 92.2
 appletalk zone smds
 clns router igrp FOO
 ipx net 1a
 xns net 17
 encapsulation SMDS
! SMDS configuration follows
 smds address c120.1580.4792
 smds static-map APPLETALK 92.1 c120.1580.4721
 smds static-map APPLETALK 92.3 c120.1580.4593
 smds static-map APPLETALK 92.4 c120.1580.4594
 smds static-map NOVELL 1a.0c00.0102.23cb c120.1580.4721
 smds static-map XNS 17.0c00.0102.23cb c120.1580.4721
 smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
 smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
 smds multicast NOVELL e180.0999.9999
 smds multicast XNS e180.0999.9999
 smds multicast IP e180.0999.9999
 smds multicast APPLETALK e180.0999.9999
 smds multicast AARP e180.0999.9999
 smds multicast CLNS_IS e180.0999.9990
 smds multicast CLNS_ES e180.0999.9990
 smds multicast DECNET_ROUTER e180.0999.9992
 smds multicast DECNET_NODE e180.0999.9992
 smds multicast DECNET e180.0999.9992
 smds enable-arp
```

IPX Dynamic Address Mapping Example

The following example enables dynamic address mapping for IPX on interface serial 0 and sets the time to live (TTL) to 14 minutes.

```
interface serial 0
 encapsulation smds
 smds address c141.5797.1313
 smds multicast ipx e180.0999.9999
 smds glean ipx 14
```

AppleTalk Configuration Examples

The following two sections provide basic examples of configuration for an extended AppleTalk network and for a nonextended AppleTalk network. For more information on AppleTalk commands, refer to the *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Extended AppleTalk Network Example

If all AppleTalk routers on the SMDS cloud are running Cisco IOS Release 10.3(4) or later releases, you can use an AppleTalk extended network. To do so, use the **appletalk cable-range** interface command.

When SMDS is configured for an extended AppleTalk network, SMDS static maps are not required and not used. Dynamic AARP is supported on the multicast channel.

```
interface Serial0
 ip address 192.168.200.1 255.255.255.0
 encapsulation smds
 appletalk cable-range 10-10
 appletalk zone SMDS
 smds address c151.0988.1923
 smds static-map ip 192.168.200.2 c151.0988.8770
 smds multicast APPLETALK e151.0988.2232
 smds multicast AARP e151.0988.2232
 smds multicast IP e151.0988.2232
 smds multicast ARP e151.0988.2232
 smds enable-arp
```

Nonextended Appletalk Network Example

The following example configures SMDS for a nonextended AppleTalk network. When SMDS is configured for a nonextended AppleTalk network, SMDS static maps are required and the **appletalk address** command is used. Dynamic AppleTalk Address Resolution Protocol (AARP) is not supported on the multicast channel.

```
interface Serial0
 ip address 192.168.200.1 255.255.255.0
 encapsulation smds
 appletalk address 10.1
 appletalk zone SMDS
 smds address c151.0988.1923
 smds static-map ip 192.168.200.2 c151.0988.8770
 smds static-map appletalk 10.2 c151.0988.8770
 smds multicast APPLETALK e151.0988.2232
 smds multicast IP e151.0988.2232
 smds multicast ARP e151.0988.2232
 smds enable-arp
```

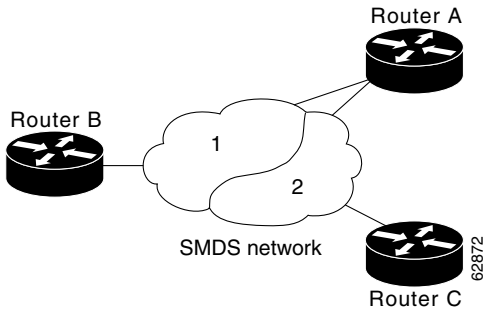
Multiple Logical IP Subnetworks over SMDS Example

In the following example, routers A, B, and C are connected to an SMDS cloud by means of two logical subnetworks labeled 1 and 2, as shown in Figure 2.

Router A recognizes two IP networks and can communicate with Routers B and C directly. Router B can communicate with router A directly, and with router C through router A. Router C can communicate with router A directly and with router B through router A.

Notice that a packet destined to router B from router C must make two hops on the cloud through the same interface on router A. Notice also that this configuration is nonstandard. This issue was considered when the multiple logical IP subnetworks proposal was made, and was deemed not to be critical.

Figure 2 Multiple Logical IP Subnetworks Configuration



The following example shows all routers as Cisco 7200 routers, but they can be other platforms.

Configuration for Router A

```
interface serial 2/0
 encapsulation smds
!
interface serial 2/0.1 multipoint
 smds addr c111.3333.3333
 ip address 2.2.2.1 255.0.0.0
 smds multicast ip e122.2222.2222
 smds enable-arp
 smds multicast ARP e122.2222.2222
```

Configuration for Router B

```
interface serial 4/0
 encapsulation smds
 smds address c111.2222.2222
 ip address 1.1.1.3 255.0.0.0
 smds multicast ip e180.0999.9999
 smds enable-arp
```

Configuration for Router C

```
interface serial 1/0
 encapsulation smds
 smds address c111.4444.4444
 ip address 2.2.2.2 255.0.0.0
 smds multicast ip e122.2222.2222
 smds enable-arp
```

Pseudobroadcasting Example

In the following example, an ARP broadcast from router A is sent to multicast address E180.0999.9999.FFFF to router B and to unicast address C120.1234.5678.FFFF to router C. The reply from router C uses the unicast address C120.1111.2222.FFFF for the return reply if it is the target of the ARP request. IGRP broadcast updates follow the same rules.

Configuration for Router A

```
interface s 0
 encapsulation smds
 smds address c120.1111.2222
 ip address 172.20.1.30 255.255.255.0
 smds multicast ip e180.0999.9999
```

```
 smds static-map ip 172.20.1.10 c120.1234.5678 broadcast
 smds enable-arp
```

Configuration for Router B

```
interface s 4
 smds address c120.9999.8888
 ip address 172.20.1.20
 smds multicast ip e180.0999.9999
 smds enable-arp
```

Configuration for Router C

```
interface serial 2
 smds address c120.1234.5678
 ip address 172.20.1.10
 smds static-map ip 172.20.1.30 c120.1111.2222 broadcast
 smds enable-arp
```

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

L2VPN Pseudowire Redundancy

First Published: April 20, 2005
Last Updated: November 20, 2009

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.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for L2VPN Pseudowire Redundancy" section on page 14.

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

Contents

- Prerequisites for L2VPN Pseudowire Redundancy, page 2
- Restrictions for L2VPN Pseudowire Redundancy, page 2
- Information About L2VPN Pseudowire Redundancy, page 3
- How to Configure L2VPN Pseudowire Redundancy, page 5
- Configuration Examples for L2VPN Pseudowire Redundancy, page 10
- Additional References, page 12
- Command Reference, page 13
- Feature Information for L2VPN Pseudowire Redundancy, page 14

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for L2VPN Pseudowire Redundancy

- This feature module requires that you understand how to configure basic L2 virtual private networks (VPNs). You can find that information in the following documents:
  - *Any Transport over MPLS*
  - *L2 VPN Interworking*
- 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

General Restrictions

- The primary and backup pseudowires must run the same type of transport service. The primary and backup pseudowires must be configured with AToM.
- Only static, on-box provisioning is supported.
- If you use L2VPN Pseudowire Redundancy with L2VPN Interworking, the interworking method must be the same for the primary and backup pseudowires.
- Setting the experimental (EXP) bit on the Multiprotocol Label Switching (MPLS) pseudowire is supported.
- Different pseudowire encapsulation types on the MPLS pseudowire are not supported.
- 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.

Restrictions for Layer 2 Tunnel Protocol Version 3 (L2TPv3) Xconnect Configurations

- Interworking is not supported.
- Local switching backup by pseudowire redundancy is not supported.
- PPP, HDLC, and Frame-Relay attachment circuit (AC) types of L2TPv3 pseudowire redundancy are not supported.
- For the edge interface, only the Cisco 7600 series SPA Interface Processor-400 (SIP-400) linecard with the following shared port adapters (SPAs) is supported:

  Cisco 2-Port Gigabit Ethernet Shared Port Adapter (SPA-2X1GE)
  Cisco 2-Port Gigabit Ethernet Shared Port Adapter, Version 2 (SPA-2X1GE-V2)
  Cisco 5-Port Gigabit Ethernet Shared Port Adapter, Version 2 (SPA-5X1GE-V2)
  Cisco 10-Port Gigabit Ethernet Shared Port Adapter, Version 2 (SPA-10X1GE-V2)

Cisco 2-Port OC3c/STM1c ATM Shared Port Adapter (SPA-2XOC3-ATM)
Cisco 4-Port OC3c/STM1c ATM Shared Port Adapter (SPA-4XOC3-ATM)
Cisco 1-Port OC12c/STM4c ATM Shared Port Adapter (SPA-1XOC12-ATM)
Cisco 1-Port OC-48c/STM-16 ATM Shared Port Adapter (SPA-1XOC48-ATM)

Information About L2VPN Pseudowire Redundancy

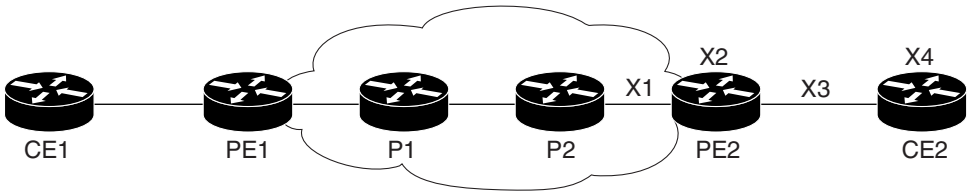
Make sure that you understand the following concept before configuring the L2VPN Pseudowire Redundancy feature:

- Introduction to L2VPN Pseudowire Redundancy, page 3

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. Figure 1 shows those parts of the network that are vulnerable to an interruption in service.

Figure 1 Points of Potential Failure in an L2VPN Network



The L2VPN Pseudowire Redundancy feature provides the ability to ensure that the CE2 router in Figure 1 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, which are shown in Figure 2, Figure 3, and Figure 4.

Figure 2 shows a network with redundant pseudowires and redundant attachment circuits.

Figure 2 L2VPN Network with Redundant PWs and Attachment Circuits

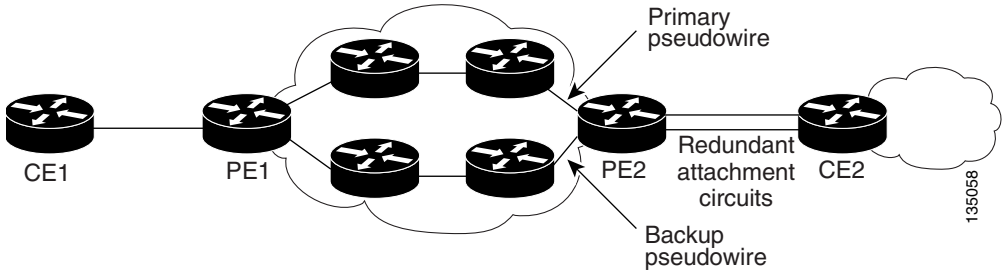


Figure 3 shows a network with redundant pseudowires, attachment circuits, and CE routers.

Figure 3 L2VPN Network with Redundant PWs, Attachment Circuits, and CE Routers

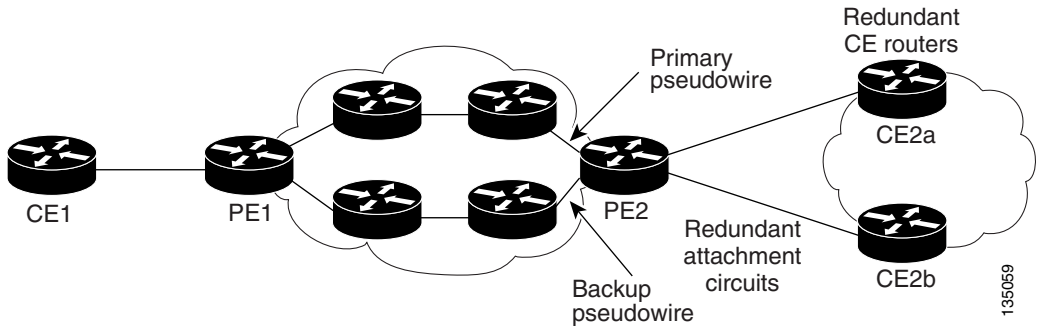
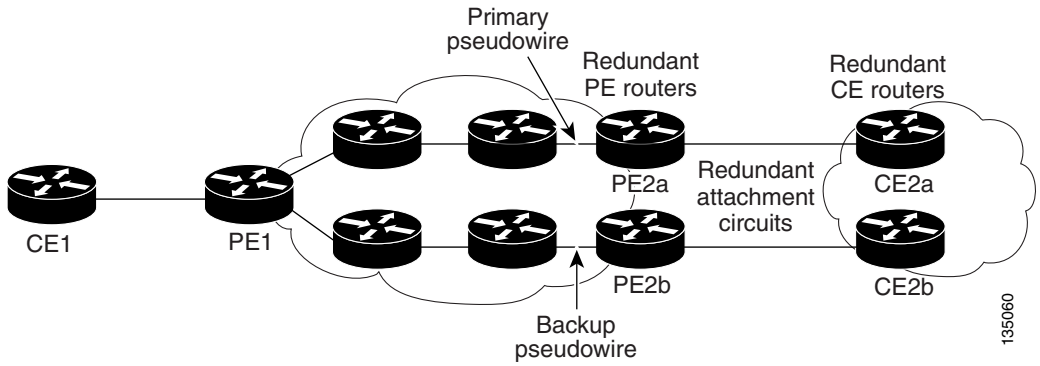


Figure 4 shows a network with redundant pseudowires, attachment circuits, CE routers, and PE routers.

Figure 4 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.

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.

The following sections explain how to configure the L2VPN Pseudowire Redundancy feature:

- Configuring the Pseudowire, page 5 (required)
- Configuring L2VPN Pseudowire Redundancy, page 6 (required)
- Forcing a Manual Switchover to the Backup Pseudowire VC, page 8 (optional)
- Verifying the L2VPN Pseudowire Redundancy Configuration, page 8 (optional)

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

	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 L2VPN Pseudowire Redundancy

Use the following steps to configure the L2VPN Pseudowire Redundancy feature.

Prerequisites

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** *enable-delay* {*disable-delay* | **never**}

DETAILED STEPS

	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. 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. 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	**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. The syntax for this command is the same as for all other Layer 2 transports. Enters xconnect configuration mode.
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 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** *enable-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 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 will move to the fully active state when the command is entered.

SUMMARY STEPS

1. **enable**
2. **xconnect backup force-switchover interface** {*interface-info* | **peer** *ip-address vcid*}

DETAILED STEPS

	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

Use the following commands 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

Step 1 **show mpls l2transport 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:

```
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:

```
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:

```
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:

```
Router(config)# xconnect logging redundancy
```

When this command is configured, the following messages will be generated during switchover events:

Activating the primary member:

```
00:01:07: %XCONNECT-5-REDUNDANCY: Activating primary member 10.55.55.2:1000
```

Activating the backup member:

```
00:01:05: %XCONNECT-5-REDUNDANCY: Activating secondary member 10.55.55.3:1001
```

Configuration Examples for L2VPN Pseudowire Redundancy

The following sections show the L2VPN Pseudowire Redundancy feature examples. These configuration examples show how the L2VPN Pseudowire Redundancy feature can be configured with the AToM (like-to-like), L2VPN Interworking, and Layer 2 Local Switching features.

- L2VPN Pseudowire Redundancy and AToM (Like to Like): Examples, page 10
- L2VPN Pseudowire Redundancy and L2VPN Interworking: Examples, page 10
- L2VPN Pseudowire Redundancy with Layer 2 Local Switching: Examples, page 11

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
```

L2VPN Pseudowire Redundancy and AToM (Like to Like): Examples

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
```

L2VPN Pseudowire Redundancy and L2VPN Interworking: Examples

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
```

L2VPN Pseudowire Redundancy with Layer 2 Local Switching: Examples

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
```

Additional References

The following sections provide references related to the L2VPN Pseudowire Redundancy feature.

Related Documents

Related Topic	Document Title
Any Transport over MPLS	*Any Transport over MPLS*
High Availability for AToM	*AToM Graceful Restart*
L2VPN Interworking	*L2VPN Interworking*
Layer 2 local switching	*Layer 2 Local Switching*
PWE3 MIB	*Pseudowire Emulation Edge-to-Edge MIBs for Ethernet and Frame Relay Services*
Packet sequencing	*Any Transport over MPLS (AToM) Sequencing Support*

Standards

Standards	Title
None	—

MIBs

MIBs	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

RFCs	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

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Multiprotocol Label Switching Command Reference* at http://www.cisco.com/en/US/docs/ios/mpls/command/reference/mp_book.html. For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases*, at http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html.

- **backup delay (L2VPN local switching)**
- **backup peer**
- **show xconnect**
- **xconnect backup force-switchover**
- **xconnect logging redundancy**

Feature Information for L2VPN Pseudowire Redundancy

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for L2VPN Pseudowire Redundancy

Feature Name	Releases	Feature Information
L2VPN Pseudowire Redundancy	12.0(31)S 12.2(28)SB 12.4(11)T 12.2(33)SRB 12.2(22)SXI	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 Release 12.0(31)S, the L2VPN Pseudowire Redundancy feature was introduced for Any Transport over MPLS (AToM) on the Cisco 12000 series routers. This feature was integrated into Cisco IOS Release 12.2(28)SB. This feature was integrated into Cisco IOS Release 12.4(11)T. This feature was integrated into Cisco IOS Release 12.2(33)SRB. This feature was integrated into Cisco IOS Release 12.2(33)SXI. The following sections provide information about this feature: • Introduction to L2VPN Pseudowire Redundancy, page 3 • Configuring the Pseudowire, page 5 • Configuring L2VPN Pseudowire Redundancy, page 6 • Forcing a Manual Switchover to the Backup Pseudowire VC, page 8 • Verifying the L2VPN Pseudowire Redundancy Configuration, page 8 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 Redundancy for L2TPv3	12.2(33)SRE	This feature provides L2VPN pseudowire redundancy for L2TPv3 xconnect configurations. In Cisco IOS Release 12.2(33)SRE, this feature was implemented on the Cisco 7600 series routers.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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

© 2005–2009 Cisco Systems, Inc. All rights reserved.

L2VPN Interworking

First Published: August 26, 2003
Last Updated:November 20, 2009

Layer 2 Virtual Private Network (L2VPN) Interworking allows you to connect disparate attachment circuits. This feature module explains how to configure the following L2VPN Interworking features:

- Ethernet/VLAN to ATM AAL5 Interworking
- Ethernet/VLAN to Frame Relay Interworking
- Ethernet/VLAN to PPP Interworking
- Ethernet to VLAN Interworking
- Frame Relay to ATM AAL5 Interworking
- Frame Relay to PPP Interworking
- Ethernet/VLAN to ATM virtual channel identifier (VPI) and virtual channel identifier (VCI) Interworking
- L2VPN Interworking: VLAN Enable/Disable Option for AToM

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for L2VPN Interworking" section on page 32.

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

Contents

- Prerequisites for L2VPN Interworking, page 2
- Restrictions for L2VPN Interworking, page 2
- Information About L2VPN Interworking, page 11
- How to Configure L2VPN Interworking, page 15
- Configuration Examples for L2VPN Interworking, page 21
- Additional References, page 29
- Feature Information for L2VPN Interworking, page 32

Prerequisites for L2VPN Interworking

Before you configure L2VPN Interworking on a router:

- You must enable Cisco Express Forwarding.
- On the Cisco 12000 series Internet router, before you configure Layer 2 Tunnel Protocol version 3 (L2TPv3) for L2VPN Interworking on an IP Services Engine (ISE/Engine 3) or Engine 5 interface, you must also enable the L2VPN feature bundle on the line card.

  To enable the feature bundle, enter the **hw-module slot np mode feature** command in global configuration mode as follows:

```
Router# configure terminal
Router(config)# hw-module slot slot-number np mode feature
```

Restrictions for L2VPN Interworking

The following sections list the L2VPN Interworking restrictions:

- General Restrictions, page 2
- Cisco 7600 Series Routers Restrictions, page 3
- Cisco 12000 Series Router Restrictions, page 5
- ATM AAL5 Interworking Restrictions, page 7
- Ethernet/VLAN Interworking Restrictions, page 8
- L2VPN Interworking: VLAN Enable/Disable Option for AToM Restrictions, page 9
- Frame Relay Interworking Restrictions, page 10
- PPP Interworking Restrictions, page 11

General Restrictions

This section lists general restrictions that apply to L2VPN Interworking. Other restrictions that are platform-specific or device-specific are listed in the following sections.

- The interworking type on one provider edge (PE) router must match the interworking type on the peer PE router.

- The following quality of service (QoS) features are supported with L2VPN Interworking:
  - Static IP type of service (ToS) or Multiprotocol Label Switching (MPLS) experimental bit (EXP) setting in tunnel header
  - IP ToS reflection in tunnel header (Layer 2 Tunnel Protocol Version 3 (L2TPv3) only)
  - Frame Relay policing
  - Frame Relay data-link connection identifier (DLCI)-based congestion management (Cisco 7500/Versatile Interface Processor (VIP))
  - One-to-one mapping of VLAN priority bits to MPLS EXP bits
- Only ATM AAL5 VC mode is supported; ATM VP and port mode are not supported.

Cisco 7600 Series Routers Restrictions

The following line cards are supported on the Cisco 7600 series router. Table 1 shows the line cards that are supported on the WAN (ATM, Frame Relay, or PPP) side of the interworking link. Table 2 shows the line cards that are supported on the Ethernet side of the interworking link. For more details on the Cisco 7600 routers supported shared port adapters and line cards, see the following documents:

- *Cisco 7600 Series Routers Documentation Roadmap*
- *Release Notes for Cisco IOS Release 12.2SR for the Cisco 7600 Series Routers*

Table 1 Cisco 7600 Series Routers: Supported Line Cards for the WAN Side

Interworking Type	Core-Facing Line Cards	Customer-Edge Line Cards
Ethernet (bridged) (ATM and Frame Relay)	Any	EflexWAN SIP-200 SIP-400
IP (routed) (ATM, Frame Relay, and PPP)	Any	EflexWAN SIP-200

Table 2 Cisco 7600 Series Routers: Supported Line Cards for the Ethernet Side

Interworking Type	Ethernet over MPLS Mode	Core-Facing Line Cards	Customer-Edge Line Cards
Ethernet (bridged)	Policy feature card (PFC) based	Any, except optical service module (OSM) and ES40	Catalyst LAN SIP-600
Ethernet (bridged)	Switched virtual interface (SVI) based	EflexWAN ES20 ES+40 SIP-200 SIP-400 SIP-600	Catalyst LAN EflexWAN (with MPB) ES20 ES+40 SIP-200 (with MPB) SIP-400 (with MPB) SIP-600

Table 2 Cisco 7600 Series Routers: Supported Line Cards for the Ethernet Side (continued)

Interworking Type	Ethernet over MPLS Mode	Core-Facing Line Cards	Customer-Edge Line Cards
Ethernet (bridged)	Scalable (with E-MPB)	Any, except OSM	ES20 SIP-600 and SIP-400 with Gigabit Ethernet (GE) SPA
IP (routed)	PFC-based	Catalyst LAN SIP-600 **Note:** PFC-based mode is not supported with routed interworking in Cisco IOS Release 12.2(33)SRD. Use SVI, Scalable, or Ethernet virtual connection (EVC) based Ethernet over MPLS (EoMPLS) instead.	Catalyst LAN SIP-600 **Note:** PFC-based mode is not supported with routed interworking in Cisco IOS Release 12.2(33)SRD. Use SVI, Scalable, or EVC-based EoMPLS instead.
IP (routed)	SVI-based	Any, except Catalyst LAN and OSM.	Catalyst LAN EflexWAN (with MPB) ES20 SIP-200 (with MPB) SIP-400 (with MPB) SIP-600

The following restrictions apply to the Cisco 7600 series routers and L2VPN Interworking:

- OAM Emulation is not required with L2VPN Interworking on the SIP-200, SIP-400, and Flexwan2 line cards.
- Cisco 7600 series routers support the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature starting in Cisco IOS Release 12.2(33)SRE. This feature has the following restrictions:
  - PFC-based EoMPLS is not supported.
  - Scalable and SVI-based EoMPLS are supported with the SIP-400 line card.
- The Cisco 7600 series routers do not support L2VPN Interworking over L2TPv3.
- Cisco 7600 series routers support only the following interworking types:
  - Ethernet/VLAN to Frame Relay (IP and Ethernet modes)
  - Ethernet/VLAN to ATM AAL5SNAP (IP and Ethernet modes)
  - Ethernet/VLAN to PPP (IP only)
  - Ethernet to VLAN Interworking
- Cisco 7600 series routers do not support the following interworking types:
  - Ethernet/VLAN to ATM AAL5MUX
  - Frame Relay to PPP Interworking
  - Frame Relay to ATM AAL5 Interworking

- Both ends of the interworking link must be configured with the same encapsulation and interworking type:
  - If you use Ethernet encapsulation, you must use the Ethernet (bridged) interworking type. If you are not using Ethernet encapsulation, you can use a bridging mechanism, such as routed bridge encapsulation (RBE).
  - If you use an IP encapsulation (such as ATM or Frame Relay), you must use the IP (routed) interworking type. The PE routers negotiate the process for learning and resolving addresses.
  - You must use the same MTU size on the attachment circuits at each end of the pseudowire.
- PFC-based EoMPLS is not supported on ES40 line cards. SVI and EVC/scalable EoMPLS are the alternative options.
- PFC-based EoMPLS is not supported for Routed/IP interworking in Cisco IOS Release 12.2(33)SRD and later releases. The alternative Routed/IP interworking options are SVI and EVC or scalable EoMPLS. However, PFC-based EoMPLS is supported for Ethernet/Bridged interworking and for like-to-like over AToM.

Cisco 12000 Series Router Restrictions

For more information about hardware requirements on the Cisco12000 series routers, see the *Cross-Platform Release Notes for Cisco IOS Release 12.0S.*

For QOS support on the Cisco 12000 series routers, see *Any Transport over MPLS (AToM): Layer 2 QoS (Quality of Service) for the Cisco 12000 Series Router*

Frame Relay to PPP and High-Level Data Link Control Interworking

The Cisco 12000 series Internet router does not support L2VPN Interworking with PPP and high-level data link control (HDLC) transport types in Cisco IOS releases earlier than Cisco IOS Release 12.0(32)S.

In Cisco IOS Release 12.0(32)S and later releases, the Cisco 12000 series Internet router supports L2VPN interworking for Frame Relay over MPLS and PPP and HDLC over MPLS only on the following shared port adapters (SPAs):

- ISE/Engine 3 SPAs:
  - SPA-2XCT3/DS0 (2-port channelized T3 to DS0)
  - SPA-4XCT3/DS0 (4-port channelized T3 to DS0)
- Engine 5 SPAs:
  - SPA-1XCHSTM1/OC-3 (1-port channelized STM-1c/OC-3c to DS0)
  - SPA-8XCHT1/E1 (8-port channelized T1/E1)
  - SPA-2XOC-48-POS/RPR (2-port OC-48/STM16 POS/RPR)
  - SPA-OC-192POS-LR (1-port OC-192/STM64 POS/RPR)
  - SPA-OC-192POS-XFP (1-port OC-192/STM64 POS/RPR)

L2VPN Interworking over L2TPv3

On the Cisco 12000 series Internet router, Ethernet (bridged) interworking is not supported for L2TPv3. Only IP (routed) interworking is supported.

IP (routed) interworking is not supported in an L2TPv3 pseudowire that is configured for data sequencing (using the **sequencing** command).

In Cisco IOS Release 12.0(32)SY and later releases, the Cisco 12000 series Internet router supports L2VPN Interworking over L2TPv3 tunnels in IP mode on ISE and Engine 5 line cards as follows:

- On an ISE interface configured for L2TPv3 tunneling, the following Layer 2 encapsulations are supported:
  - ATM adaptation layer type-5 (AAL5)
  - Ethernet
  - 802.1q (VLAN)
  - Frame Relay DLCI
- On an Engine 5 interface configured for L2TPv3 tunneling, the following Layer 2 encapsulations are supported:
  - Ethernet
  - 802.1q (VLAN)
  - Frame Relay DLCI

For more information, refer to *Layer 2 Tunnel Protocol Version 3*.

The only frame format supported for L2TPv3 interworking on Engine 5 Ethernet SPAs is Ethernet Version 2 (also known as Ethernet II) with the Ether type 0x0800 value set as Internet Protocol Payload and (optionally) 802.1q VLAN. Ethernet packets with other Ethernet frame formats are dropped.

Remote Ethernet Port Shutdown Support

The Cisco Remote Ethernet Port Shutdown feature (which minimizes potential data loss after a remote link failure) is supported only on the following Engine 5 Ethernet SPAs:

- SPA-8XFE (8-port Fast Ethernet)
- SPA-2X1GE (2-port Gigabit Ethernet)
- SPA-5X1GE (5-port Gigabit Ethernet)
- SPA-10X1GE (10-port Gigabit Ethernet)
- SPA-1X10GE (1-port 10-Gigabit Ethernet)

For more information about this feature, refer to *Any Transport over MPLS (AToM): Remote Ethernet Port Shutdown.*

L2VPN Any-to-Any Interworking on Engine 5 Line Cards

Table 3 shows the different combinations of transport types supported for L2VPN interworking on Engine 3 and Engine 5 SPA interfaces connected through an attachment circuit over MPLS or L2TPv3.

Table 3 Engine 3 and Engine 5 Line Cards/SPAs Supported for L2VPN Interworking

Attachment Circuit 1 (AC1)	Attachment Circuit 2 (AC2)	Interworking Mode	AC1 Engine Type and Line Card/SPA	AC2 Engine Type and Line Card/SPA
Frame Relay	Frame Relay	IP	Engine 5 POS and channelized	Engine 3 ATM line cards
Frame Relay	ATM	Ethernet	Engine 5 POS and channelized	Engine 3 ATM line cards
Frame Relay	ATM	IP	Engine 5 POS and channelized	Engine 3 ATM line cards

Table 3 Engine 3 and Engine 5 Line Cards/SPAs Supported for L2VPN Interworking

Attachment Circuit 1 (AC1)	Attachment Circuit 2 (AC2)	Interworking Mode	AC1 Engine Type and Line Card/SPA	AC2 Engine Type and Line Card/SPA
Frame Relay	Ethernet	Ethernet	Engine 5 POS and channelized	Engine 5 Gigabit Ethernet
Frame Relay	Ethernet	IP	Engine 5 POS and channelized	Engine 5 Gigabit Ethernet
Frame Relay	VLAN	Ethernet	Engine 5 POS and channelized	Engine 5 Gigabit Ethernet
Frame Relay	VLAN	IP	Engine 5 POS and channelized	Engine 5 Gigabit Ethernet
Ethernet	Ethernet	Ethernet	Engine 5 Gigabit Ethernet	Engine 5 Gigabit Ethernet
Ethernet	Ethernet	IP	Engine 5 Gigabit Ethernet	Engine 5 Gigabit Ethernet
Ethernet	VLAN	Ethernet	Engine 5 Gigabit Ethernet	Engine 5 Gigabit Ethernet
Ethernet	VLAN	IP	Engine 5 Gigabit Ethernet	Engine 5 Gigabit Ethernet
ATM	Ethernet	Ethernet	Engine 3 ATM line cards	Engine 5 Gigabit Ethernet
ATM	Ethernet	IP	Engine 3 ATM line cards	Engine 5 Gigabit Ethernet

On the Cisco 12000 series Engine 3 line card, Network Layer Protocol ID (NLPID) encapsulation is not supported in routed mode; and neither NLPID nor AAL5MUX is supported in bridged mode.

- On the Cisco 12000 series Internet router, Ethernet (bridged) interworking is not supported for L2TPv3.

  In an L2VPN Interworking configuration, after you configure L2TPv3 tunnel encapsulation for a pseudowire using the **encapsulation l2tpv3** command, you cannot enter the **interworking ethernet** command.

- On Ethernet SPAs on the Cisco 12000 series Internet router, the only frame format supported for L2TPv3 interworking is Ethernet Version 2 (also known as Ethernet II) with the Ether type 0x0800 value set as Internet Protocol Payload and [optionally] 802.1q VLAN.

  Ethernet packets with other Ethernet frame formats are dropped.

ATM AAL5 Interworking Restrictions

The following restrictions apply to ATM AAL5 Interworking:

- Switched virtual circuits (SVCs) are not supported.
- Inverse Address Resolution Protocol (ARP) is not supported with IP interworking.
- Customer edge (CE) routers must use point-to-point subinterfaces or static maps.

- Both AAL5MUX and AAL5SNAP encapsulation are supported. In the case of AAL5MUX, no translation is needed.
- In the Ethernet end-to-end over ATM scenario, the following translations are supported:
  - Ethernet without LAN frame check sequence (FCS) (AAAA030080C200070000)
  - Spanning tree (AAAA030080c2000E)

  Everything else is dropped.
- In the IP over ATM scenario, the IPv4 (AAAA030000000800) translation is supported. Everything else is dropped.
- Operation, Administration, and Management (OAM) emulation for L2VPN Interworking is the same as like-to-like. The end-to-end F5 loopback cells are looped back on the PE router. When the pseudowire is down, an F5 end-to-end segment Alarm Indication Signal (AIS)/Remote Defect Identification (RDI) is sent from the PE router to the CE router.
- Interim Local Management Interface (ILMI) can manage virtual circuits (VCs) and permanent virtual circuits (PVCs).
- To enable ILMI management, configure ILMI PVC 0/16 on the PE router's ATM interface. If a PVC is provisioned or deleted, an ilmiVCCChange trap is sent to the CE router.
- Only the user side of the User-Network Interface (UNI) is supported; the network side of the UNI is not supported.

Ethernet/VLAN Interworking Restrictions

The following restrictions apply to Ethernet/VLAN interworking:

- When you configure VLAN to Ethernet interworking, VLAN to Frame Relay (routed), or ATM using Ethernet (bridged) interworking, the PE router on the Ethernet side that receives a VLAN tagged frame from the CE router removes the VLAN tag. In the reverse direction, the PE router adds the VLAN tag to the frame before sending the frame to the CE router.

  (If you enable the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature with the **interworking vlan** command, VLAN ID is included as part of the Ethernet frame. See the "VLAN Interworking" section on page 13 for more information.)
- In bridged interworking from VLAN to Frame Relay, the Frame Relay PE router does not strip off VLAN tags from the Ethernet traffic it receives.
- The Cisco 10720 Internet router supports Ethernet to VLAN Interworking Ethernet only over L2TPv3.
- Ethernet interworking for a raw Ethernet port or a VLAN trunk is not supported. Traffic streams are not kept separate when traffic is sent between transport types.
- In routed mode, only one CE router can be attached to an Ethernet PE router.
- There must be a one-to-one relationship between an attachment circuit and the pseudowire. Point-to-multipoint or multipoint-to-point configurations are not supported.
- Configure routing protocols for point-to-point operation on the CE routers when configuring an Ethernet to non-Ethernet setup.
- In the IP interworking mode, the IPv4 (0800) translation is supported. The PE router captures ARP (0806) packets and responds with its own MAC address (proxy ARP). Everything else is dropped.

- The Ethernet or VLAN must contain only two IP devices: PE router and CE router. The PE router performs proxy ARP and responds to all ARP requests it receives. Therefore, only one CE and one PE router should be on the Ethernet or VLAN segment.
- If the CE routers are doing static routing, you can perform the following tasks:
  - The PE router needs to learn the MAC address of the CE router to correctly forward traffic to it. The Ethernet PE router sends an Internet Control Message Protocol (ICMP) Router discovery protocol (RDP) solicitation message with the source IP address as zero. The Ethernet CE router responds to this solicitation message. To configure the Cisco CE router's Ethernet or VLAN interface to respond to the ICMP RDP solicitation message, issue the **ip irdp** command in interface configuration mode. If you do not configure the CE router, traffic is dropped until the CE router sends traffic toward the PE router.
  - To disable the CE routers from running the router discovery protocol, issue the **ip irdp maxadvertinterval 0** command in interface mode.
- This restriction applies if you configure interworking between Ethernet and VLAN with Catalyst switches as the CE routers. The spanning tree protocol is supported for Ethernet interworking. Ethernet interworking between an Ethernet port and a VLAN supports spanning tree protocol only on VLAN 1. Configure VLAN 1 as a nonnative VLAN.
- When you change the interworking configuration on an Ethernet PE router, clear the ARP entry on the adjacent CE router so that it can learn the new MAC address. Otherwise, you might experience traffic drops.

L2VPN Interworking: VLAN Enable/Disable Option for AToM Restrictions

The following restrictions apply to the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature, which allows the VLAN ID to be included as part of the Ethernet frame:

- The L2VPN Interworking: VLAN Enable/Disable Option for AToM feature is supported on the following releases:
  - Cisco IOS release 12.2(52)SE for the Cisco Catalyst 3750 Metro switches
  - Cisco IOS Release 12.2(33)SRE for the Cisco 7600 series routers
- L2VPN Interworking: VLAN Enable/Disable Option for AToM is not supported with L2TPv3. You can configure the featue only with AToM.
- If the interface on the PE router is a VLAN interface, it is not necessary to specify the **interworking vlan** command on that PE router.
- The L2VPN Interworking: VLAN Enable/Disable Option for AToM feature works only with the following attachment circuit combinations:
  - Ethernet to Ethernet
  - Ethernet to VLAN
  - VLAN to VLAN
- If you specify an interworking type on a PE router, that interworking type must be enforced. The interworking type must match on both PE routers. Otherwise, the VC may be in an incompatible state and remain in the down state. If the attachment circuit (AC) is VLAN, the PE router can negotiate (autosense) the VC type using Label Distribution Protocol (LDP).

  For example, both PE1 and PE2 use Ethernet interfaces, and VLAN interworking is specified on PE1 only. PE2 is not configured with an interworking type and cannot autosense the interworking type. The result is an incompatible state where the VC remains in the down state.

On the other hand, if PE1 uses an Ethernet interface and VLAN interworking is enabled (which will enforce VLAN as the VC type), and PE2 uses a VLAN interface and interworking is not enabled (which causes PE2 to use Ethernet as its default VC type), PE2 can autosense and negotiate the interworking type and select VLAN as the VC type.

Table 4 summarizes shows the AC types, interworking options, and VC types after negotiation.

Table 4 Negotiating Ethernet and VLAN Interworking Types

PE1 AC Type	Interworking Option	PE2 AC Type	Interworking Option	VC Type after Negotiation
Ethernet	none	Ethernet	none	Ethernet
Vlan	none	Ethernet	none	Ethernet
Ethernet	none	Vlan	none	Ethernet
Vlan	none	Vlan	none	Ethernet
Ethernet	Vlan	Ethernet	none	Incompatible
Vlan	Vlan	Ethernet	none	Incompatible
Ethernet	Vlan	Vlan	none	Vlan
Vlan	Vlan	Vlan	none	Vlan
Ethernet	none	Ethernet	Vlan	Incompatible
Vlan	none	Ethernet	Vlan	Vlan
Ethernet	none	Vlan	Vlan	Incompatible
Vlan	none	Vlan	Vlan	Vlan
Ethernet	Vlan	Ethernet	Vlan	Vlan
Vlan	Vlan	Ethernet	Vlan	Vlan
Ethernet	Vlan	Vlan	Vlan	Vlan
Vlan	Vlan	Vlan	Vlan	Vlan

Frame Relay Interworking Restrictions

The following restrictions apply to Frame Relay interworking:

- The attachment circuit maximum transmission unit (MTU) sizes must match when you connect them over MPLS. By default, the MTU size associated with a Frame Relay DLCI is the interface MTU. This may cause problems, for example, when connecting some DLCIs on a PoS interface (with a default MTU of 4470 bytes) to Ethernet or VLAN (with a default MTU of 1500 bytes) and other DLCIs on the same PoS interface to ATM (with a default MTU of 4470 bytes). To avoid reducing all the interface MTUs to the lowest common denominator (1500 bytes in this case), you can specify the MTU for individual DLCIs using the **mtu** command.
- Only DLCI mode is supported. Port mode is not supported.
- Configure Frame Relay switching to use DCE or Network-to-Network Interface (NNI). DTE mode does not report status in the Local Management Interface (LMI) process. If a Frame Relay over MPLS circuit goes down and the PE router is in DTE mode, the CE router is never informed of the disabled circuit. You must configure the **frame-relay switching** command in global configuration mode in order to configure DCE or NNI.

- Frame Relay policing is non-distributed on the Cisco 7500 series routers. If you enable Frame Relay policing, traffic is sent to the route switch processor for processing.
- Inverse ARP is not supported with IP interworking. CE routers must use point-to-point subinterfaces or static maps.
- The PE router automatically supports translation of both the Cisco encapsulations and the Internet Engineering Task Force (IETF) encapsulations that come from the CE, but translates only to IETF when sending to the CE router. This is not a problem for the Cisco CE router, because it can handle IETF encapsulation on receipt even if it is configured to send Cisco encapsulation.
- With Ethernet interworking, the following translations are supported:
  - Ethernet without LAN FCS (0300800080C20007 or 6558)
  - Spanning tree (0300800080C2000E or 4242)

  All other translations are dropped.
- With IP interworking, the IPv4 (03CC or 0800) translation is supported. All other translations are dropped.
- PVC status signaling works the same way as in like-to-like case. The PE router reports the PVC status to the CE router, based on the availability of the pseudowire. PVC status detected by the PE router will also be reflected into the pseudowire. LMI to OAM interworking is supported when you connect Frame Relay to ATM.

PPP Interworking Restrictions

The following restrictions apply to PPP interworking:

- There must be a one-to-one relationship between a PPP session and the pseudowire. Multiplexing of multiple PPP sessions over the pseudowire is not supported.
- There must be a one-to-one relationship between a PPP session and a Frame Relay DLCI. Each Frame Relay PVC must have only one PPP session.
- Only IP (IPv4 (0021) interworking is supported. Link Control Protocol (LCP) packets and Internet Protocol Control Protocol (IPCP) packets are terminated at the PE router. Everything else is dropped.
- Proxy IPCP is automatically enabled on the PE router when IP interworking is configured on the pseudowire.
- By default, the PE router assumes that the CE router knows the remote CE router's IP address.
- Password Authentication Protocol (PAP) and Challenge-Handshake Authentication Protocol (CHAP) authentication are supported.

Information About L2VPN Interworking

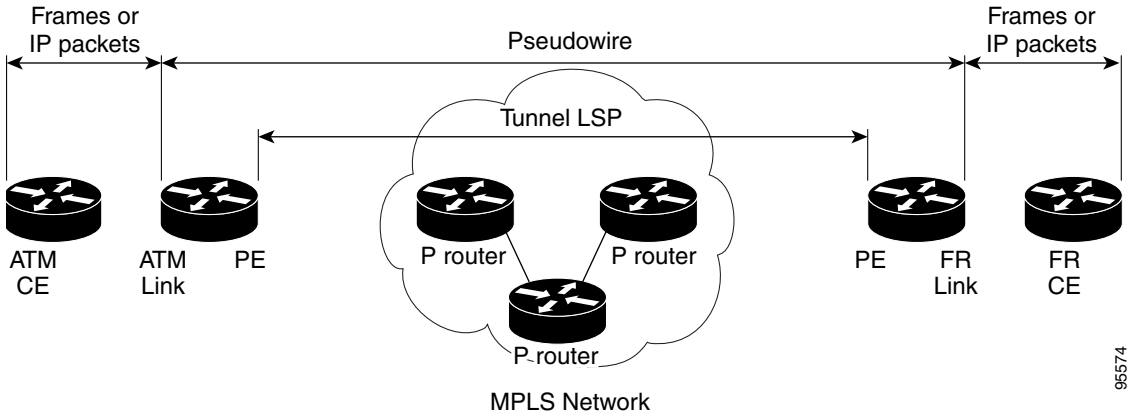
The following sections provide an introduction to L2VPN interworking.

- Overview of L2VPN Interworking, page 12
- L2VPN Interworking Modes, page 12
- L2VPN Interworking: Support Matrix, page 14
- Static IP Addresses for L2VPN Interworking for PPP, page 14

Overview of L2VPN 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. Figure 1 is an example of Layer 2 interworking, where ATM and Frame Relay packets travel over the MPLS cloud.

Figure 1 ATM to Frame Relay Interworking Example



The L2VPN Interworking feature supports Ethernet, 802.1Q (VLAN), Frame Relay, ATM AAL5, and PPP attachment circuits over MPLS and L2TPv3. The features and restrictions for like-to-like functionality also apply to L2VPN Interworking.

L2VPN Interworking Modes

L2VPN Interworking works in either Ethernet (“bridged”) mode, IP (“routed”), or Ethernet VLAN mode. You specify the mode by issuing the **interworking** {**ethernet** | **ip** |**vlan**} command in pseudowire-class configuration mode.

Ethernet (Bridged) Interworking

The **ethernet** keyword causes Ethernet frames to be extracted from the attachment circuit and sent over the pseudowire. Ethernet end-to-end transmission is assumed. Attachment circuit frames that are not Ethernet are dropped. In the case of VLAN, the VLAN tag is removed, leaving an untagged Ethernet frame.

Ethernet Interworking is also called bridged interworking. Ethernet frames are bridged across the pseudowire. The CE routers could be natively bridging Ethernet or could be routing using a bridged encapsulation model, such as Bridge Virtual Interface (BVI) or RBE. The PE routers operate in Ethernet like-to-like mode.

This mode is used to offer the following services:

- LAN services—An example is an enterprise that has several sites, where some sites have Ethernet connectivity to the service provider (SP) network and others have ATM connectivity. The enterprise wants LAN connectivity to all its sites. In this case, traffic from the Ethernet or VLAN of one site can be sent through the IP/MPLS network and encapsulated as bridged traffic over an ATM VC of another site.
- Connectivity services—An example is an enterprise that has different sites that are running an Internal Gateway Protocol (IGP) routing protocol, which has incompatible procedures on broadcast and nonbroadcast links. The enterprise has several sites that are running an IGP, such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS), between the sites. In this scenario, some of the procedures (such as route advertisement or designated router) depend on the underlying Layer 2 protocol and are different for a point-to-point ATM connection versus a broadcast Ethernet connection. Therefore, the bridged encapsulation over ATM can be used to achieve homogenous Ethernet connectivity between the CE routers running the IGP.

IP (Routed) Interworking

The **ip** keyword causes IP packets to be extracted from the attachment circuit and sent over the pseudowire. Attachment circuit frames that do not contain IPv4 packets are dropped.

IP Interworking is also called routed interworking. The CE routers encapsulate IP on the link between the CE and PE routers. A new VC type is used to signal the IP pseudowire in MPLS and L2TPv3. Translation between the Layer 2 and IP encapsulations across the pseudowire is required. Special consideration needs to be given to address resolution and routing protocol operation, because these are handled differently on different Layer 2 encapsulations.

This mode is used to provide IP connectivity between sites, regardless of the Layer 2 connectivity to these sites. It is different from a Layer 3 VPN because it is point-to-point in nature and the service provider does not maintain any customer routing information.

Address resolution is encapsulation dependent:

- Ethernet uses ARP
- Frame Relay and ATM use Inverse ARP
- PPP uses IPCP

Therefore, address resolution must be terminated on the PE router. End-to-end address resolution is not supported. Routing protocols operate differently over broadcast and point-to-point media. For Ethernet, the CE routers must either use static routing or configure the routing protocols to treat the Ethernet side as a point-to-point network.

VLAN Interworking

The **vlan** keyword allows the VLAN ID to be included as part of the Ethernet frame. In Cisco IOS Release 12.2(52)SE, you can configure Catalyst 3750 Metro switches to use Ethernet VLAN for Ethernet (bridged) interworking. You can specify the Ethernet VLAN (type 4) by issuing the **interworking vlan** command in pseudowire-class configuration mode. This allows the VLAN ID to be included as part of the Ethernet frame. In releases previous to Cisco IOS Release 12.2(52)SE, the only way to achieve VLAN encapsulation is to ensure the CE router is connected to the PE router through an Ethernet VLAN interface/subinterface.

L2VPN Interworking: Support Matrix

The supported L2VPN Interworking features are listed in Table 5.

Table 5 L2VPN Interworking Supported Features

Feature	MPLS or L2TPv3 Support	IP or Ethernet Support
Ethernet/VLAN to ATM AAL5	MPLS L2TPv3 (12000 series only)	IP Ethernet
Ethernet/VLAN to Frame Relay	MPLS L2TPv3	IP Ethernet
Ethernet/VLAN to PPP	MPLS	IP
Ethernet to VLAN	MPLS L2TPv3	IP Ethernet[1]
L2VPN Interworking: VLAN Enable/Disable Option for AToM	MPLS	Ethernet VLAN
Frame Relay to ATM AAL5	MPLS L2TPv3 (12000 series only)	IP
Frame Relay to Ethernet or VLAN	MPLS L2TPv3	IP Ethernet
Frame Relay to PPP	MPLS L2TPv3	IP

Note: On the Cisco 12000 series Internet router:

- Ethernet (bridged) interworking is not supported for L2TPv3.
- IP (routed) interworking is not supported in an L2TPv3 pseudowire configured for data sequencing (using the **sequencing** command).

1. With the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature, VLAN interworking can also be supported. For more information, see the "VLAN Interworking" section on page 13.

Static IP Addresses for L2VPN Interworking for PPP

If the PE router needs to perform address resolution with the local CE router for PPP, you can configure the remote CE router's IP address on the PE router. Issue the **ppp ipcp address proxy** command with the remote CE router's IP address on the PE router's xconnect PPP interface. The following example shows a sample configuration:

```
pseudowire-class ip-interworking
 encapsulation mpls
 interworking ip

interface Serial2/0
 encapsulation ppp
 xconnect 10.0.0.2 200 pw-class ip-interworking
 ppp ipcp address proxy 10.65.32.14
```

You can also configure the remote CE router's IP address on the local CE router with the **peer default ip address** command if the local CE router performs address resolution.

How to Configure L2VPN Interworking

The following sections explain the tasks you can perform to configure L2VPN Interworking:

- Configuring L2VPN Interworking, page 15 (required)
- Verifying the L2VPN Interworking Configuration, page 16 (optional)
- Configuring L2VPN Interworking: VLAN Enable/Disable Option for AToM, page 19 (optional)

Configuring L2VPN Interworking

L2VPN Interworking allows you to connect disparate attachment circuits. Configuring the L2VPN Interworking feature requires that you add the **interworking** command to the list of commands that make up the pseudowire. The steps for configuring the pseudowire for L2VPN Interworking are included in this section. You use the **interworking** command as part of the overall AToM or L2TPv3 configuration. For specific instructions on configuring AToM or L2TPv3, see the following documents:

- *Layer 2 Tunnel Protocol Version 3*
- *Any Transport over MPLS*

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hw-module slot** *slot-number* **np mode feature**
4. **pseudowire-class** *name*
5. **encapsulation** {**mpls** | **l2tpv3**}
6. **interworking** {**ethernet** | **ip** | **vlan**}

DETAILED STEPS

	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	**hw-module slot** *slot-number* **np mode feature** **Example:** `Router(config)# hw-module slot 3 np mode feature`	(Optional) Enables L2VPN Interworking functionality on the Cisco 12000 series router. **Note** Enter this command only on a Cisco 12000 series Internet router if you use L2TPv3 for L2VPN Interworking on an ISE (Engine 3) or Engine 5 interface. In this case, you must first enable the L2VPN feature bundle on the line card by entering the **hw-module slot** *slot-number* **np mode feature** command.
Step 4	**pseudowire-class** *name* **Example:** `Router(config)# pseudowire-class class1`	Establishes a pseudowire class with a name that you specify and enters pseudowire class configuration mode.
Step 5	**encapsulation** {**mpls** | **l2tpv3**} **Example:** `Router(config-pw)# encapsulation mpls`	Specifies the tunneling encapsulation, which is either **mpls** or **l2tpv3.**
Step 6	**interworking** {**ethernet** | **ip**} | **vlan**} **Example:** `Router(config-pw)# interworking ip`	Specifies the type of pseudowire and the type of traffic that can flow across it. **Note** On the Cisco 12000 series Internet router, Ethernet (bridged) interworking is not supported for L2TPv3. After you configure the L2TPv3 tunnel encapsulation for the pseudowire using the **encapsulation l2tpv3** command, you cannot enter the **interworking ethernet** command.

Verifying the L2VPN Interworking Configuration

To verify the L2VPN Interworking configuration, you can use the following commands.

SUMMARY STEPS

1. **enable**
1. **show l2tun session all**
2. **show arp**
3. **ping**
4. **show l2tun session interworking**
5. **show mpls l2transport vc detail**

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Step 2 **show l2tun session all** (L2TPv3 only)

For L2TPv3, you can verify the L2VPN Interworking configuration using the **show l2tun session all** command on the PE routers.

In the following example, the interworking type is shown in bold.

PE1

```
Router# show l2tun session all

 Session Information Total tunnels 1 sessions 1

Session id 15736 is up, tunnel id 35411
Call serial number is 4035100045
Remote tunnel name is PE2
   Internet address is 10.9.9.9
   Session is L2TP signalled
   Session state is established, time since
change 1d22h
   16 Packets sent, 16 received
   1518 Bytes sent, 1230 received
   Receive packets dropped:
     out-of-order:              0
     total:                     0
   Send packets dropped:
     exceeded session MTU:      0
     total:                     0
   Session vcid is 123
   Session Layer 2 circuit, type is Ethernet,
name is FastEthernet1/1/0
   Circuit state is UP
     Remote session id is 26570, remote tunnel
id 46882
   DF bit off, ToS reflect disabled, ToS value
0, TTL value 255
   No session cookie information available
   FS cached header information:
     encap size = 24 bytes
       00000000 00000000 00000000 00000000
       00000000 00000000
   Sequencing is off
```

PE2

```
Router# show l2tun session all

 Session Information Total tunnels 1 sessions 1

Session id 26570 is up, tunnel id 46882
Call serial number is 4035100045
Remote tunnel name is PE1
   Internet address is 10.8.8.8
   Session is L2TP signalled
   Session state is established, time since change 1d22h
   16 Packets sent, 16 received
   1230 Bytes sent, 1230 received
   Receive packets dropped:
     out-of-order:              0
     total:                     0
   Send packets dropped:
     exceeded session MTU:      0
     total:                     0
   Session vcid is 123
   Session Layer 2 circuit, type is Ethernet Vlan, name is
FastEthernet2/0.1:10
   Circuit state is UP, interworking type is Ethernet
    Remote session id is 15736, remote tunnel id 35411
    DF bit off, ToS reflect disabled, ToS value 0, TTL
value 255
   No session cookie information available
   FS cached header information:
     encap size = 24 bytes
     00000000 00000000 00000000 00000000
     00000000 00000000
   Sequencing is off
```

Step 3 **show arp**

You can issue the **show arp** command between the CE routers to ensure that data is being sent:

```
Router# show arp

Protocol   Address        Age (min)   Hardware Addr    Type    Interface
Internet   10.1.1.5             134    0005.0032.0854   ARPA    FastEthernet0/0
Internet   10.1.1.7               -    0005.0032.0000   ARPA    FastEthernet0/0
```

Step 4 **ping**

You can issue the **ping** command between the CE routers to ensure that data is being sent:

```
Router# ping 10.1.1.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Step 5 **show l2tun session interworking** (L2TPv3 only)

For L2TPv3, you can verify that the interworking type is correctly set using the **show l2tun session interworking** command. Enter the command on the PE routers that are performing the interworking translation.

- In Example 1, the PE router performs the raw Ethernet translation. The command output displays the interworking type with a dash (-).
- In Example 2, the PE router performs the Ethernet VLAN translation. The command output displays the interworking type as ETH.

Example 1 Command Output for Raw Ethernet Translation

```
Router# show l2tun session interworking

Session Information Total tunnels 1 sessions 1

LocID      TunID      Peer-address    Type IWrk Username, Intf/Vcid, Circuit
15736      35411      10.9.9.9         ETH   -   123,      Fa1/1/0
```

Example 2 Command Output for Ethernet VLAN Translation

```
Router# show l2tun session interworking

Session Information Total tunnels 1 sessions 1

LocID      TunID      Peer-address    Type IWrk Username, Intf/Vcid, Circuit
26570      46882      10.8.8.8         VLAN ETH  123,      Fa2/0.1:10
```

Step 6 **show mpls l2transport vc detail** (AToM only)

You can verify the AToM configuration by using the **show mpls l2transport vc detail** command. In the following example, the interworking type is shown in bold.

PE1	PE2

`Router# show mpls l2transport vc detail`
 	
`Local interface: Fa1/1/0 up, line protocol up, Ethernet up`
`  Destination address: 10.9.9.9, VC ID: 123, VC status: up`
`    Preferred path: not configured`
`    Default path: active`
`    Tunnel label: 17, next hop 10.1.1.3`
`    Output interface: Fa4/0/0, imposed label stack {17 20}`
`  Create time: 01:43:50, last status change time: 01:43:33`
`  Signaling protocol: LDP, peer 10.9.9.9:0 up`
`    MPLS VC labels: local 16, remote 20`
`    Group ID: local 0, remote 0`
`    MTU: local 1500, remote 1500`
`    Remote interface description:`
`  Sequencing: receive disabled, send disabled`
`  VC statistics:`
`    packet totals: receive 15, send 4184`
`    byte totals:   receive 1830, send 309248`
`    packet drops:  receive 0, send 0`
	`Router# show mpls l2transport vc detail`
	
	`Local interface: Fa2/0.3 up, line protocol up, Eth VLAN 10 up`
	`  MPLS VC type is Ethernet, interworking type is Ethernet`
	`  Destination address: 10.8.8.8, VC ID: 123, VC status: up`
	`    Preferred path: not configured`
	`    Default path: active`
	`    Tunnel label: 16, next hop 10.1.1.3`
	`    Output interface: Fa6/0, imposed label stack {16 16}`
	`  Create time: 00:00:26, last status change time: 00:00:06`
	`  Signaling protocol: LDP, peer 10.8.8.8:0 up`
	`    MPLS VC labels: local 20, remote 16`
	`    Group ID: local 0, remote 0`
	`    MTU: local 1500, remote 1500`
	`    Remote interface description:`
	`  Sequencing: receive disabled, send disabled`
	`  VC statistics:`
	`    packet totals: receive 5, send 0`
	`    byte totals:   receive 340, send 0`
	`    packet drops:  receive 0, send 0`

Configuring L2VPN Interworking: VLAN Enable/Disable Option for AToM

You can specify the Ethernet VLAN (type 4) by issuing the **interworking vlan** command in pseudowire-class configuration mode. This allows the VLAN ID to be included as part of the Ethernet frame. In releases previous to Cisco IOS Release 12.2(52)SE and Cisco IOS Release 12.2(33)SRE, the only way to achieve VLAN encapsulation is to ensure the CE router is connected to the PE router through an Ethernet link.

Restrictions

In Cisco IOS Release 12.2(52)SE and Cisco IOS Release 12.2(33)SRE, the **encapsulation** command supports only the **mpls** keyword. The **l2tpv3** keyword is not supported. The **interworking** command supports only the **ethernet** and **vlan** keywords. The **ip** keyword is not supported.

Prerequisites

For complete instructions on configuring AToM, see *Any Transport over MPLS.*

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **pseudowire-class** *name*

4. **encapsulation** {**mpls** | **l2tpv3**}
5. **interworking** {**ethernet** | **ip** | **vlan**}
6. **end**
7. **show mpls l2transport vc** [**vcid** *vc-id* | **vcid** *vc-id-min vc-id-max*] [**interface** *type number* [*local-circuit-id*]] [**destination** *ip-address* | *name*] [**detail**]

DETAILED STEPS

	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 class1`	Establishes a pseudowire class with a name that you specify and enters pseudowire class configuration mode.
Step 4	**encapsulation** {**mpls** | **l2tpv3**} **Example:** `Router(config-pw)# encapsulation mpls`	Specifies the tunneling encapsulation, which is either **mpls** or **l2tpv3**. • For the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature, only MPLS encapsulation is supported.
Step 5	**interworking** {**ethernet** | **ip** | **vlan**} **Example:** `Router(config-pw)# interworking vlan`	Specifies the type of pseudowire and the type of traffic that can flow across it. • For the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature, specify the **vlan** keyword.
Step 6	**end** **Example:** `Router(config-pw)# end`	Exits pseudowire class configuration mode and enters privileged EXEC mode.
Step 7	**show mpls l2transport vc** [**vcid** *vc-id* | **vcid** *vc-id-min vc-id-max*] [**interface** *type number* [*local-circuit-id*]] [**destination** *ip-address* | *name*] [**detail**] **Example:** `Router# show mpls l2transport vc detail`	Displays information about AToM VCs.

Examples

When the pseudowire on an interface is different from the VC type, the interworking type is displayed in the **show mpls l2transport vc detail** command output. In the following example, the pseudowire is configured on an Ethernet port and VLAN interworking is configured in the pseudowire class. The relevant output is shown in bold:

```
PE1# show mpls l2 vc 34 detail

Local interface: Et0/1 up, line protocol up, Ethernet up
  MPLS VC type is Ethernet, interworking type is Eth VLAN
  Destination address: 10.1.1.2, VC ID: 34, VC status: down
    Output interface: if-?(0), imposed label stack {}
    Preferred path: not configured
    Default path: no route
    No adjacency
  Create time: 00:00:13, last status change time: 00:00:13
  Signaling protocol: LDP, peer unknown
    Targeted Hello: 10.1.1.1(LDP Id) -> 10.1.1.2
    Status TLV support (local/remote)   : enabled/None (no remote binding)
      LDP route watch                   : enabled
      Label/status state machine        : local standby, AC-ready, LnuRnd
      Last local dataplane   status rcvd: No fault
      Last local SSS circuit status rcvd: No fault
      Last local SSS circuit status sent: Not sent
      Last local  LDP TLV    status sent: None
      Last remote LDP TLV    status rcvd: None (no remote binding)
      Last remote LDP ADJ    status rcvd: None (no remote binding)
    MPLS VC labels: local 2003, remote unassigned
    Group ID: local 0, remote unknown
    MTU: local 1500, remote unknown
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 0, send 0
    byte totals:   receive 0, send 0
    packet drops:  receive 0, seq error 0, send 0
```

Configuration Examples for L2VPN Interworking

The following sections show examples of L2VPN Interworking:

- Ethernet to VLAN over L2TPV3 (Bridged): Example, page 22
- Ethernet to VLAN over AToM (Bridged): Example, page 22
- Frame Relay to VLAN over L2TPV3 (Routed): Example, page 23
- Frame Relay to VLAN over AToM (Routed): Example, page 24
- Frame Relay to ATM AAL5 over AToM (Routed): Example, page 25
- VLAN to ATM AAL5 over AToM (Bridged): Example, page 26
- Frame Relay to PPP over L2TPv3 (Routed): Example, page 27
- Frame Relay to PPP over AToM (Routed): Example, page 28
- Ethernet/VLAN to PPP over AToM (Routed): Example, page 29
- Additional References, page 29

Ethernet to VLAN over L2TPV3 (Bridged): Example

The following example shows the configuration of Ethernet to VLAN over L2TPv3:

PE1	PE2
ip cef ! l2tp-class interworking-class authentication hostname PE1 password 0 lab ! pseudowire-class inter-ether-vlan encapsulation l2tpv3 interworking ethernet protocol l2tpv3 interworking-class ip local interface Loopback0 ! interface Loopback0 ip address 10.8.8.8 255.255.255.255 ! interface FastEthernet1/0 xconnect 10.9.9.9 1 pw-class inter-ether-vlan	ip cef ! l2tp-class interworking-class authentication hostname PE2 password 0 lab ! pseudowire-class inter-ether-vlan encapsulation l2tpv3 interworking ethernet protocol l2tpv3 interworking-class ip local interface Loopback0 ! interface Loopback0 ip address 10.9.9.9 255.255.255.255 ! interface FastEthernet0/0 no ip address ! interface FastEthernet0/0.3 encapsulation dot1Q 10 xconnect 10.8.8.8 1 pw-class inter-ether-vlan

Ethernet to VLAN over AToM (Bridged): Example

The following example shows the configuration of Ethernet to VLAN over AToM:

PE1	PE2
ip cef ! mpls label protocol ldp mpls ldp router-id Loopback0 force ! pseudowire-class atom-eth-iw encapsulation mpls interworking ethernet ! interface Loopback0 ip address 10.8.8.8 255.255.255.255 ! interface FastEthernet1/0.1 encapsulation dot1q 100 xconnect 10.9.9.9 123 pw-class atom-eth-iw	ip cef ! mpls label protocol ldp mpls ldp router-id Loopback0 force ! pseudowire-class atom encapsulation mpls ! interface Loopback0 ip address 10.9.9.9 255.255.255.255 ! interface FastEthernet0/0 no ip address ! interface FastEthernet1/0 xconnect 10.9.9.9 123 pw-class atom

Frame Relay to VLAN over L2TPV3 (Routed): Example

The following example shows the configuration of Frame Relay to VLAN over L2TPv3:

PE1	PE2

configure terminal
ip cef
frame-relay switching
!
!
interface loopback 0
ip address 10.8.8.8 255.255.255.255
no shutdown
!
pseudowire-class ip
 encapsulation l2tpv3
 interworking ip
 ip local interface loopback0
!
interface POS1/0
encapsulation frame-relay
clock source internal
logging event dlci-status-change
no shutdown
no fair-queue
!
connect fr-vlan POS1/0 206 l2transport
 xconnect 10.9.9.9 6 pw-class ip
!
router ospf 10
 network 10.0.0.2 0.0.0.0 area 0
 network 10.8.8.8 0.0.0.0 area 0
	configure terminal
	ip routing
	ip cef
	frame-relay switching
	!
	interface loopback 0
	ip address 10.9.9.9 255.255.255.255
	no shutdown
	!
	pseudowire-class ip
	 encapsulation l2tpv3
	 interworking ip
	 ip local interface loopback0
	!
	interface FastEthernet1/0/1
	  speed 10
	  no shutdown
	!
	interface FastEthernet1/0/1.6
	encapsulation dot1Q 6
	xconnect 10.8.8.8 6 pw-class ip
	no shutdown
	!
	router ospf 10
	 network 10.0.0.2 0.0.0.0 area 0
	 network 10.9.9.9 0.0.0.0 area 0

Frame Relay to VLAN over AToM (Routed): Example

The following example shows the configuration of Frame Relay to VLAN over AToM:

PE1	PE2
configure terminal ip cef frame-relay switching ! mpls label protocol ldp mpls ldp router-id loopback0 mpls ip ! pseudowire-class atom encapsulation mpls interworking ip ! interface loopback 0 ip address 10.8.8.8 255.255.255.255 no shutdown ! connect fr-vlan POS1/0 206 l2transport xconnect 10.9.9.9 6 pw-class atom	configure terminal ip routing ip cef frame-relay switching ! mpls label protocol ldp mpls ldp router-id loopback0 mpls ip ! pseudowire-class atom encapsulation mpls interworking ip ! interface loopback 0 ip address 10.9.9.9 255.255.255.255 no shutdown ! interface FastEthernet1/0/1.6 encapsulation dot1Q 6 xconnect 10.8.8.8 6 pw-class atom no shutdown

Frame Relay to ATM AAL5 over AToM (Routed): Example

Note Frame Relay to ATM AAL5 is available only with AToM in IP mode.

The following example shows the configuration of Frame Relay to ATM AAL5 over AToM:

PE1	PE2
ip cef frame-relay switching mpls ip mpls label protocol ldp mpls ldp router-id loopback0 force pseudowire-class fratmip encapsulation mpls interworking ip interface Loopback0 ip address 10.33.33.33 255.255.255.255 interface serial 2/0 encapsulation frame-relay ietf frame-relay intf-type dce connect fr-eth serial 2/0 100 l2transport xconnect 10.22.22.22 333 pw-class fratmip interface POS1/0 ip address 10.1.7.3 255.255.255.0 crc 32 clock source internal mpls ip mpls label protocol ldp router ospf 10 passive-interface Loopback0 network 10.33.33.33 0.0.0.0 area 10 network 10.1.7.0 0.0.0.255 area 10	ip cef mpls ip mpls label protocol ldp mpls ldp router-id loopback0 force pseudowire-class fratmip encapsulation mpls interworking ip interface Loopback0 ip address 10.22.22.22 255.255.255.255 interface ATM 2/0 pvc 0/203 l2transport encapsulation aa5snap xconnect 10.33.33.33 333 pw-class fratmip interface POS1/0 ip address 10.1.1.2 255.255.255.0 crc 32 clock source internal mpls ip mpls label protocol ldp router ospf 10 passive-interface Loopback0 network 10.22.22.22 0.0.0.0 area 10 network 10.1.1.0 0.0.0.255 area 10

VLAN to ATM AAL5 over AToM (Bridged): Example

The following example shows the configuration of VLAN to ATM AAL5 over AToM:

PE1	PE2
ip cef ! mpls ip mpls label protocol ldp mpls ldp router-id Loopback0 ! pseudowire-class inter-ether encapsulation mpls interworking ethernet ! interface Loopback0 ip address 10.8.8.8 255.255.255.255 ! interface ATM1/0.1 point-to-point pvc 0/100 l2transport encapsulation aal5snap xconnect 10.9.9.9 123 pw-class inter-ether ! interface FastEthernet1/0 xconnect 10.9.9.9 1 pw-class inter-ether ! router ospf 10 log-adjacency-changes network 10.8.8.8 0.0.0.0 area 0 network 10.1.1.1 0.0.0.0 area 0	ip cef ! mpls ip mpls label protocol ldp mpls ldp router-id Loopback0 ! pseudowire-class inter-ether encapsulation mpls interworking ethernet ! interface Loopback0 ip address 10.9.9.9 255.255.255.255 ! interface FastEthernet0/0 no ip address ! interface FastEthernet0/0.1 encapsulation dot1Q 10 xconnect 10.8.8.8 123 pw-class inter-ether ! router ospf 10 log-adjacency-changes network 10.9.9.9 0.0.0.0 area 0 network 10.1.1.2 0.0.0.0 area 0

Frame Relay to PPP over L2TPv3 (Routed): Example

The following example shows the configuration of Frame Relay to PPP over L2TPv3:

PE1	PE2
ip cef ip routing ! ! ! pseudowire-class ppp-fr encapsulation l2tpv3 interworking ip ip local interface Loopback0 ! interface Loopback0 ip address 10.1.1.1 255.255.255.255 ! interface FastEthernet1/0/0 ip address 10.16.1.1 255.255.255.0 ! interface Serial3/0/0 no ip address encapsulation ppp ppp authentication chap ! ip route 10.0.0.0 255.0.0.0 10.16.1.2 ! xconnect 10.2.2.2 1 pw-class ppp-fr ppp ipcp address proxy 10.65.32.14	ip cef ip routing ! frame-relay switching ! pseudowire-class ppp-fr encapsulation l2tpv3 interworking ip ip local interface Loopback0 ! interface Loopback0 ip address 10.2.2.2 255.255.255.255 ! interface FastEthernet1/0/0 ip address 10.16.2.1 255.255.255.0 ! interface Serial3/0/0 no ip address encapsulation frame-relay frame-relay intf-type dce ! ip route 10.0.0.0 255.0.0.0 10.16.2.2 ! connect ppp-fr Serial3/0/0 100 l2transport xconnect 10.1.1.1 100 pw-class ppp-fr

Frame Relay to PPP over AToM (Routed): Example

The following example shows the configuration of Frame Relay to PPP over AToM:

PE1	PE2
ip cef ip routing mpls label protocol ldp mpls ldp router-id loopback0 force ! ! ! pseudowire-class ppp-fr encapsulation mpls interworking ip ip local interface Loopback0 ! interface Loopback0 ip address 10.1.1.1 255.255.255.255 ! interface FastEthernet1/0/0 ip address 10.16.1.1 255.255.255.0 mpls ip label protocol ldp ! interface Serial3/0/0 no ip address encapsulation ppp ppp authentication chap xconnect 10.2.2.2 1 pw-class ppp-fr ppp ipcp address proxy 10.65.32.14 ! ip route 10.0.0.0 255.0.0.0 10.16.1.2	ip cef ip routing mpls label protocol ldp mpls ldp router-id loopback0 force ! frame-relay switching ! pseudowire-class ppp-fr encapsulation mpls interworking ip ip local interface Loopback0 ! interface Loopback0 ip address 10.2.2.2 255.255.255.255 ! interface FastEthernet1/0/0 ip address 10.16.2.1 255.255.255.0 mpls ip mpls label protocol ldp ! interface Serial3/0/0 no ip address encapsulation frame-relay frame-relay intf-type dce ! ip route 10.0.0.0 255.0.0.0 10.16.2.2 ! connect ppp-fr Serial3/0/0 100 l2transport xconnect 10.1.1.1 100 pw-class ppp-fr

Ethernet/VLAN to PPP over AToM (Routed): Example

The following example shows the configuration of Ethernet VLAN to PPP over AToM:

PE1	PE2
configure terminal mpls label protocol ldp mpls ldp router-id Loopback0 mpls ip ! pseudowire-class ppp-ether encapsulation mpls interworking ip ! interface Loopback0 ip address 10.8.8.8 255.255.255.255 no shutdown ! interface POS2/0/1 no ip address encapsulation ppp no peer default ip address ppp ipcp address proxy 10.10.10.1 xconnect 10.9.9.9 300 pw-class ppp-ether no shutdown	configure terminal mpls label protocol ldp mpls ldp router-id Loopback0 mpls ip ! pseudowire-class ppp-ether encapsulation mpls interworking ip ! interface Loopback0 ip address 10.9.9.9 255.255.255.255 no shutdown ! interface vlan300 mtu 4470 no ip address xconnect 10.8.8.8 300 pw-class ppp-ether no shutdown ! interface GigabitEthernet6/2 switchport switchport trunk encapsulation dot1q switchport trunk allowed vlan 300 switchport mode trunk no shutdown

Additional References

The following sections provide references related to the L2VPN Interworking feature.

Related Documents

Related Topic	Document Title
Layer 2 Tunnel Protocol Version 3	*Layer 2 Tunnel 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	• *Cisco 7600 Series Routers Documentation Roadmap* • *Release Notes for Cisco IOS Release 12.2SR for the Cisco 7600 Series Routers*
Cisco 3270 series routers hardware support	*Cisco IOS Software Releases 12.2SE Release Notes*

Standards

Standards	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

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs 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/techsupport

Feature Information for L2VPN Interworking

Table 6 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 6 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 6 Feature Information for L2VPN Interworking

Feature Name	Releases	Feature Information
L2VPN Interworking	12.0(26)S 12.0(30)S 12.0(32)S 12.0(32)SY 12.2(33)SRA 12.4(11)T 12.2(33)SXH 12.2(33)SRD 12.2(52)SE 12.2(33)SRE	This feature allows disparate attachment circuits to be connected. An interworking function facilitates the translation between the different Layer 2 encapsulations. This feature was introduced in Cisco IOS Release 12.0(26)S. In Cisco IOS Release 12.0(30)S, support was added for Cisco 12000 series Internet routers. In Cisco IOS Release 12.0(32)S, support was added on Engine 5 line cards (SIP-401, SIP-501, SIP-600, and SIP-601) in Cisco 12000 series routers for the following four transport types: • Ethernet/VLAN to Frame Relay Interworking • Ethernet/VLAN to ATM AAL5 Interworking • Ethernet to VLAN Interworking • Frame Relay to ATM AAL5 Interworking On the Cisco 12000 series Internet router, support was added for IP Services Engine (ISE) and Engine 5 line cards that are configured for L2TPv3 tunneling (see *Layer 2 Tunnel Protocol Version 3*). In Cisco IOS Release 12.2(33)SRA, support was added for the Cisco 7600 series routers. In Cisco IOS Release 12.4(11)T, support was added for the following transport types: • Ethernet to VLAN Interworking • Ethernet/VLAN to Frame Relay Interworking This feature was integrated into Cisco IOS Release 12.2(33)SXH. In Cisco IOS Release 12.2(33)SRD, support for routed and bridged interworking on SIP-400 was added for the Cisco 7600 series routers. In Cisco IOS Release 12.2(52)SE, the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature was added for the Cisco 3750 Metro switch. In Cisco IOS Release 12.2(33)SRE, the L2VPN Interworking: VLAN Enable/Disable Option for AToM feature was added for the Cisco 7600 series router. The following commands were introduced or modified: **interworking**

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website 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. (0910R)

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

© 2003-2009 Cisco Systems, Inc. All rights reserved.

Layer 2 Local Switching

First Published: December 17, 2003
Last Updated: April 21, 2008

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 interface-to-interface switching combinations supported by this feature are:

- ATM to ATM
- ATM to Ethernet
- ATM to Frame Relay
- Ethernet to Ethernet VLAN
- Frame Relay to Frame Relay (and Multilink Frame Relay in Cisco IOS Release 12.0(28)S and later)
- High-Level Data Link Control (HDLC) HDLC

The following same-port switching features are supported:

- ATM (Permanent Virtual Circuit (PVC) and Permanent Virtual Path (PVP)
- Ethernet VLAN
- Frame Relay

Finding Feature Information in This Module

Your Cisco IOS software release may not support all of the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for Layer 2 Local Switching" section on page 34.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

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

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2003—2008 Cisco Systems, Inc. All rights reserved.

Contents

- Prerequisites for Layer 2 Local Switching, page 2
- Restrictions for Layer 2 Local Switching, page 2
- Information About Layer 2 Local Switching, page 5
- How to Configure Layer 2 Local Switching, page 7
- Configuration Examples for Layer 2 Local Switching, page 27
- Additional References, page 32
- Command Reference, page 33
- Feature Information for Layer 2 Local Switching, page 34

Prerequisites for Layer 2 Local Switching

- You must enable Cisco Express Forwarding for the Cisco 7200 series router. You must use Cisco Express Forwarding or Distributed Cisco Express Forwarding for the Cisco 7500 series router. (Distributed Cisco Express Forwarding is enabled already by default on the Cisco 12000 series routers).
- For Frame Relay local switching, you must globally issue the **frame-relay switching** command.

Restrictions for Layer 2 Local Switching

The following sections list the restrictions for the Layer 2 Local Switching feature:

- Cisco 7200 and 7500 Series Router Restrictions, page 2
- Cisco 7600 and 6500 Series Router Restrictions, page 4
- Cisco 7600 and 6500 Series Router Restrictions, page 4
- Cisco 10000 Series Router Restrictions, page 4
- Cisco 12000 Series Router Restrictions, page 4
- Unsupported Hardware, page 5

Cisco 7200 and 7500 Series Router Restrictions

- In ATM single cell relay AAL0, the ATM virtual path identifier/virtual channel identifier (VPI/VCI) values must match between the ingress and egress ATM interfaces on the Cisco 7200 series and 7500 series routers. If Layer 2 local switching is desired between two ATM VPIs and VCIs whose values do not match and are on two different interfaces, choose ATM AAL5. However, if the ATM AAL5 is using Operation, Administration, and Maintenance (OAM) transparent mode, the VPI and VCI values must match.
- NSF/SSO: Layer 2 Local Switching is supported on Cisco 7500 series routers.

Layer 2 local switching is supported on the following interface processors in the Cisco 7200 series routers:

- C7200-I/O-2FE

- C7200-I/O-GE+E (Only the Gigabit Ethernet port of this port adapter is supported.)
- C7200-I/O-FE

Layer 2 local switching is supported on the following interface processors in the Cisco 7500 series routers:

- GEIP (Gigabit Ethernet interface processor)
- GEIP+ (enhanced Gigabit Ethernet interface processor)

Layer 2 local switching is supported on the following port adapters in the Cisco 7200 and 7500 series routers:

- PA-FE-TX (single-port Fast Ethernet 100BASE-TX)
- PA-FE-FX (single-port Fast Ethernet 100BASE-FX)
- PA-2FE-TX (dual-port Fast Ethernet 100BASE-TX)
- PA-2FE-FX (dual-port Fast Ethernet 100BASE-FX)
- PA-4E (4-port Ethernet adapter)
- PA-8E (8-port Ethernet adapter)
- PA-4T (4-port synchronous serial port adapter)
- PA-4T+ (enhanced 4-port synchronous serial port adapter)
- PA-8T (8-port synchronous serial port adapter)
- PA-12E/2FE (12-port Ethernet/2-port Fast Ethernet (FE) adapter) [Cisco 7200 only]
- PA-GE (Gigabit Ethernet port adapter) [Cisco 7200 only]
- PA-H (single-port High-Speed Serial Interface (HSSI) adapter)
- PA-2H (dual-port HSSI adapter)
- PA-MC-8E1 (8-port multichannel E1 G.703/G.704 120-ohm interfaces)
- PA-MC-2EI (2-port multichannel E1 G.703/G.704 120-ohm interfaces)
- PA-MC-8T1 (8-port multichannel T1 with integrated data service units (DSUs) and channel service units CSUs))
- PA-MC-4T1 (4-port multichannel T1 with integrated CSUs and DSUs)
- PA-MC-2T1 (2-port multichannel T1 with integrated CSUs and DSUs)
- PA-MC-8TE1+ (8-port multichannel T1/E1)
- PA-MC-T3 (1-port multichannel T3 interface)
- PA-MC-E3 (1-port multichannel E3 interface)
- PA-MC-2T3+ (2-port enhanced multichannel T3 port adapter)
- PA-MC-STM1 (1-port multichannel STM-1 port adapter) [Cisco 7500 only]
- PA-T3 (single-port T3 port adapter)
- PA-E3 (single-port E3 port adapter)
- PA-2E3 (2-port E3 port adapter)
- PA-2T3 (2-port T3 port adapter)
- PA-POS-OC-3SML (single-port Packet over SONET (POS), single-mode, long reach)
- PA-POS-OC-3SMI (single-port PoS, single-mode, intermediate reach)

- PA-POS-OC-3MM (single-port PoS, multimode)
- PA-A3-OC-3 (1-port ATM OC-3/STM1 port adapter, enhanced)
- PA-A3-OC-12 (1-port ATM OC-12/STM-4 port adapter, enhanced) [Cisco 7500 only]
- PA-A3-T3 (DS3 high-speed interface)
- PA-A3-E3 (E3 medium-speed interface)
- PA-A3-8T1IMA (ATM inverse multiplexer over ATM port adapter with 8 T1 ports)
- PA-A3-8E1IMA (ATM inverse multiplexer over ATM port adapter with 8 E1 ports)
- PA-A6 (Cisco ATM Port Adapter)

Cisco 7600 and 6500 Series Router Restrictions

- Layer 2 Local Switching supports the following port adapters and interface processors on the Cisco 7600-SUP720/MSFC3 router:
  - All port adapters on the Enhanced FlexWAN module
  - All SPAs on the SIP-200 line cards
- On the Cisco 6500 series and 7600 series routers, only *like-to-like* local switching is supported (ATM to ATM and Frame Relay to Frame Relay).
- Same-port switching is not supported on the Cisco 6500 series and 7600 series routers.

Cisco 10000 Series Router Restrictions

For information about Layer 2 Local Switching on the Cisco 10000 series routers, see the "Configuring Layer 2 Local Switching " section of the *Cisco 10000 Series Router Broadband Aggregation, Leased-Line, and MPLS Configuration Guide*.

Cisco 12000 Series Router Restrictions

- VPI/VCI rewrite is supported on the Cisco 12000 series routers.
- All Cisco 12000 series line cards support Frame Relay-to-Frame Relay local switching.
- 8-port OC-3 ATM Engine 2 line cards support only like-to-like Layer 2 local switching.
- ISE (Engine 3) line cards support like-to-like and any-to-any local switching. Non-ISE line cards support only like-to-like local switching.

  Starting in Cisco IOS Release 12.0(31)S2, ISE customer edge-facing interfaces support the following types of like-to-like and any-to-any local switching:
  - ATM to ATM
  - ATM to Ethernet
  - ATM to Frame Relay
  - Ethernet to Ethernet VLAN
  - Frame Relay to Frame Relay (including Multilink Frame Relay)
  - Same-port switching for ATM (PVC and PVP)

  - Same-port switching for Ethernet VLAN
  - Same-port switching for Frame Relay

Note Native Layer 2 Tunnel Protocol Version 3 (L2TPv3) tunnel sessions on customer edge-facing line cards can coexist with tunnel sessions that use a tunnel-server card.

- Starting in Cisco IOS Release 12.0(32)SY, customer edge-facing interfaces on Engine 5 shared port adapters (SPAs) and SPA Interface Processors (SIPs) support the following types of like-to-like local switching:
  - Ethernet to Ethernet VLAN
  - Frame Relay to Frame Relay (including Multilink Frame Relay)
  - Same-port switching for Ethernet VLAN
  - Same-port switching for Frame Relay
- For ATM-to-ATM local switching, the following ATM types are supported for the Layer 2 Local Switching feature:
  - ATM adaptation layer 5 (AAL5)
  - ATM single cell relay adaptation layer 0 (AAL0), VC mode
  - ATM single cell relay VP mode on the Cisco 12000 series router
  - ATM single cell relay VC and VP modes on ISE line cards on the Cisco 12000 series router
- Starting with Cisco IOS Release 12.0(30)S, you can use Local Switching and cell packing with ATM VP or VC mode on the Cisco 12000 series router on IP Services Engine (ISE/Engine 3) line cards. For information about how to configure cell packing, refer to *Any Transport over MPLS*.

Unsupported Hardware

The following hardware is not supported:

- Cisco 7200—non-VXR chassis
- Cisco 7500—Route Switch Processor (RSP)1 and 2
- Cisco 7500—Versatile Interface Processor (VIP) 2-40 and below
- Cisco 12000 series—4-port OC-3 ATM Engine-0 line card
- Cisco 12000 series—4-port OC-12 ATM Engine-2 line card
- Cisco 12000 series—1-port OC-12 ATM Engine-0 line card
- Cisco 12000 series—Ethernet Engine-1, Engine-2, and Engine-4 line cards

Information About Layer 2 Local Switching

To configure the the Layer 2 Local Switching feature, you should understand the following concepts:

- Layer 2 Local Switching Overview, page 6
- NSF/SSO—Local Switching Overview, page 6

- Layer 2 Local Switching Applications, page 6

For information about Layer 2 Local Switching on the Cisco 10000 series routers, see the "Configuring Layer 2 Local Switching " section of the *Cisco 10000 Series Router Broadband Aggregation, Leased-Line, and MPLS Configuration Guide*.

Layer 2 Local Switching Overview

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.

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 to Ethernet VLAN
- Frame Relay to Frame Relay

Layer 2 Local Switching Applications

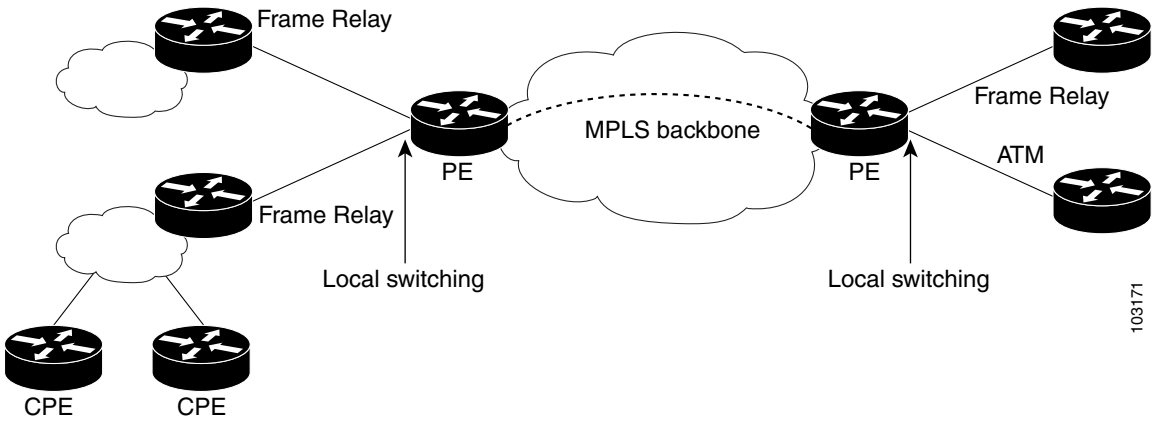
Incumbent local exchange carriers (ILECs) who use an interexchange carrier (IXC) to carry traffic between two local exchange carriers can use the Layer 2 Local Switching feature. Telecom regulations require the ILECs to pay the IXCs to carry that traffic. At times, the ILECs cannot terminate customer 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.

Figure 1 shows a network that uses local switching for both Frame Relay to Frame Relay and ATM to Frame Relay local switching.

Figure 1 Local Switching Example



How to Configure Layer 2 Local Switching

The following sections explain the how to configure the each type of Layer 2 Local Switching:

- Configuring ATM-to-ATM PVC Local Switching and Same-Port Switching, page 7 (optional)
- Configuring ATM-to-ATM PVP Local Switching, page 9 (optional)
- Configuring ATM PVP Same-Port Switching, page 10 (optional)
- Configuring ATM-to-Ethernet Port Mode Local Switching, page 11 (optional)
- Configuring ATM-to-Ethernet VLAN Mode Local Switching, page 13 (optional)
- Configuring Ethernet VLAN Same-Port Switching, page 15 (optional)
- Configuring Ethernet Port Mode to Ethernet VLAN Local Switching, page 17 (optional)
- Configuring ATM-to-Frame Relay Local Switching, page 18 (optional)
- Configuring Frame Relay-to-Frame Relay Local Switching, page 20 (optional)
- Configuring Frame Relay Same-Port Switching, page 22 (optional)
- Configuring HDLC Local Switching, page 23 (optional)
- Verifying Layer 2 Local Switching, page 25 (optional)

For information about Layer 2 Local Switching on the Cisco 10000 series routers, see the "Configuring Layer 2 Local Switching " section of the *Cisco 10000 Series Router Broadband Aggregation, Leased-Line, and MPLS Configuration Guide*.

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.

Note Starting with Cisco IOS Release 12.0(30)S, you can configure same-port switching following the steps in this section.

Use the following steps to configure ATM-to-ATM PVC local switching and same-port switching.

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

	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`	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 1/200 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.
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.

	Command or Action	Purpose
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 0/100 atm2/0/0 0/100`	Creates a local connection between the two specified permanent virtual circuits.

Configuring ATM-to-ATM PVP Local Switching

Use the following steps 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" section on page 10.

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

	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	**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`	In global configuration mode, creates a local connection between the two specified permanent virtual paths.

Configuring ATM PVP Same-Port Switching

Use the following steps 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

	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`	In global configuration mode, creates the local connection between the two specified permanent virtual paths.

Configuring ATM-to-Ethernet Port Mode Local Switching

For ATM to Ethernet port mode local switching, creating the ATM PVC is not required. If you do not create a PVC, one is created for you. For ATM-to-Ethernet local switching, the autoprovisioned PVC is given the default encapsulation type AAL5SNAP.

ATM-to-Ethernet local switching supports both the IP and Ethernet interworking types. When the Ethernet interworking type is used, the interworking device (router) expects a bridged packet. Therefore, configure the ATM CPE for either IRB or RBE.

Note Enabling ICMP Router Discovery Protocol on the Ethernet side is recommended.

ATM-to-Ethernet local switching supports the following encapsulation types:

- ATM-to-Ethernet with IP interworking: AAL5SNAP, AAL5MUX
- ATM-to-Ethernet with Ethernet interworking: AAL5SNAP

Use the following steps to configure local switching between ATM and Ethernet port mode.

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. **interface fastethernet***slot*/*subslot*/*port*
9. **exit**
10. **connect** *connection-name interface pvc interface* **interworking ip | ethernet**

DETAILED STEPS

	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`	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 1/200 l2transport`	Assigns a VPI and VCI and enters 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 aal5snap`	Specifies the encapsulation type for the PVC.
Step 6	**exit** **Example:** `Router(config-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	**interface fastethernet***slot***/***subslot***/***port*** **Example:** `Router(config)# interface fastethernet6/0/0`	Specifies a Fast Ethernet line card, subslot (if available), and port, and enters 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 pvc interface* **interworking ip** | **ethernet** **Example:** `Router(config)# connect atm-eth-con atm1/0 0/100 fastethernet6/0/0 interworking ethernet`	In global configuration mode, creates a local connection between the two interfaces and specifies the interworking type. • Both the IP and Ethernet interworking types are supported.

Configuring ATM-to-Ethernet VLAN Mode Local Switching

For ATM-to-Ethernet VLAN mode local switching, creating the ATM PVC is not required. If you do not create a PVC, one is created for you. For ATM-to-Ethernet local switching, the autoprovisioned PVC is given the default encapsulation type AAL5SNAP.

ATM-to-Ethernet local switching supports both the IP and Ethernet interworking types. When the Ethernet interworking type is used, the interworking device (router) expects a bridged packet. Therefore, configure the ATM CPE for either IRB or RBE.

Note Enabling ICMP Router Discovery Protocol on the Ethernet side is recommended.

ATM-to-Ethernet local switching supports the following encapsulation types:

- ATM-to-Ethernet with IP interworking: AAL5SNAP, AAL5MUX
- ATM-to-Ethernet with Ethernet interworking: AAL5SNAP

The VLAN header is removed from frames that are received on an Ethernet subinterface.

Use the following steps to configure local switching for ATM to Ethernet in VLAN mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm***slot*/*subslot*/*port*
4. **pvc** *vpi*/*vci* **l2transport**
5. **encapsulation** *layer-type*
6. **exit**
7. **interface fastethernet***slot*/*port*/*subinterface-number*
8. **encapsulation dot1q** *vlan-id*
9. **exit**
10. **connect** *connection-name interface pvc interface* **interworking ip | ethernet**

DETAILED STEPS

	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	**pvc** *vpi*/*vci* **l2transport** **Example:** `Router(config-if)# pvc 1/200 l2transport`	Assigns a VPI and VCI and enters PVC l2transport configuration mode. • The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.
Step 5	**encapsulation** *layer-type* **Example:** `Router(cfg-if-atm-l2trans-pvc)# encapsulation aal5snap`	Specifies the encapsulation type for the PVC.
Step 6	**exit** **Example:** `Router(cfg-if-atm-l2trans-pvc)# exit`	Exits PVC l2transport configuration mode and returns to interface configuration mode.

	Command or Action	Purpose
Step 7	**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.
Step 8	**encapsulation dot1q** *vlan-id* **Example:** `Router(config-subif)# encapsulation dot1q 100`	Enables the interface to accept 802.1Q VLAN packets.
Step 9	**exit** **Example:** `Router(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 10	**connect** *connection-name interface pvc interface* **interworking ip** | **ethernet** **Example:** `Router(config)# connect atm-eth-vlan-con atm1/0/0 0/100 fastethernet6/0/0.1 interworking ethernet`	In global configuration mode, creates a local connection between the two interfaces and specifies the interworking type. • Both the IP and Ethernet interworking types are supported.

Configuring Ethernet VLAN Same-Port Switching

Use the following steps 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 interface interface*

DETAILED STEPS

	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 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 that 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`	In global configuration mode, 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 this 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 interface interface* **Example:** `Router(config)# connect conn fastethernet6/0.1 fastethernet6/0.2`	In global configuration mode, 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

This section explains how 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 interface interface* [**interworking ip** | **ethernet**]

DETAILED STEPS

	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)# 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.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Router(config-subif)# exit`	Exits subinterface configuration mode and returns to global configuration mode.
Step 7	**connect** *connection-name interface interface* [**interworking ip** | **ethernet**] **Example:** `Router(config)# connect eth-ethvlan-con fastethernet3/0/0 fastethernet6/0/0.1 interworking ethernet`	Creates a local connection between the two interfaces and specifies the interworking type. • Both the IP and Ethernet interworking types are supported.

Configuring ATM-to-Frame Relay Local Switching

You use the **interworking ip** keywords for configuring ATM-to-Frame Relay local switching.

FRF.8 Frame Relay-to-ATM service interworking functionality is not supported. Frame Relay discard-eligible (DE) bits do not get mapped to ATM cell loss priority (CLP) bits, and forward explicit congestion notification (FECN) bits do not get mapped to ATM explicit forward congestion indication (EFCI) bits.

For additional information about ATM-to-Frame Relay Local Switching, see the "Configuring Frame Relay-ATM Interworking" section of the *Cisco IOS Wide Area Networking Configuration Guide*.

Creating the PVC is not required. If you do not create a PVC, one is created for you. For ATM-to-Ethernet local switching, the automatically provisioned PVC is given the default encapsulation type AAL5SNAP.

ATM-to-Frame Relay local switching supports the following encapsulation types:

- AAL5SNAP
- AAL5NLPID (Cisco 12000 series router uses AAL5MUX instead, for IP interworking)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface atm***slot*/*port*
4. **pvc** *vpi*/*vci* **l2transport**
5. **encapsulation** *layer-type*
6. **exit**
7. **interface serial***slot*/*port*
8. **encapsulation frame-relay** [**cisco** | **ietf**]
9. **frame-relay interface-dlci** *dlci* **switched**
10. **exit**
11. **connect** *connection-name interface pvc interface dlci* [**interworking ip** | **ethernet**]

DETAILED STEPS

	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`	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 1/200 l2transport`	Assigns a VPI and VCI and enters PVC l2transport configuration mode. • The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.
Step 5	**encapsulation** *layer-type* **Example:** `Router(cfg-if-atm-l2trans-pvc)# encapsulation aal5snap`	Specifies the encapsulation type for the PVC.
Step 6	**exit** **Example:** `Router(cfg-if-atm-l2trans-pvc)# exit`	Exits PVC l2transport configuration mode and returns to interface configuration mode.
Step 7	**interface serial***slot***/***subslot***/***port* **Example:** `Router(config-if)# interface serial6/0/0`	Specifies a channelized line card, subslot (if available), and serial port.
Step 8	**encapsulation frame-relay** [**cisco** | **ietf**] **Example:** `Router(config-if)# encapsulation frame-relay ietf`	Specifies Frame Relay encapsulation for the interface. • The encapsulation type does not matter for local switching. It has relevance only for terminated circuits.
Step 9	**frame-relay interface-dlci** *dlci* **switched** **Example:** `Router(config-if)# frame-relay interface-dlci 100 switched`	(Optional) Configures a switched Frame Relay DLCI. • If you do not create a Frame Relay PVC in this step, one is automatically created by the **connect** command.

	Command or Action	Purpose
Step 10	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 11	**connect** *connection-name interface pvc interface dlci* [**interworking ip** | **ethernet**] **Example:** `Router(config)# connect atm-fr-con atm1/0 0/100 serial6/0/0 100 interworking ip`	Creates a local connection between the two interfaces.

Configuring Frame Relay-to-Frame Relay Local Switching

For background information about Frame Relay-to-Frame Relay Local Switching, see the *Distributed Frame Relay Switching* feature module.

With Cisco IOS Release 12.0(30)S, you can switch between virtual circuits on the same port, as detailed in the "Configuring Frame Relay Same-Port Switching" section on page 22.

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

	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	**ip cef** [**distributed**] **Example:** `Router(config)# ip cef`	Enables Cisco Express Forwarding operation. • For the Cisco 7500 series router, use the **ip cef distributed** command. (On the Cisco 12000 series router, this command is already enabled by default). • For the Cisco 7200 series router, use the **ip cef** command.
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.
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.

Configuring Frame Relay Same-Port Switching

Use the following steps to configure Frame Relay switching on the same interface.

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 intf-type** [**dte** | **dce** | **nni**]
8. **frame-relay interface-dlci** *dlci* **switched**
9. **exit**
10. exit
11. **connect** *connection-name interface dlci interface dlci*

DETAILED STEPS

	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`	Enables Cisco Express Forwarding operation. • For the Cisco 7500 series router, use the **ip cef distributed** command. (On the Cisco 12000 series router, this command is already enabled by default). • For the Cisco 7200 series router, use the **ip cef** command.
Step 4	**frame-relay switching** **Example:** `Router(config)# frame-relay switching`	Enables PVC switching on a Frame Relay DCE device or a NNI.
Step 5	**interface** *type number* **Example:** `Router(config)# interface serial 0`	Specifies a Frame Relay interface and enters interface configuration mode.

	Command or Action	Purpose
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 intf-type** [**dce** | **dte** | **nni**] **Example:** `Router(config-if)# frame-relay intf-type nni`	(Optional) Enables support for a particular type of connection: • DCE • DTE (default) • NNI
Step 8	**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. • If you do not create a Frame Relay PVC in this step, it will automatically be created by the **connect** command.
Step 9	**exit** **Example:** `Router(config-fr-dlci)# exit`	Exits Frame Relay DLCI configuration mode and returns to interface configuration mode.

Step 10	`exit`
	
	Example:
	`Router(config-if)# exit`
		Exits interface configuration mode and returns to global configuration mode.

Step 11	**connect** *connection-name interface dlci interface dlci* **Example:** `Router(config)# connect connection1 serial1/0 100 serial1/0 200`	Defines a connection between the two data links.

Configuring HDLC Local Switching

This section explains how to configure local switching for HDLC. The PE routers are configured with HDLC encapsulation. The CE routers are configured with any HDLC-based encapsulation, including HDLC, PPP, and Frame Relay.

Restrictions

- Do not configure other settings on the interfaces configured for HDLC encapsulation. If you assign an IP address on the interface, the **connect** command is rejected and the following error message displays:

  ```
  Incompatible with IP address command on interface - command rejected.
  ```

  If you configure other settings on the interface that is enabled for HDLC encapsulation, the local switching feature may not work.
- Interworking is not supported.

- Same-port local switching for HDLC is not supported.
- Dialer and ISDN inerfaces are not supported. Only serial, HSSI, and POS interfaces can be configured for HDLC local switching.

Prerequisites

- Ensure that the interfaces you configure for HDLC encapsulation can handle ping packets that are smaller, the same size as, or larger than the CE interface MTU.
- Enable Cisco Express Forwarding.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip cef** [**distributed**]
4. **interface** *type number*
5. **exit**
6. **connect** *connection-name interface interface*

DETAILED STEPS

	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** **Example:** `Router(config)# ip cef`	Enables Cisco Express Forwarding operation.
Step 4	**interface** *type number* **Example:** `Router(config)# interface serial 2/0`	Specifies an interface and enters interface configuration mode.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**connect** *connection-name interface interface* **Example:** `Router(config)# connect connection1 serial1/0 serial1/0`	Defines a connection between the two interfaces.

Verifying Layer 2 Local Switching

This section provides the following verification tasks and troubleshooting information:

- Verifying Layer 2 Local Switching Configuration, page 25
- Verifying the NSF/SSO Local Switching Configuration, page 26
- Troubleshooting Tips, page 27

Verifying Layer 2 Local Switching Configuration

To verify configuration of the Layer 2 Local Switching feature, use the following commands on the provider edge (PE) router:

SUMMARY STEPS

1. **show connection** [**all** | *element* | **id** *ID* | **name** *name* | **port** *port*]
2. **show atm pvc**
3. **show frame-relay pvc** [*pvc*]

DETAILED STEPS

Step 1 **show connection** [**all** | *element* | **id** *ID* | **name** *name* | **port** *port*]

The **show connection** command displays the local connection between an ATM interface and a Fast Ethernet interface:

```
Router# show connection name atm-eth-con

ID  Name            Segment 1              Segment 2         State
=====================================================================
1   atm-eth-con     ATM0/0/0 AAL5 0/100    FastEthernet6/0/0  UP
```

This example displays the local connection between an ATM interface and a serial interface:

```
Router# show connection name atm-fr-con

ID  Name            Segment 1              Segment 2         State
=====================================================================
1   atm-fr-con     ATM0/0/0 AAL5 0/100    Serial1/0/0  16     UP
```

This example displays a same-port connection on a serial interface.

```
Router# show connection name same-port

ID  Name            Segment 1              Segment 2         State
=====================================================================
1   same-port      Serial1/1/1  101       Serial1/1/1  102    UP
```

Step 2 **show atm pvc**

The **show atm pvc** command shows that interface ATM3/0 is UP:

```
Router# show atm pvc

            VCD/                                          Peak   Avg/Min   Burst
Interface   Name   VPI   VCI   Type   Encaps      SC  Kbps   Kbps      Cells    Sts
3/0         10     1     32    PVC    FRATMSRV    UBR 155000                    UP
```

Step 3 **show frame-relay pvc** [*pvc*]

The **show frame-relay pvc** command shows a switched Frame Relay PVC:

```
Router# show frame-relay pvc 16

PVC Statistics for interface POS5/0 (Frame Relay NNI)
DLCI = 16, DLCI USAGE = SWITCHED, PVC STATUS = UP, INTERFACE = POS5/0
LOCAL PVC STATUS = UP, NNI PVC STATUS = ACTIVE
input pkts 0 output pkts 0 in bytes 0
out bytes 0 dropped pkts 100 in FECN pkts 0
in BECN pkts 0 out FECN pkts 0 out BECN pkts 0
in DE pkts 0 out DE pkts 0
out bcast pkts 0 out bcast bytes 0
switched pkts 0
Detailed packet drop counters:
no out intf 0 out intf down 100 no out PVC 0
in PVC down 0 out PVC down 0 pkt too big 0
pvc create time 00:25:32, last time pvc status changed 00:06:31
```

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 (port mode) to Ethernet VLAN
- Frame Relay to Frame Relay

For information about configuring NSF/SSO on the Route Processors, see the *Stateful Switchover* feature module. To verify that the NSF/SSO: Layer 2 Local Switching is working correctly, follow the steps in this section.

SUMMARY STEPS

1. **ping**
2. **redundancy force-switchover**
3. **show connect all**
4. **ping**

DETAILED STEPS

Step 1 Issue the **ping** command or initiate traffic between the two CE routers.

Step 2 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 Issue the **show connect all** command to ensure that the Layer 2 Local Switching connection on the dual RP is operating.

```
Router# show connect all

ID      Name          Segment 1     Segment 2     State
2      Eth-Vlan1      Fa1/1/1       Fa6/0/0/0.1   UP
```

Step 4 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 atm l2transport**
- **debug conn**
- **debug frame-relay pseudowire**
- **show frame-relay pvc**
- **show connection**
- **show atm pvc**

Configuration Examples for Layer 2 Local Switching

This section provides the following configuration examples:

- ATM-to-ATM Local Switching: Example, page 28
- ATM PVC Same-Port Switching: Example, page 28
- ATM PVP Same-Port Switching: Example, page 28
- ATM-to-Ethernet Local Switching: Examples, page 28
- Ethernet VLAN Same-Port Switching: Example, page 29
- ATM-to-Frame Relay Local Switching: Example, page 29
- Frame Relay-to-Frame Relay Local Switching: Example, page 29
- Frame Relay DLCI Same-Port Switching: Example, page 30
- HDLC Local Switching: Example, page 30
- NSF/SSO: Ethernet Port Mode to Ethernet VLAN Local Switching: Example

ATM-to-ATM Local Switching: Example

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
```

ATM PVC Same-Port Switching: Example

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
```

ATM PVP Same-Port Switching: Example

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
```

ATM-to-Ethernet Local Switching: Examples

ATM-to-Ethernet local switching terminates an ATM frame to an Ethernet/VLAN frame over the same PE router. Two interworking models are used: Ethernet mode and IP mode.

ATM to Ethernet VLAN: Example

The following example shows an Ethernet interface configured for Ethernet VLAN, and an ATM PVC interface configured for AAL5 encapsulation. The **connect** command allows local switching between these two interfaces and specifies the interworking type as Ethernet mode.

```
interface fastethernet6/0/0.1
 encapsulation dot1q 10

interface atm2/0/0
 pvc 0/400 l2transport
 encapsulation aal5

connect atm-ethvlan-con atm2/0/0 0/400 fastethernet6/0/0.1 interworking ethernet
```

ATM to Ethernet Port Mode: Example

The following example shows an Ethernet interface configured for Ethernet and an ATM interface configured for AAL5SNAP encapsulation. The **connect** command allows local switching between these two interfaces and specifies the interworking type as IP mode.

```
interface atm0/0/0
 pvc 0/100 l2transport
 encapsulation aal5snap

interface fastethernet6/0/0

connect atm-eth-con atm0/0/0 0/100 fastethernet6/0/0 interworking ip
```

Ethernet VLAN Same-Port Switching: Example

The following example shows same-port switching between two VLANs on one Ethernet interface:

```
interface fastethernet0/0.1
 encapsulation dot1q 1
interface fastethernet0/0.2
 encapsulation dot1q 2

connect conn FastEthernet0/0.1 FastEthernet0/0.2
```

ATM-to-Frame Relay Local Switching: Example

The following example shows a serial interface configured for Frame Relay and an ATM interface configured for AAL5SNAP encapsulation. The **connect** command allows local switching between these two interfaces.

```
interface serial1/0
 encapsulation frame-relay

interface atm1/0
 pvc 7/100 l2transport
 encapsulation aal5snap

connect atm-fr-conn atm1/0 7/100 serial1/0 100 interworking ip
```

Frame Relay-to-Frame Relay Local Switching: Example

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
```

Frame Relay DLCI Same-Port Switching: Example

The following example shows same-port switching between two data links on one Frame Relay interface:

```
interface serial1/0
 encapsulation frame-relay
 frame-relay int-type nni

connect conn serial1/0 100 serial1/0 200
```

HDLC Local Switching: Example

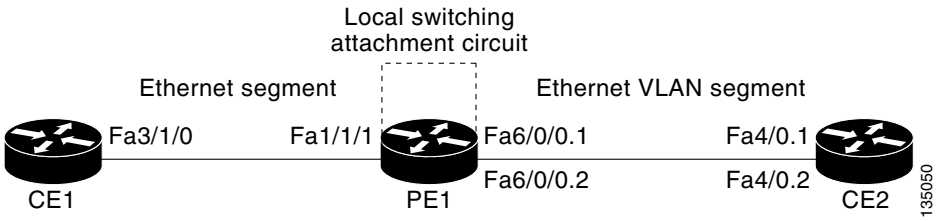
The following example shows local switching of two serial interfaces for HDLC:

```
interface serial1/0
 no ip address

interface serial2/0
 no ip address

connect conn1 serial1/0 serial1/0
```

NSF/SSO: Ethernet Port Mode to Ethernet VLAN Local Switching: Example

The following configuration uses the network topology shown in Figure 2.

Figure 2 NSF/SSO: Layer 2 Local Switching: Ethernet to Ethernet VLAN



The following example shows the configuration of the 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
!
hw-module slot 2 image disk0:rsp-pv-mz.shaft.111004
hw-module slot 3 image disk0:rsp-pv-mz.shaft.111004
!
ip routing
ip cef distributed
!
interface fa1/1/1
 description - connection to CE1 fa3/1/0
 no shutdown
 no ip address
!
interface fa4/0/0
 description - connection to CE3 fa6/0
 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
```

The following example shows the configuration of ICMP Router Discovery Protocol (IRDP) on the CE router for Interworking IP for ARP mediation:

CE1	CE2
interface FastEthernet3/1/0 ip irdp ip irdp maxadvertinterval 0	interface FastEthernet4/0.1 ip irdp ip irdp maxadvertinterval 0

The following example shows the configuration of OSPF on the CE routers:

CE1	CE2
interface loopback 1 ip address 10.11.11.11 255.255.255.255 ! router ospf 10 network 10.11.11.11 0.0.0.0 area 0 network 192.168.1.1 0.0.0.0 area 0	interface loopback 1 ip address 12.12.12.12 255.255.255.255 ! router ospf 10 network 10.12.12.12 0.0.0.0 area 0 network 192.168.1.2 0.0.0.0 area 0

The following example shows the configuration of local switching on the PE1 router for interworking Ethernet:

```
connect eth-vlan1 fa1/1/1 fa6/0/0.1 interworking ethernet
connect eth-vlan2 fa4/0/0 fa6/0/0.2 interworking ethernet
```

The following example shows the configuration of local switching on the PE1 router for interworking IP:

```
connect eth-vlan1 fa1/1/1 fa6/0/0.1 interworking ip
connect eth-vlan2 fa4/0/0 fa6/0/0.2 interworking ip
```

Additional References

The following sections provide references related to the Layer 2 Local Switching feature.

Related Documents

Related Topic	Document Title
MPLS	MPLS Product Literature

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

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Wide-Area Networking Command Reference* at http://www.cisco.com/en/US/docs/ios/wan/command/reference/wan_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **connect (L2VPN local switching)**
- **encapsulation (Layer 2 local switching)**
- **show connection**

Feature Information for Layer 2 Local Switching

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for Layer 2 Local Switching

Feature Name	Releases	Feature Information
Layer 2 Local Switching	12.0(27)S 12.2(25)S 12.0(30)S 12.0(31)S2 12.0(32)SY 12.2(28)SB 12.4(11)T 12.2(33)SRB 12.2(33)SXH 12.2(33)SB	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. The feature was introduced in Cisco IOS Release 12.0(27)S on the Cisco 7200 and 7500 series routers. The feature was integrated into Cisco IOS Release 12.2(25)S for the Cisco 7500 series router. In Cisco IOS Release 12.0(30)S, support for same-port switching was added. Support for Layer 2 interface-to-interface local switching was added on the Cisco 12000 series router. In Cisco IOS Release 12.0(31)S2, support was added for customer edge-facing IP Service Engine (ISE) interfaces on the Cisco 12000 series router. In Cisco IOS Release 12.0(32)SY, support was added for customer edge-facing interfaces on Engine 5 shared port adapters (SPAs) and SPA Interface Processors (SIPs) on the Cisco 12000 series router.

Table 1 Feature Information for Layer 2 Local Switching (Continued)

Feature Name	Releases	Feature Information
		In Cisco IOS Release 12.2(28)SB, this feature was updated to include NSF/SSO support on the Cisco 7500 series routers for the following local switching types on nonstop forwarding/stateful switchover (NSF/SSO): • NSF/SSO—Ethernet-to-Ethernet VLAN local switching support • NSF/SSO—Frame Relay-to-Frame Relay local switching support In Cisco IOS Release 12.4(11)T, support was added for the following local switching types for the Cisco 7200 series router: • Ethernet to Ethernet VLAN • Same-port switching for Ethernet VLAN • Frame Relay to Frame Relay • Same-port switching for Frame Relay In Cisco IOS Release 12.2(28)SB, supported was added for Local Switching on the Cisco 10000 series router. For information about Layer 2 Local Switching on the Cisco 10000 series routers, see the "Configuring Layer 2 Local Switching " section of the *Cisco 10000 Series Router Broadband Aggregation, Leased-Line, and MPLS Configuration Guide*. In Cisco IOS Release 12.2(33)SXH, support was added for like-to-like Local Switching (ATM to ATM, and FR to FR only) on Cisco 6500 series switches and Cisco 7600 series routers. Same-port switching is not supported on those routers. In Cisco IOS Release 12.2(33)SB, support was added for HDLC Local Switching on the Cisco 7200 series router and the Cisco 10000 series router. For information about the Layer 2 Local Switching feature on the Cisco 10000 series routers, see the "Configuring Layer 2 Local Switching" section of the *Cisco 10000 Series Router Broadband Aggregation, Leased-Line, and MPLS Configuration Guide*.

CCDE, CCVP, Cisco Eos, Cisco StadiumVision, the Cisco logo, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn is a service mark; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0801R)

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

coincidental.

© 2008 Cisco Systems, Inc. All rights reserved.

X.25 and LAPB

Configuring X.25 and LAPB

This chapter describes how to configure connections through Link Access Procedure, Balanced (LAPB) connections and X.25 networks. LAPB tasks are presented first for users who only want to configure a simple, reliable serial encapsulation method. This chapter contains the following sections:

- LAPB Overview
- LAPB Configuration Task List
- X.25 Configuration Task List

For further general information about X.25 and LAPB, see the chapter "Wide-Area Networking Overview" at the beginning of this book.

For a complete description of the commands mentioned in this chapter, refer to the chapter "X.25 and LAPB Commands" in the *Cisco IOS Wide-Area Networking Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the section "Identifying Supported Platforms" in the chapter "Using Cisco IOS Software."

For information on the following related topics, see the corresponding Cisco publications:

Task	Resource
Configuring PAD access	"Configuring the Cisco PAD Facility for X.25 Connections" chapter in the *Cisco IOS Terminal Services Configuration Guide*
Translating between an X.25 PAD connection and another protocol	*Cisco IOS Terminal Services Command Reference* (commands in alphabetical order.
Configuring X.25 traffic over an ISDN D channel	"Configuring X.25 on ISDN" and "Configuring X.25 on ISDN using Always On/Direct ISDN (AO/DI)" chapters in the *Cisco IOS Dial Technologies Configuration Guide*
Referencing a complete list of Dial commands	*Cisco IOS Dial Technologies Command Reference* (commands in alphabetical order)

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

LAPB Overview

You use LAPB as a serial encapsulation method only if you have a private serial line. You must use one of the X.25 packet-level encapsulations when attaching to an X.25 network.

LAPB standards distinguish between the following two types of hosts:

- Data terminal equipment (DTE)
- Data circuit-terminating equipment (DCE)

At Level 2 (data link layer) in the OSI model, LAPB allows orderly and reliable exchange of data between a DTE and a DCE device. A router using LAPB encapsulation can act as a DTE or DCE at the protocol level, which is distinct from the hardware DTE or DCE identity.

Using LAPB under heavy traffic conditions can result in greater throughput than is possible using High-Level Data Link Control (HDLC) encapsulation. When LAPB detects a missing frame, the router resends the frame instead of waiting for the higher layers to recover the lost information. This behavior is useful only if the host timers are relatively slow. In the case of quickly expiring host timers, however, LAPB spends much time sending host retransmissions. If the line is not busy with data traffic, HDLC encapsulation is more efficient than LAPB. When long-delay satellite links are used, for example, the lockstep behavior of LAPB makes HDLC encapsulation the better choice.

LAPB Configuration Task List

To configure LAPB, perform the tasks in the following sections:

- Configuring a LAPB Datagram Transport (Required)
- Configuring Compression of LAPB Data (Optional)
- Modifying LAPB Protocol Parameters (Optional)
- Configuring Priority and Custom Queueing for LAPB (Optional)
- Configuring Transparent Bridging over Multiprotocol LAPB (Optional)

To monitor and maintain LAPB, see the section "Monitoring and Maintaining LAPB and X.25" later in this chapter.

For an example of configuring LAPB operation, see the sections "Typical LAPB Configuration Example" and "Transparent Bridging for Multiprotocol LAPB Encapsulation Example" later in this chapter.

Configuring a LAPB Datagram Transport

To set the appropriate LAPB encapsulation to run datagrams over a serial interface, use the following command in global configuration mode. One end of the link must be a DTE device, and the other must be DCE. Because the default serial encapsulation is HDLC, you must explicitly configure a LAPB encapsulation method. You should shut down the interface before changing the encapsulation.

Command	Purpose
`Router(config)# interface type number`	Specifies a serial interface.

To select an encapsulation and protocol (if you are using a single protocol), or to select the multiple protocol operation, use one or more of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# encapsulation lapb dce [protocol]`[1]	Enables encapsulation of a single protocol on the line using DCE operation.
`Router(config-if)# encapsulation lapb dte [protocol]`[1]	Enables encapsulation of a single protocol on the line using DTE operation.
`Router(config-if)# encapsulation lapb dce multi`	Enables use of multiple protocols on the line using DCE operation.
`Router(config-if)# encapsulation lapb dte multi`[2]	Enable use of multiple protocols on the line using DTE operation.

1. Single protocol LAPB defaults to IP encapsulation.
2. Multiprotocol LAPB does not support source-route bridging or TCP/IP header compression, but does support transparent bridging. A multiprotocol LAPB encapsulation supports all of the protocols available to a single-protocol LAPB encapsulation plus transparent bridging.

For an example of configuring LAPB operation, see the section "Typical LAPB Configuration Example" later in this chapter.

Configuring Compression of LAPB Data

You can configure point-to-point software compression on serial interfaces that use a LAPB or multi-LAPB encapsulation. Compression reduces the size of a LAPB or multi-LAPB frame via lossless data compression. Compression is performed in the software and can substantially affect system performance. You should disable compression if the router CPU load exceeds 65 percent. To display the CPU load, use the **show process cpu** command.

Predictor compression is recommended when the bottleneck is caused by the load on the router or access server. Stacker compression is recommended when the bottleneck is the result of line bandwidth. Compression is not recommended if the majority of your traffic is already compressed files. Compression is also not recommended for line speeds greater than T1. The added processing time slows performance on fast lines.

To configure compression over LAPB, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# encapsulation lapb [protocol]`	Enables encapsulation of a single protocol on the serial line.
Step 2	`Router(config-if)# compress [predictor | stac]`	Enables compression.

To configure compression over multi-LAPB, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# encapsulation lapb multi`	Enables encapsulation of multiple protocols on the serial line.
Step 2	`Router(config-if)# compress [predictor | stac]`	Enables compression.

When using compression, adjust the maximum transmission unit (MTU) for the serial interface and the LAPB N1 parameter as in the following example, to avoid informational diagnostics regarding excessive MTU or N1 sizes:

```
interface serial 0
 encapsulation lapb
 compress predictor
 mtu 1509
 lapb n1 12072
```

For information about configuring X.25 TCP/IP header compression and X.25 payload compression, see the sections "Setting X.25 TCP/IP Header Compression" and "Configuring X.25 Payload Compression" later in this chapter.

Modifying LAPB Protocol Parameters

LAPB specifies methods for exchanging data (frames), detecting out-of-sequence or missing frames, retransmitting frames, and acknowledging frames. Several protocol parameters can be modified to change LAPB protocol performance on a particular link. Because X.25 operates the Packet Level Protocol (PLP) on top of the LAPB protocol, these tasks apply to both X.25 links and LAPB links. The parameters and their default values are summarized in Table 1. Detailed descriptions of each parameter are given after the table.

Table 1 LAPB Parameters

Command	Purpose (LAPB Parameter)	Values or Ranges	Default
lapb modulo *modulus*	Sets the modulo.	8 or 128	8
lapb k *window-size*	Sets the window size (K).	1– (modulo minus 1) frames	7
lapb n1 *bits*	Sets the maximum bits per frame (N1).	Bits (multiple of 8)	Based on hardware MTU and protocol overhead
lapb n2 *tries*	Sets the count for sending frames (N2).	1–255 tries	20
lapb t1 *milliseconds*	Sets the retransmission timer (T1).	1–64000 milliseconds	3000
lapb interface-outage *milliseconds*	Sets the hardware outage period.		0 (disabled)
lapb t4 *seconds*	Sets the idle link period (T4).		0 (disabled)

The following sections provide more information about the LAPB parameters in the Table 1:

- LAPB modulo—The LAPB modulo determines the operating mode. Modulo 8 (basic mode) is widely available because it is required for all standard LAPB implementations and is sufficient for most links. Modulo 128 (extended mode) can achieve greater throughput on high-speed links that have a low error rate (satellite links) by increasing the number of frames that can be sent before the sending device must wait for acknowledgment (as configured by LAPB parameter K).
- LAPB parameter K—LAPB K must be at most one less than the operating modulo. Modulo 8 links can send seven frames before an acknowledgment must be received by the sending device; modulo 128 links can send as many as 127 frames. By default, LAPB links use the basic mode with a window of 7.

- LAPB N1—When you configure a connection to an X.25 network, use the N1 parameter value set by the network administrator. This value is the maximum number of bits in a LAPB frame, which determines the maximum size of an X.25 packet. When you use LAPB over leased lines, the N1 parameter should be eight times the hardware MTU size plus any protocol overhead. The LAPB N1 range is dynamically calculated by the Cisco IOS software whenever an MTU change, a Layer 2/Layer 3 modulo change, or a compression change occurs on a LAPB interface.

Caution The LAPB N1 parameter provides little benefit beyond the interface MTU, and can easily cause link failures if misconfigured. Cisco recommends that you leave this parameter at its default value.

- LAPB N2—The transmit counter (N2) is the number of unsuccessful transmit attempts that are made before the link is declared down.
- LAPB T1—The retransmission timer (T1) determines how long a sent frame can remain unacknowledged before the Cisco IOS software polls for an acknowledgment. For X.25 networks, the retransmission timer setting should match that of the network.

  For leased-line circuits, the T1 timer setting is critical because the design of LAPB assumes that a frame has been lost if it is not acknowledged within period T1. The timer setting must be large enough to permit a maximum-sized frame to complete one round trip on the link. If the timer setting is too small, the software will poll before the acknowledgment frame can return, which may result in duplicated frames and severe protocol problems. If the timer setting is too large, the software waits longer than necessary before requesting an acknowledgment, slowing throughput.
- LAPB interface outage—Another LAPB timer function that allows brief hardware failures while the protocol is up, without requiring a protocol reset. When a brief hardware outage occurs, the link continues uninterrupted if the outage corrects before the specified outage period expires.
- LAPB T4—The LAPB standards define a timer to detect unsignaled link failures (T4). The T4 timer resets every time a frame is received from the partner on the link. If the T4 timer expires, a Receiver Ready frame with the Poll bit set is sent to the partner, which is required to respond. If the partner does not respond, the standard polling mechanism is used to determine whether the link is down. The period of T4 must be greater than the period of T1.

For an example of configuring the LAPB T1 timer, see the section "Typical LAPB Configuration Example" later in this chapter.

Configuring Priority and Custom Queueing for LAPB

LAPB uses priority and custom queueing, which improves the responsiveness of a link to a given type of traffic by specifying the handling of that type of traffic for transmission on the link.

Priority queueing is a mechanism that classifies packets based on certain criteria and then assigns packets to one of four output queues, with high, medium, normal, or low priority.

Custom queueing similarly classifies packets, assigns them to one of ten output queues, and controls the percentage of the available bandwidth of an interface that is used for a queue.

For example, you can use priority queueing to ensure that all Telnet traffic is processed promptly and that Simple Mail Transfer Protocol (SMTP) traffic is sent only when there is no other traffic to send. Priority queueing in this example can starve the non-Telnet traffic; custom queueing can be used instead to ensure that some traffic of all categories is sent.

Both priority and custom queueing can be defined, but only one can be assigned to a given interface. To configure priority and custom queueing for LAPB, perform these tasks in the following order:

1. Perform standard priority and custom queueing tasks *except* the task of assigning a priority or custom group to the interface, as described in the chapters"Configuring Prioirity Queueing" and "Configuring Custom Queueing" in the *Cisco IOS Quality of Service Solutions Configuration Guide*.
2. Perform standard LAPB encapsulation tasks, as specified in the section "Configuring a LAPB Datagram Transport" earlier in this chapter.
3. Assign either a priority group or a custom queue to the interface,as described in the chapters "Configuring Prioirity Queueing" and "Configuring Custom Queueing" in the *Cisco IOS Quality of Service Solutions Configuration Guide*.
4. The **lapb hold-queue** command is no longer supported, but the same functionality is provided by the standard queue control command **hold-queue** *size* **out**.

Configuring Transparent Bridging over Multiprotocol LAPB

To configure transparent bridging over multiprotocol LAPB, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface serial number`	Enters interface configuration mode.
Step 2	`Router(config-if)# no ip address`	Assigns no IP address to the interface.
Step 3	`Router(config-if)# encapsulation lapb multi`	Configures multiprotocol LAPB encapsulation.
Step 4	`Router(config-if)# bridge-group bridge-group`	Assigns the interface to a bridge group.
Step 5	`Router(config)# bridge bridge-group protocol {ieee | dec}`	Defines the type of Spanning-Tree Protocol.

Note You must use the **encapsulation lapb multi** command rather than the **encapsulation lapb** *protocol* **bridge** command to configure transparent bridging over multiprotocol LAPB.

For an example of configuring transparent bridging over multiprotocol LAPB, see the section "Transparent Bridging for Multiprotocol LAPB Encapsulation Example" later in this chapter.

X.25 Configuration Task List

To configure X.25, complete the tasks in the following sections. The interface, datagram transport, and routing tasks are divided into sections based on how common the feature is and how often it is used. Those features and parameters that are less common are found in the sections "Configuring Additional X.25 Interface Parameters," "Configuring Additional X.25 Datagram Transport Features," and "Configuring Additional X.25 Routing Features." LAPB frame parameters can be modified to optimize X.25 operation, as described earlier in this chapter. All these features can coexist on an X.25 interface.

- Configuring an X.25 Interface (Required)
- Configuring Additional X.25 Interface Parameters (Optional)

- Configuring an X.25 Datagram Transport (Optional)
- Configuring Additional X.25 Datagram Transport Features (Optional)
- Configuring X.25 Routing (Required)
- Configuring Additional X.25 Routing Features (Optional)
- Configuring DNS-Based X.25 Routing (Optional)
- Configuring X.25 over Frame Relay (Annex G) (Optional)
- Configuring CMNS Routing (Optional)
- Configuring Priority Queueing or Custom Queueing for X.25 (Optional)
- Configuring X.25 Closed User Groups (Optional)
- Configuring DDN or BFE X.25 (Optional)
- Configuring X.25 Remote Failure Detection (Optional)
- Creating X.29 Access Lists (Optional)
- Creating an X.29 Profile Script (Optional)
- Monitoring and Maintaining LAPB and X.25 (Optional)

Default parameters are provided for X.25 operation. However, you can change the settings to meet the needs of your X.25 network or as defined by your X.25 service supplier. Cisco also provides additional configuration settings to optimize your X.25 usage.

Note If you connect a router to an X.25 network, use the parameters set by your network administrator for the connection. These parameters will typically be those described in the sections "Configuring an X.25 Interface" and "Modifying LAPB Protocol Parameters" in this chapter. Also, note that the X.25 Level 2 parameters described in the section "Modifying LAPB Protocol Parameters" affect X.25 Level 3 operations.

For examples of configuring X.25, see the "X.25 and LAPB Configuration Examples" section later in this chapter.

Configuring an X.25 Interface

The following tasks describe essential parameters for correct X.25 behavior. To configure an X.25 interface, perform the tasks in the following sections. The first task is required, the others might be required or optional, depending on what the router is expected to do with the X.25 attachment.

- Configuring X.25 Encapsulation (Required)
- Setting the Virtual Circuit Ranges (Required/Optional)
- Setting the Packet-Numbering Modulo (Required/Optional)
- Setting the X.121 Address (Required/Optional)
- Configuring X.25 Switch Local Acknowledgment (Required/Optional)
- Enabling Flow Control Parameter Negotiation (Required/Optional)
- Setting Default Flow Control Values (Required/Optional)
- Enabling Asymmetrical Flow Control (Required/Optional)

You can also configure less common parameters as specified in the section "Configuring Additional X.25 Interface Parameters."

Configuring X.25 Encapsulation

A router using X.25 Level 3 encapsulation can act as a DTE or DCE protocol device (according to the needs of your X.25 service supplier), can use DDN or BFE encapsulation, or can use the Internet Engineering Task Force (IETF) standard encapsulation, as specified by RFC 1356.

Because the default serial encapsulation is HDLC, you must explicitly configure an X.25 encapsulation method.

Note We recommend that you use the **no encapsulation x25** command to remove all X.25 configurations from the interface before changing the encapsulation.

To configure the mode of operation and encapsulation type for a specified interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# encapsulation x25 [dte | dce] [[ddn | bfe] | [ietf]]`	Sets the X.25 mode of operation.

Typically a public data network (PDN) will require attachment as a DTE device. (This requirement is distinct from the hardware interface DTE or DCE identity.) The default mode is DTE, and the default encapsulation method is the Cisco pre-IETF method. If either DDN or BFE operation is needed, it must be explicitly configured. For an example of configuring X.25 DTE operation, see the section "Typical X.25 Configuration Example" later in this chapter.

Setting the Virtual Circuit Ranges

X.25 maintains multiple connections—virtual circuits (VCs) or logical circuits (LCs)—over one physical link between a DTE and a DCE device. X.25 can maintain up to 4095 VCs. A VC is identified by its logical channel identifier (LCI) or virtual circuit number (VCN).

Note Many documents use the terms *virtual circuit* and *LC*, *VCN*, *LCN*, and *LCI* interchangeably. Each of these terms refers to the VC number.

An important part of X.25 operation is the range of VC numbers. These numbers are broken into the following four ranges:

1. Permanent virtual circuits (PVCs)
2. Incoming-only circuits
3. Two-way circuits
4. Outgoing-only circuits

The incoming-only, two-way, and outgoing-only ranges define the VC numbers over which a switched virtual circuit (SVC) can be established by the placement of an X.25 call, much as a telephone network establishes a switched voice circuit when a call is placed.

The rules about DCE and DTE devices initiating calls are as follows:

- Only the DCE can initiate a call in the incoming-only range.
- Only the DTE can initiate a call in the outgoing-only range.
- Both the DCE and DTE can initiate a call in the two-way range.

Note The International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) functions in place of the former Consultative Committee for International Telegraph and Telephone (CCITT). ITU-T *Recommendation X.25* defines "incoming" and "outgoing" in relation to the DTE or DCE interface role. Cisco documentation uses the more intuitive sense. Unless the ITU-T sense is explicitly referenced, a call received from the interface is an *incoming call* and a call sent out to the interface is an *outgoing call*.

There is no difference in the operation of SVCs in the different ranges except the restrictions on which device can initiate a call. These ranges can be used to prevent one side from monopolizing the VCs, which is important for X.25 interfaces with a small number of SVCs available. Six X.25 parameters define the upper and lower limit of each of the three SVC ranges. These ranges cannot overlap. A PVC must be assigned a number lower than those assigned to the SVC ranges.

Note Because X.25 requires the DTE and DCE devices to have identical VC ranges, changes you make to the VC range limits when the interface is up are held until X.25 restarts the packet service.

To configure X.25 VC ranges, use the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 lic circuit-number`	Sets the lowest incoming-only circuit number. The default is 0.
`Router(config-if)# x25 hic circuit-number`	Sets the highest incoming-only circuit number. The default is 0
`Router(config-if)# x25 ltc circuit-number`	Sets the lowest two-way circuit number. The default is 1.
`Router(config-if)# x25 htc circuit-number`	Sets the highest two-way circuit number. The default is 1024 for X.25; 4095 for CMNS.
`Router(config-if)# x25 loc circuit-number`	Sets the lowest outgoing-only circuit number. The default is 0.
`Router(config-if)# x25 hoc circuit-number`	Sets the highest outgoing-only circuit number. The default is 0.

Each of these parameters can range from 1 to 4095. The values for these parameters must be the same on both ends of the X.25 link. For connection to a PDN, these values must be set to the values assigned by the network. An SVC range is unused if its lower and upper limits are set to 0; other than this use for marking unused ranges, VC 0 is not available. For an example of configuring VC ranges, see the "VC Ranges Example" section later in this chapter.

Setting the Packet-Numbering Modulo

The Cisco implementation of X.25 supports modulo 8 (default) and modulo 128 packet sequence numbering.

To set the packet-numbering modulo, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 modulo {8 | 128}`	Sets the packet-numbering modulo.

Note Because X.25 requires the DTE and DCE devices to have identical modulos, changes you make to the modulo when the interface is up remain until X.25 restarts the packet service.

The X.25 modulo and the LAPB modulo are distinct and serve different purposes. LAPB modulo 128 (or extended mode) can be used to achieve higher throughput across the DTE or DCE interface, which affects only the local point of attachment. X.25 PLP modulo 128 can be used to achieve higher end-to-end throughput for VCs by allowing more data packets to be in transit across the X.25 network.

Setting the X.121 Address

If your router does not originate or terminate calls but only participates in X.25 switching, this task is optional. However, if your router is attached to a PDN, you must set the interface X.121 address assigned by the X.25 network service provider. Interfaces that use the DDN or BFE mode will have an X.121 address generated from the interface IP address; for correct DDN or BFE operation, any such X.121 address must not be modified.

To set the X.121 address, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 address x121-address`	Sets the X.121 address.

For an example of configuring the X.25 interface address, see the section "Typical X.25 Configuration Example" later in this chapter.

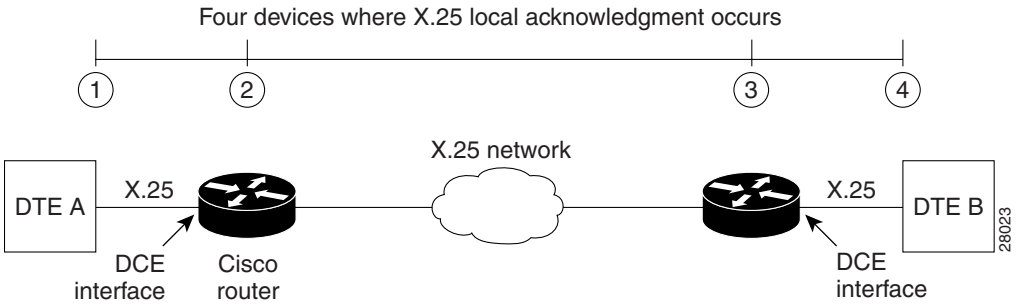
Configuring X.25 Switch Local Acknowledgment

X.25 switch local acknowledgment allows you the choice of configuring local or end-to-end acknowledgment on your router. End-to-end acknowledgment can result in lower overall throughput and restrictive performance because an endpoint can only have a limited number of its packets in transit at any given time. End-to-end acknowledgment cannot send more packets until all have been acknowledged by the transmission and receipt of the delivery-confirming packet containing the D-bit.

Local acknowledgment means that the Cisco router can send acknowledgments for packets that do not have the D-bit set, before receiving an acknowledgment from the interface to which the packet was forwarded. This results in higher throughput of packets because acknowledgment is sent between local hops much faster and more efficiently than between end-to-end hops.

Figure 1 shows the Cisco router receiving packets from DTE A destined for DTE B. Without local acknowledgment enabled, the router forwards packets to the X.25 network and then forwards acknowledgments from the network back to DTE A. With local acknowledgment enabled, the router can acknowledge packets received from DTE A before it has received acknowledgments from the network for the forwarded packets. In this illustration, the X.25 network may also generate local acknowledgments.

Figure 1 Local Acknowledgment Between DTE A and DTE B



To configure local acknowledgment, use the following command in global configuration mode:

Command	Purpose
`Router(config)# x25 routing acknowledge local`	Enables X.25 switching with local acknowledgment.

For an example of configuring local acknowledgment, see the section "Configuring Local Acknowledgment Example" later in this chapter, and for verification see the section "Verifying Local Acknowledgment," next.

Verifying Local Acknowledgment

To verify that local acknowledgment is configured on your router, use the **show running-configuration** command in EXEC mode. In the following example, X.25 encapsulation has been set on serial interface 1/4 with acknowledgment set to "local":

```
Router# show running-configuration

x25 routing acknowledge local
```

You can also use the **show protocol** command in EXEC mode to verify local acknowledgment:

```
Router# show protocol
Global values:
   Internet Protocol routing is enabled
   X.25 routing is enabled, acknowledgements have local significance only
```

Enabling Flow Control Parameter Negotiation

Flow control is an X.25 optional user facility. When the **x25 subscribe flow-control** command is used, it permits flow control parameter negotiation of packet sizes and window sizes. This command can be altered to one of three states: default behavior (**no x25 subscribe flow-control**), facilities **always** included, or facilities **never** included (flow control parameter negotiation is not enabled). By default, these flow control parameter negotiation facilities are included in call setup (outgoing) packets only when their values differ from the default values.

When flow control parameter negotiation is enabled, the **x25 subscribe windowsize** and **x25 subscribe packetsize** commands allow you to configure flow control restrictions by specifying window size and packet size ranges for permitted and target values. A value that cannot be negotiated into the permitted range is treated as illegal, causing the call to fail. The router first attempts values within the target range,

but allows values outside the target range to be considered as long as the range complies with procedures defined in the ITU-T *Recommendation X.25*. With this feature, the Cisco router allows different flow control value configurations and acceptable window and packet size formats for both DTE devices.

The ability to disable flow control parameter negotiation provides compatibility with equipment that does not support flow control parameter negotiation. Similarly, forcing flow control parameter negotiation provides compatibility with devices that require the flow control parameter negotiation facilities to be present in all calls.

To control packet transmission flow values on the interface, use one or more of the flow control commands—**x25 subscribe flow-control**, **x25 subscribe windowsize**, or **x25 subscribe packetsize**—in interface configuration mode.

Command	Purpose
`Router(config-if)# x25 subscribe flow-control {always | never}`	Determines flow control parameter negotiation behavior.
`Router(config-if)# x25 subscribe windowsize {permit wmin wmax | target wmin wmax}`	Sets permitted and target ranges for window size negotiation.
`Router(config-if)# x25 subscribe packetsize {permit pmin pmax | target pmin pmax}`	Sets permitted and target ranges for packet size negotiation.

The flow control subscription commands may be applied to an X.25 interface, to an X.25 profile, or to a LAN interface on which the **cmns enable** command has been configured. For X.25 over TCP (XOT), the flow control parameter negotiation facilities are always included (the equivalent of **x25 subscribe flow-control always**).

For an example of setting flow control parameter negotiation, see the sections "Setting Asymmetrical Window and Packet Sizes Flow Control Never Example" and "Configuring Flow Control Always Example" later in this chapter, and for verification see the following section, "Verifying Flow Control Parameter Negotiation."

Verifying Flow Control Parameter Negotiation

To verify flow control parameter settings, use the **show running-configuration** command in EXEC mode. In the following example, X.25 encapsulation has been set on serial interface 1/4 with flow control negotiation set to "always." Permitted packet sizes are set at 64 (minimum) and 1024 (maximum), with target packet sizes set at 128 (minimum) and 1024 (maximum). Permitted window sizes are set at 1 (minimum) and 7 (maximum), with target window sizes set at 2 (minimum) and 4 (maximum).

```
Router# show running-configuration

 x25 subscribe flow-control always
 x25 subscribe packetsize permit 64 1024 target 128 1024
 x25 subscribe windowsize permit 1 7 target 2 4
```

Setting Default Flow Control Values

Setting correct default flow control parameters of window size and packet size is essential for correct operation of the link because X.25 is a strongly flow controlled protocol. However, it is easy to overlook this task because many networks use standard default values. Mismatched default flow control values will cause X.25 local procedure errors, evidenced by Clear and Reset events.

To configure flow control parameters, complete the tasks in the following sections. These tasks are optional if your X.25 attachment uses the standard default values for maximum packet sizes (128 bytes incoming and outgoing) and window sizes (2 packets incoming and outgoing).

- Setting Default Window Sizes
- Setting Default Packet Sizes

Note Because X.25 requires the DTE and DCE devices to have identical default maximum packet sizes and default window sizes, changes made to the window and packet sizes when the interface is up are held until X.25 restarts the packet service.

Setting Default Window Sizes

X.25 networks have a default input and output window size (the default is 2) that is defined by your network administrator. You must set the Cisco IOS software default input and output window sizes to match those of the network. These defaults are the values that an SVC takes on if it is set up without explicitly negotiating its window sizes. Any PVC also uses these default values unless different values are configured.

To set the default window sizes, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# x25 win packets`	Sets input maximum window size.
Step 2	`Router(config-if)# x25 wout packets`	Sets output maximum window size.

For an example of setting the default window sizes, see the sections "Typical X.25 Configuration Example" and "DDN X.25 Configuration Example" later in this chapter.

Setting Default Packet Sizes

X.25 networks have a default maximum input and output packet size (the default is 128) that is defined by your network administrator. You must set the Cisco IOS software default input and output maximum packet sizes to match those of the network. These defaults are the values that an SVC takes on if it is set up without explicit negotiatiation of its maximum packet sizes. Any PVC also uses these default values unless different values are configured.

To set the default input and output maximum packet sizes, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# x25 ips bytes`	Sets input maximum packet size.
Step 2	`Router(config-if)# x25 ops bytes`	Sets output maximum packet size.

To send a packet larger than the agreed-on X.25 packet size over an X.25 VC, the Cisco IOS software must break the packet into two or more X.25 packets with the M-bit ("more data" bit) set. The receiving device collects all packets in the M-bit sequence and reassembles them into the original packet.

It is possible to define default packet sizes that cannot be supported by the lower layer (see the LAPB N1 parameter). However, the router will negotiate lower maximum packet sizes for all SVCs so the agreed-on sizes can be carried. The Cisco IOS software will also refuse a PVC configuration if the resulting maximum packet sizes cannot be supported by the lower layer.

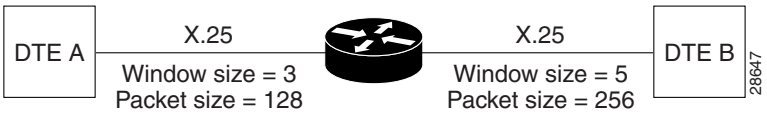
For an example of setting the default maximum packet sizes, see the sections "Typical X.25 Configuration Example" and "DDN X.25 Configuration Example" later in this chapter.

Enabling Asymmetrical Flow Control

Asymmetrical flow control is supported by the permitted configuration of asymmetrical window and packet sizes. For data flow from a channel with a smaller packet size than its outbound channel, the switch may combine data packets, and for a channel with a larger packet size than its outbound channel, the switch will fragment the packets.

Figure 2 shows asymmetrical configuration of the Cisco router. DTE A (window size 3; packet size 128) and DTE B (window size 5; packet size 256) are able to communicate despite differing window and packet sizes.

Figure 2 Asymmetrical Window and Packet Sizes Between DTE A and DTE B



To use asymmetrical flow control effectively, use the **x25 subscribe flow-control never** command to disable flow control parameter negotiation, and use the **x25 routing acknowledge local** command to enable local acknowledgment.

	Command	Purpose
Step 1	`Router(config)# x25 routing acknowledge local`	Enables X.25 switching with local acknowledgment.
Step 2	`Router(config-if)# x25 subscribe flow-control never`	Disables flow control parameter negotiation behavior.

For an example of enabling asymmetrical flow control, see the section "Setting Asymmetrical Window and Packet Sizes Flow Control Never Example" later in this chapter.

Configuring Additional X.25 Interface Parameters

Some X.25 applications have unusual or special needs. Several X.25 parameters are available to modify X.25 behavior for these applications.

To configure X.25 interface parameters for these special needs, perform the tasks in the following sections, as needed:

- Configuring X.25 Failover
- Configuring the X.25 Level 3 Timers
- Configuring X.25 Addresses
- Establishing a Default VC Protocol

- Disabling PLP Restarts

Configuring X.25 Failover

Multiple routes can be configured in an X.25 routing table to allow one or more secondary or backup interfaces to be used when a preferred (primary) interface is not usable. Routes are examined in the order in which they appear in the X.25 routing table, and the first matching route is taken. However, since X.25 traffic is circuit-oriented, once a connection is established via the secondary interface, the connection remains active even after the primary interface returns to service. This situation is undesirable when the path via the secondary interface is slower or more expensive than the path via the primary interface.

X.25 Failover enables you to configure the secondary or backup interface to reset once the primary interface has come back up and remained operational for a specified amount of time, terminating any connections that are still using the secondary interface. Subsequent calls will then be forwarded over the preferred interface.

X.25 Failover supports Annex G (X.25 over Frame Relay), but it does not support XOT.

You can configure X.25 Failover on an X.25 interface or X.25 profile.

Configuring X.25 Failover on an Interface

To configure X.25 failover on an interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Configures an interface type and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation x25`	Specifies the operation of a serial interface as an X.25 device.
Step 3	`Router(config-if)# x25 fail-over seconds interface type number [dlci | MAC address]`	Specifies a secondary interface and sets the number of seconds for which the primary interface must be up before the secondary interface resets.

For an example X.25 failover configuration, see the section "X.25 Failover Example" later in this chapter.

Configuring X.25 Failover on an X.25 Profile

To configure X.25 failover on an X.25 profile, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# x25 profile name {dce | dte | dxe}`	Configures an X.25 profile.
Step 2	`Router(config-x25)# x25 fail-over seconds interface type number [dlci | MAC address]`	Specifies a secondary interface and sets the number of seconds for which the primary interface must be up before the secondary interface resets.

Verifying X.25 Failover

To display information about the X.25 Failover feature, use the following EXEC command:

Command	Purpose
`Router# show x25 context`	Displays information about all X.25 links.

Configuring the X.25 Level 3 Timers

The X.25 Level 3 event timers determine how long the Cisco IOS software waits for acknowledgment of control packets. You can set these timers independently. Only those timers that apply to the interface are configurable. (A DTE interface does not have the T1*x* timers, and a DCE interface does not have the T2*x* timers.)

To set the event timers, use any of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 t20` *seconds*	Sets DTE T20 Restart Request timeout.
`Router(config-if)# x25 t10` *seconds*	Sets DCE T10 Restart Indication timeout.
`Router(config-if)# x25 t21` *seconds*	Sets DTE T21 Call Request timeout.
`Router(config-if)# x25 t11` *seconds*	Sets DCE T11 Incoming Call timeout.
`Router(config-if)# x25 t22` *seconds*	Sets DTE T22 Reset Request timeout.
`Router(config-if)# x25 t12` *seconds*	Sets DCE T12 Reset Indication timeout.
`Router(config-if)# x25 t23` *seconds*	Sets DTE T23 Clear Request timeout.
`Router(config-if)# x25 t13` *seconds*	Sets DCE T13 Clear Indication timeout.

For an example of setting the event timers, see the section "DDN X.25 Configuration Example" later in this chapter.

Configuring X.25 Addresses

When you establish SVCs, X.25 uses addresses in the form defined by ITU-T *Recommendation X.121* (or simply an "X.121 address"). An X.121 address has from zero to 15 digits. Because of the importance of addressing to call setup, several interface addressing features are available for X.25.

The X.121 address of an X.25 interface is used when it is the source or destination of an X.25 call. The X.25 call setup procedure identifies both the calling (source) and the called (destination) X.121 addresses. When an interface is the source of a call, it encodes the interface X.121 address as the source address. An interface determines that it is the destination of a received call if the destination address matches the address of the interface.

Cisco IOS X.25 software can also route X.25 calls, which involves placing and accepting calls, but the router is neither the source nor the destination for these calls. Routing X.25 does not modify the source or destination addresses, thus preserving the addresses specified by the source host. Routed (switched) X.25 simply connects two logical X.25 channels to complete an X.25 VC. An X.25 VC, then, is a connection between two hosts (the source host and the destination host) that is switched between zero or more routed X.25 links.

The null X.121 address (the X.121 address that has zero digits) is a special case. The router acts as the destination host for any call it receives that has the null destination address.

A subaddress is an X.121 address that matches the digits defined for the X.121 address of the interface, but has one or more additional digits after the base address. X.25 acts as the destination host for an incoming PAD call with a destination that is a subaddress of the address of the interface; the trailing digits specify which line a PAD connection is requesting. This feature is described in the chapter "Configuring Protocol Translation and Virtual Asynchronous Devices" in the *Cisco IOS Terminal Services Configuration Guide*. Other calls that use a subaddress can be accepted if the trailing digit or digits are zeros; otherwise, the router will not act as the destination host of the call.

To configure X.25 addresses, perform the tasks in the following sections:

- Configuring an Interface Alias Address
- Suppressing or Replacing the Calling Address
- Suppressing the Called Address

Configuring an Interface Alias Address

You can supply alias X.121 addresses for an interface. Supplying alias addresses allows the interface to act as the destination host for calls having a destination address that is neither the address of the interface, an allowed subaddress of the interface, nor the null address.

Local processing (for example, IP encapsulation) can be performed only for incoming calls whose destination X.121 address matches the serial interface or alias of the interface.

To configure an alias, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 alias` *x121-address-pattern* `[cud` *pattern*`]`	Enables an alias X.121 address for the interface.

Suppressing or Replacing the Calling Address

Some attachments require that no calling (source) address be presented in outgoing calls. This requirement is called *suppressing the calling address*. When attached to a PDN, X.25 may need to ensure that outgoing calls use only the assigned X.121 address for the calling (source) address. Routed X.25 normally uses the original source address. Although individual X.25 route configurations can modify the source address, Cisco provides a simple command to force the use of the interface address in all calls sent; this requirement is called *replacing the calling address*.

To suppress or replace the calling address, use the appropriate command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 suppress-calling-address`	Suppresses the calling (source) X.121 address in outgoing calls.
`Router(config-if)# x25 use-source-address`	Replaces the calling (source) X.121 address in switched calls.

Suppressing the Called Address

Some attachments require that no called (destination) address be presented in outgoing calls; this requirement is called *suppressing the called address*.

To suppress the called address, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 suppress-called-address`	Suppresses the called (destination) X.121 address in outgoing calls.

Establishing a Default VC Protocol

The Call Request packet that sets up a VC can encode a field called the Call User Data (CUD) field. Typically the first few bytes of the CUD field identify which high-level protocol is carried by the VC. The router, when acting as a destination host, normally refuses a call if the CUD is absent or the protocol identification is not recognized. The PAD protocol, however, specifies that unidentified calls be treated as PAD connection requests. Other applications require that they be treated as IP encapsulation connection requests, in accordance with RFC 877, *A Standard for the Transmission of IP Datagrams over Public Data Networks*.

To configure either PAD or IP encapsulation treatment of unidentified calls, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 default {ip | pad}`	Establishes a default VC protocol.

Disabling PLP Restarts

By default, a PLP restart is performed when the link level resets (for example, when LAPB reconnects). Although PLP restarts can be disabled for those few networks that do not allow restarts, we do not recommend disabling these restarts because doing so can cause anomalous packet layer behavior.

Caution Very few networks require this feature. We do not recommend that it be enabled except when you are attaching to a network that requires it.

To disable PLP restarts, use the following command in interface configuration mode:

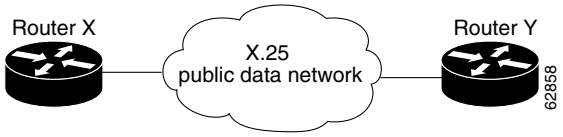
Command	Purpose
`Router(config-if)# no x25 linkrestart`	Disables packet-level restarts.

Configuring an X.25 Datagram Transport

X.25 support is most commonly configured as a transport for datagrams across an X.25 network. Datagram transport (or encapsulation) is a cooperative effort between two hosts communicating across an X.25 network. You configure datagram transport by establishing a mapping on the encapsulating interface between the protocol address of the far host (for example, IP or DECnet) and its X.121 address. Because the call identifies the protocol that the VC will carry (by encoding a Protocol Identifier, or PID, in the first few bytes of the CUD field), the terminating host can accept the call if it is configured to exchange the identified traffic with the source host.

Figure 3 illustrates two routers sending datagrams across an X.25 PDN.

Figure 3 Transporting LAN Protocols Across an X.25 PDN



To complete the X.25 configuration for your network needs, perform the tasks in the following sections:

- Configuring Point-to-Point and Multipoint Subinterfaces
- Mapping Protocol Addresses to X.121 Addresses
- Establishing an Encapsulation PVC
- Setting X.25 TCP/IP Header Compression
- Configuring X.25 Bridging

Configuring the X.25 parameters and special features, including payload compression and X.25 user facilities, is described in the section "Configuring Additional X.25 Datagram Transport Features" later in this chapter.

Configuring Point-to-Point and Multipoint Subinterfaces

Subinterfaces are virtual interfaces that can be used to connect several networks to each other through a single physical interface. Subinterfaces are made available on Cisco routers because routing protocols, especially those using the split horizon principle, may need help to determine which hosts need a routing update. The split horizon principle, which allows routing updates to be distributed to other routed interfaces except the interface on which the routing update was received, works well in a LAN environment in which other routers reached by the interface have already received the routing update.

However, in a WAN environment using connection-oriented interfaces (like X.25 and Frame Relay), other routers reached by the same physical interface might not have received the routing update. Rather than forcing you to connect routers by separate physical interfaces, Cisco provides subinterfaces that are treated as separate interfaces. You can separate hosts into subinterfaces on a physical interface, X.25 is unaffected, and routing processes recognize each subinterface as a separate source of routing updates, so all subinterfaces are eligible to receive routing updates.

There are two types of subinterfaces: point-to-point and multipoint. Subinterfaces are implicitly multipoint unless configured as point-to-point.

A point-to-point subinterface is used to encapsulate one or more protocols between two hosts. An X.25 point-to-point subinterface will accept only a single encapsulation command (such as the **x25 map** or **x25 pvc** command) for a given protocol, so there can be only one destination for the protocol. (However, you can use multiple encapsulation commands, one for each protocol, or multiple protocols for one map or PVC.) All protocol traffic routed to a point-to-point subinterface is forwarded to the one destination host defined for the protocol. (Because only one destination is defined for the interface, the routing process need not consult the destination address in the datagrams.)

A multipoint subinterface is used to connect one or more hosts for a given protocol. There is no restriction on the number of encapsulation commands that can be configured on a multipoint subinterface. Because the hosts appear on the same subinterface, they are not relying on the router to distribute routing updates among them. When a routing process forwards a datagram to a multipoint

subinterface, the X.25 encapsulation process must be able to map the destination address of the datagram to a configured encapsulation command. If the routing process cannot find a map for the datagram destination address, the encapsulation will fail.

Note Because of the complex operations dependent on a subinterface and its type, the router will not allow a subinterface's type to be changed, nor can a subinterface with the same number be reestablished once it has been deleted. After a subinterface has been deleted, you must reload the Cisco IOS software (by using the **reload** command) to remove all internal references. However, you can easily reconstitute the deleted subinterface by using a different subinterface number.

To configure subinterfaces on your X.25 network, perform the tasks in the section "Creating and Configuring X.25 Subinterfaces," next.

Creating and Configuring X.25 Subinterfaces

To create and configure a subinterface, use the Step 1 command and one or both of the Step 2 commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface serial type number.subinterface-number [point-to-point | multipoint]`	Creates a point-to-point or multipoint subinterface.

Step 2	`Router(config-subif)# x25 map protocol address [protocol2 address2 [... [protocol9 address9]]] x121-address [option]`
	
	and/or
	
	`Router(config-subif)# x25 pvc circuit protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address [option]`
		Configures an X.25 encapsulation map for the subinterface.
		
		Establishes an encapsulation PVC for the subinterface.

For an example of configuring an X.25 subinterface and using multiple encapsulation commands for a single destination address, see the section "Point-to-Point Subinterface Configuration Example" later in this chapter.

For more general information about configuring subinterfaces, refer to the chapter "Configuring Serial Interfaces" in the *Cisco IOS Interface Configuration Guide*.

When configuring IP routing over X.25, you might need to make adjustments to accommodate split horizon effects. Refer to the chapter "Configuring RIP" in the *Cisco IOS IP Configuration Guide* for details about possible split horizon conflicts. By default, split horizon is enabled for X.25 attachments.

Mapping Protocol Addresses to X.121 Addresses

This section describes the X.25 single-protocol and multiprotocol encapsulation options that are available and describes how to map protocol addresses to an X.121 address for a remote host. The following sections include reference information about how protocols are identified:

- Understanding Protocol Encapsulation for Single-Protocol and Multiprotocol VCs
- Understanding Protocol Identification

Perform the mapping tasks in the following sections, as necessary:

- Mapping Datagram Addresses to X.25 Hosts
- Configuring PAD Access

Understanding Protocol Encapsulation for Single-Protocol and Multiprotocol VCs

Cisco has long supported encapsulation of a number of datagram protocols across X.25, using a standard method when available or a proprietary method when necessary. These traditional methods assign a protocol to each VC. If more than one protocol is carried between the router and a given host, each active protocol will have at least one VC dedicated to carrying its datagrams.

Cisco also supports a newer standard, RFC 1356, *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode*,which standardizes a method for encapsulating most datagram protocols over X.25. It also specifies how one VC can carry datagrams from more than one protocol.

The Cisco IOS software can be configured to use any of the available encapsulation methods with a particular host.

After you establish an encapsulation VC using any method, the Cisco IOS software sends and receives a datagram by simply fragmenting it into and reassembling it from an X.25 complete packet sequence. An X.25 complete packet sequence is one or more X.25 data packets that have the M-bit set in all but the last packet. A VC that can carry multiple protocols includes protocol identification data as well as the protocol data at the start of each complete packet sequence.

Understanding Protocol Identification

This section contains background material only.

The various methods and protocols used in X.25 SVC encapsulation are identified in a specific field of the call packet; this field is defined by X.25 to carry CUD. Only PVCs do not use CUD to identify their encapsulation (because PVCs do not use the X.25 call setup procedures).

The primary difference between the available Cisco and IETF encapsulation methods is the specific value used to identify a protocol. When any of the methods establishes a VC for carrying a single protocol, the protocol is identified in the call packet by the CUD.

Table 2 summarizes the values used in the CUD field to identify protocols.

Table 2 Protocol Identification in the CUD Field

Protocol	Cisco Protocol Identifier	IETF RFC 1356 Protocol Identifier
Apollo Domain	0xD4	0x80 (5-byte SNAP encoding)[1]
AppleTalk	0xD2	0x80 (5-byte SNAP encoding)
Banyan VINES	0xC0 00 80 C4[2]	0x80 (5-byte SNAP encoding)
Bridging	0xD5	Not implemented
ISO CLNS	0x81	0x81[3]
Compressed TCP	0xD8	0x00 (multiprotocol)[4]
DECnet	0xD0	0x80 (5-byte SNAP encoding)
IP	0xCC	0xCC[5] or 0x80 (5-byte SNAP encoding)
Novell IPX	0xD3	0x80 (5-byte SNAP encoding)

Table 2 Protocol Identification in the CUD Field (Continued)

Protocol	Cisco Protocol Identifier	IETF RFC 1356 Protocol Identifier
PAD	0x01 00 00 00[6]	0x01 00 00 00[6]
QLLC	0xC3	Not available
XNS	0xD1	0x80 (5-byte SNAP encoding)
Multiprotocol	Not available	0x00

1. SNAP encoding is defined according to the Assigned Numbers RFC; the Cisco implementation recognizes only the IETF organizational unique identifier (OUI) 0x00 00 00 followed by a 2-byte Ethernet protocol type.
2. The use of 0xC0 00 80 C4 for Banyan VINES is defined by Banyan.
3. The use of 0x81 for CLNS is compatible with ISO/IEC 8473-3:1994.
4. Compressed TCP traffic has two types of datagrams, so IETF encapsulation requires a multiprotocol VC.
5. The use of 0xCC for IP is backward-compatible with RFC 877.
6. The use of 0x01 00 00 00 for PAD is defined by ITU-T *Recommendation X.29*.

Once a multiprotocol VC has been established, datagrams on the VC have protocol identification data before the actual protocol data; the protocol identification values are the same as those used by RFC 1356 in the CUD field for an individual protocol.

Note IP datagrams can be identified with a 1-byte identification (0xCC) or a 6-byte identification (0x80 followed by the 5-byte SNAP encoding). The 1-byte encoding is used by default, although the SNAP encoding can be configured.

Mapping Datagram Addresses to X.25 Hosts

Encapsulation is a cooperative process between the router and another X.25 host. Because X.25 hosts are reached with an X.121 address (an X.121 address has 0 to 15 decimal digits), the router must have a means to map protocols and addresses of the host to its X.121 address.

Each encapsulating X.25 interface must be configured with the relevant datagram parameters. For example, an interface that encapsulates IP typically will have an IP address.

A router set up for DDN or BFE service uses a dynamic mapping technique to convert between IP and X.121 addresses. These techniques have been designed specifically for attachment to the DDN network and to Blacker encryption equipment. Their design, restrictions, and operation make them work well for these specific applications, but not for other networks.

You must also establish the X.121 address of an encapsulating X.25 interface using the **x25 address** interface configuration command. This X.121 address is the address to which encapsulation calls are directed, and is also the source X.121 address used for originating an encapsulation call. It is used by the destination host to map the source host and protocol to the protocol address. An encapsulation VC must be a mapped at both the source and destination host interfaces. A DDN or BFE interface will have an X.121 address generated from the interface IP address, which, for proper operation, should not be modified.

For each X.25 interface, you must explicitly map the protocols and addresses for each destination host to its X.121 address. If needed and the destination host has the capability, one host map can be configured to support several protocols; alternatively, you can define one map for each supported protocol.

To establish an X.25 map, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address [option]`	Maps one or more host protocol addresses to the X.121 address of the host.

For example, if you are encapsulating IP over a given X.25 interface, you must define an IP address for the interface and, for each of the desired destination hosts, map the IP address of the host to its X.121 address.

Note You can map an X.121 address to as many as nine protocol addresses, but each protocol can be mapped only once in the command line.

An individual host map can use keywords to specify the following protocols:

- **apollo**—Apollo Domain
- **appletalk**—AppleTalk
- **bridge**—Bridging
- **clns**—OSI Connectionless Network Service
- **compressedtcp**—TCP/IP header compression
- **decnet**—DECnet
- **ip**—IP
- **ipx**—Novell IPX
- **pad**—Packet assembler/disassembler
- **qllc**—IBM QLLC
- **vines**—Banyan VINES
- **xns**—XNS

Each mapped protocol, except bridging and CLNS, takes a datagram address. All bridged datagrams are either broadcast to all bridging destinations or sent to the X.121 address of a specific destination host, and CLNS uses the mapped X.121 address as the subnetwork point of attachment (SNPA), which is referenced by a **clns neighbor** command. The configured datagram protocols and their relevant addresses are mapped to the X.121 address of the destination host. All protocols that are supported for RFC 1356 operation can be specified in a single map. (Bridging and QLLC are not supported for RFC 1356 encapsulation.) If IP and TCP/IP header compression are both specified, the same IP address must be given for both protocols.

When setting up the address map, you can include options such as enabling broadcasts, specifying the number of VCs allowed and defining various user facility settings.

Note Multiprotocol maps, especially those configured to carry broadcast traffic, can result in significantly larger traffic loads, requiring a larger hold queue, larger window sizes, or multiple VCs.

For specific information about how to establish a protocol to run over X.25, refer to the appropriate protocol chapters in the *Cisco IOS IP Configuration Guide*, *Cisco IOS AppleTalk and Novell IPX Configuration Guide*, and *Cisco IOS Apollo Domain, Banyan VINES, DECnet, ISO CLNS, and XNS Configuration Guide*.

You can simplify the configuration for the Open Shortest Path First (OSPF) protocol by adding the optional **broadcast** keyword. See the **x25 map** command description in the chapter "X.25 and LAPB Commands" in the *Cisco IOS Wide-Area Networking Command Reference* for more information.

Configuring PAD Access

By default, PAD connection attempts are processed for session creation or protocol translation (subject to the configuration of those functions) from all hosts. To restrict PAD connections only to statically mapped X.25 hosts, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# x25 pad-access`	Restricts PAD access.
Step 2	`Router(config-if)# x25 map pad` *x121-address* [*option*]	Configures a host for PAD access.

You can configure outgoing PAD access using the optional features of the **x25 map pad** command without restricting incoming PAD connections to the configured hosts.

Establishing an Encapsulation PVC

PVCs are the X.25 equivalent of leased lines; they are never disconnected. You need not configure an address map before defining a PVC; an encapsulation PVC implicitly defines a map.

To establish a PVC, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 pvc` *circuit protocol address* [*protocol2 address2* [...[*protocol9 address9*]]] *x121-address* [*option*]	Sets an encapsulation PVC.

The **x25 pvc** command uses the same protocol keywords as the **x25 map** command. See the section "Mapping Datagram Addresses to X.25 Hosts" earlier in this chapter for a list of protocol keywords. Encapsulation PVCs also use a subset of the options defined for the **x25 map** command.

The user may establish multiple, parallel PVCs that carry the same set of encapsulation traffic by specifying the identical mappings for each PVC. Additionally, the user can permit a mixture of SVCs and PVCs to carry the traffic set by using the **x25 map** command to specify an **nvc** *count* that exceeds the number of configured PVCs. The total number of VCs, of whatever type, can never exceed 8.

For an example of configuring a PVC, see the section "PVC Used to Exchange IP Traffic Example" later in this chapter.

Setting X.25 TCP/IP Header Compression

Cisco supports RFC 1144 TCP/IP header compression (THC) on serial lines using HDLC and X.25 encapsulation. THC encapsulation is only slightly different from other encapsulation traffic, but the differences are worth noting. The implementation of compressed TCP over X.25 uses one VC to pass the compressed packets. Any IP traffic (including standard TCP) is separate from THC traffic; it is carried over separate IP encapsulation VCs or identified separately in a multiprotocol VC.

Note If you specify both **ip** and **compressedtcp** in the same **x25 map compressedtcp** command, they must both specify the same IP address.

To set up a separate VC for X.25 THC, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 map compressedtcp ip-address [protocol2 address2 [...[protocol9 address9]]] x121-address [option]`	Allows a separate VC for compressed packets.

Configuring X.25 Bridging

Cisco IOS transparent bridging software supports bridging over X.25 VCs. Bridging is not supported for RFC 1356 operation. Bridge maps must include the **broadcast** option for correct operation.

To enable the X.25 bridging capability, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 map bridge x121-address broadcast [option]`	Defines bridging of X.25 frames.

Configuring Additional X.25 Datagram Transport Features

The Cisco IOS software allows you to configure additional X.25 datagram transport features, including various user facilities defined for X.25 call setup.

This section describes the X.25 datagram transport features you can configure by using the options in the **x25 map** or **x25 pvc** encapsulation command (or by setting an interface default). The tasks you perform depend upon your needs, the structure of your network, and the requirements of the service provider.

To configure the optional parameters, user facilities, and special features, perform one or more of the tasks described in the following sections:

- Configuring X.25 Payload Compression
- Configuring the Encapsulation VC Idle Time
- Increasing the Number of VCs Allowed
- Configuring the Ignore Destination Time
- Establishing the Packet Acknowledgment Policy
- Configuring X.25 User Facilities
- Defining the VC Packet Hold Queue Size
- Restricting Map Usage

Configuring X.25 Payload Compression

For increased efficiency on relatively slow networks, the Cisco IOS software supports X.25 payload compression of outgoing encapsulation traffic.

The following restrictions apply to X.25 payload compression:

- The compressed VC must connect two Cisco routers, because X.25 payload compression is not standardized.

  The data packets conform to X.25 rules, so a compressed VC can be switched through standard X.25 equipment. However, only Cisco routers can compress and decompress the data.

- Only datagram traffic can be compressed, although all the encapsulation methods supported by Cisco routers are available (for example, an IETF multiprotocol VC can be compressed).

  SVCs cannot be translated between compressed and uncompressed data, nor can PAD data be compressed.

- X.25 payload compression must be applied carefully.

  Each compressed VC requires significant memory resources (for a dictionary of learned data patterns) and computation resources (every data packet received is decompressed and every data packet sent is compressed). Excessive use of compression can cause unacceptable overall performance.

- X.25 compression must be explicitly configured for a map command.

  A received call that specifies compression will be rejected if the corresponding host map does not specify the **compress** option. An incoming call that does not specify compression can, however, be accepted by a map that specifies compression.

To enable payload compression over X.25, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 map` *protocol address* [*protocol2 address2* [...[*protocol9 address9*]]] *x121-address* `compress`	Enables payload compression over X.25.

This command specifies that X.25 compression is to be used between the two hosts. Because each VC established for compressed traffic uses significant amounts of memory, compression should be used with careful consideration of its impact on the performance.

The **compress** keyword may be specified for an encapsulation PVC.

Configuring the Encapsulation VC Idle Time

The Cisco IOS software can clear a datagram transport or PAD SVC after a set period of inactivity. Routed SVCs are not timed for inactivity.

To set the time, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	`Router(config-if)# x25 idle minutes`	Sets an idle time for clearing encapsulation.
Step 2	`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address idle minutes`	Specifies idle time for clearing SVCs of a map.

For an example of configuring the SVC idle timer, see the section "Typical X.25 Configuration Example" later in this chapter. See the section "Monitoring and Maintaining LAPB and X.25," later in this chapter, for additional commands that clear VCs.

Increasing the Number of VCs Allowed

For X.25 datagram transport, you can establish up to eight VCs to one host for each map.

To increase the number of VCs allowed, use one or both of the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 nvc count`	Specifies the default maximum number of SVCs that can be open simultaneously to one host for each map.
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address nvc count`	Specifies the maximum number of SVCs allowed for a map.

For an example of increasing the number of VCs allowed, see the sections "Typical X.25 Configuration Example" and "DDN X.25 Configuration Example" later in this chapter.

Configuring the Ignore Destination Time

Upon receiving a Clear for an outstanding datagram transport Call Request, the X.25 encapsulation code immediately tries another Call Request if it has more traffic to send. This action can overrun some X.25 switches.

To define the number of minutes for which the Cisco IOS software will prevent calls from going to a previously failed destination, use the following command in interface configuration mode (incoming calls will still be accepted and cancel the timer):

Command	Purpose
`Router(config-if)# x25 hold-vc-timer minutes`	Configures the ignore destination time.

Establishing the Packet Acknowledgment Policy

You can instruct the Cisco IOS software to send an acknowledgment packet when it has received a threshold of data packets it has not acknowledged, instead of waiting until its input window is full. A value of 1 sends an acknowledgment for each data packet received if it cannot be acknowledged in an outgoing data packet. This approach improves line responsiveness at the expense of bandwidth. A value of 0 restores the default behavior of waiting until the input window is full.

To establish the acknowledgment threshold, use the following command in interface configuration mode (the packet acknowledgment threshold also applies to encapsulation PVCs):

Command	Purpose
`Router(config-if)# x25 threshold delay-count`	Sets data packet acknowledgement threshold.

Configuring X.25 User Facilities

X.25 software provides commands to support X.25 user facilities options (specified by the ITU-T *Recommendation X.25*) that allow you to use network features such as reverse charging, user identification, and flow control negotiation. You can choose to configure facilities on a per-map basis or on a per-interface basis. In the following table, the **x25 map** commands configure facilities on a per-map basis; the **x25 facility** commands specify the values set for all encapsulation calls originated by the interface. Routed calls are not affected by the facilities specified for the outgoing interface.

To set the supported X.25 user facilities options, use one or more of the following commands in interface configuration mode:

Command	Purpose

`Router(config-if)# x25 facility cug number`
or
`x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address cug group-number`
	Selects the closed user group (CUG).

`Router(config-if)# x25 facility packetsize in-size out-size`
or
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address packetsize in-size out-size`
or
`Router(config-if)# x25 facility windowsize in-size out-size`
or
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address windowsize in-size out-size`
	Sets the flow control parameter negotiation values to be requested on outgoing calls.

`Router(config-if)# x25 facility reverse`
 	
or
 	
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address reverse`
	Sets reverse charging.

`Router(config-if)# x25 accept-reverse`
 	
or
 	
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address accept-reverse`
	Allows reverse charging acceptance.

`Router(config-if)# x25 facility throughput in out`
 	
or
 	
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address throughput in out`
	Selects throughput class negotiation.

`Router(config-if)# x25 facility transit-delay milliseconds`
 	
or
 	
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address transit-delay milliseconds`
	Selects transit delay.

`Router(config-if)# x25 facility roa name`
 	
or
 	
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address roa name`
	Sets which Recognized Operating Agency (ROA) to use.

`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address nuid username password`	Sets the Cisco standard network user identification.
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address nudata string`	Sets a user-defined network user identification, allowing the format to be determined by your network administrator.

The **packetsize** and **windowsize** and options are supported for PVCs, although the options have a slightly different meaning on PVCs from what they they mean on interfaces because PVCs do not use the call setup procedure. If the PVC does not use the interface defaults for the flow control parameters, these options must be used to specify the values. Not all networks will allow a PVC to be defined with arbitrary flow control values.

Additionally, the D-bit is supported, if negotiated. PVCs allow the D-bit procedure because there is no call setup to negotiate its use. Both restricted and unrestricted fast select are also supported and are transparently handled by the software. No configuration is required for use of the D-bit or fast select facilities.

Defining the VC Packet Hold Queue Size

To define the maximum number of packets that can be held while a VC is unable to send data, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 hold-queue packets`	Defines the VC packet hold queue size.

A hold queue size of an encapsulation VC is determined when it is created; the **x25 hold-queue** command does not affect existing VCs. This command also defines the hold queue size of encapsulation PVCs.

Restricting Map Usage

An X.25 map can be restricted so that it will not be used to place calls or so that it will not be considered when incoming calls are mapped.

To restrict X.25 map usage, use the following commands in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address no-incoming`	Restricts incoming calls from a map.
`Router(config-if)# x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address no-outgoing`	Restricts outgoing calls from a map.

Configuring X.25 Routing

The X.25 software implementation allows VCs to be routed from one X.25 interface to another and from one router to another. The routing behavior can be controlled with switching and XOT configuration commands, based on a locally built table.

X.25 encapsulation can share an X.25 serial interface with the X.25 switching support. Switching or forwarding of X.25 VCs can be done two ways:

- Incoming calls received from a local serial interface running X.25 can be forwarded to another local serial interface running X.25. This method is known as *local X.25 switching* because the router handles the complete path. It does not matter whether the interfaces are configured as DTE or DCE devices, because the software takes the appropriate actions.
- An incoming call can also be forwarded using the XOT service (previously *remote switching* or *tunneling*). Upon receipt of an incoming X.25 call, a TCP connection is established to the destination XOT host (for example, another Cisco router) that will, in turn, handle the call using its own criteria. All X.25 packets are sent and received over the reliable TCP data stream. Flow control is maintained end-to-end. It does not matter whether the interface is configured for DTE or DCE devices, because the software takes the appropriate actions.

Running X.25 over TCP/IP provides a number of benefits. The datagram containing the X.25 packet can be switched by other routers using their high-speed switching abilities. X.25 connections can be sent over networks running only the TCP/IP protocols. The TCP/IP protocol suite runs over many different networking technologies, including Ethernet, Token Ring, T1 serial, and FDDI. Thus X.25 data can be forwarded over these media to another router, where it can, for example, be switched to an X.25 interface.

When the connection is made locally, the switching configuration is used; when the connection is across a LAN, the XOT configuration is used. The basic function is the same for both types of connections, but different configuration commands are required for each type of connection.

The X.25 switching subsystem supports the following facilities and parameters:

- D-bit negotiation (data packets with the D-bit set are passed through transparently)
- Variable-length interrupt data (if not operating as a DDN or BFE interface)
- Flow control parameter negotiation:
  - Window size up to 7, or 127 for modulo 128 operation
  - Packet size up to 4096 (if the LAPB layers used are capable of handling the requested size)
- Basic CUG selection
- Throughput class negotiation
- Reverse charging and fast select

The handing of these facilities is described in the appendix "X.25 Facility Handling."

To configure X.25 routing, perform the tasks in the following sections:

- Enabling X.25 Routing
- Configuring an X.25 Route
- Configuring a PVC Switched Between X.25 Interfaces
- Configuring X.25 Switching Between PVCs and SVCs

See the section "Configuring Additional X.25 Routing Features" for further configuration options for your network.

Enabling X.25 Routing

You must enable X.25 routing to use switch VCs.

To enable X.25 routing, use the following command in global configuration mode:

Command	Purpose
`Router(config)# x25 routing [use-tcp-if-defs]`	Enables X.25 routing.

The **use-tcp-if-defs** keyword is used by some routers that receive remote routed calls from older versions of XOT; it might be needed if the originating router cannot be updated to a new software release. The use of this keyword is described in the section "Configuring XOT to Use Interface Default Flow Control Values" later in this chapter.

For examples of configuring X.25 routing, see the sections "X.25 Route Address Pattern Matching Example" and "X.25 Routing Examples" later in this chapter.

Configuring an X.25 Route

An X.25 route table enables you to control which destination is selected for several applications. When an X.25 service receives a call that must be forwarded, the X.25 route table determines which X.25 service (X.25, CMNS, or XOT) and destination should be used. When a PAD call is originated by the router, either from a user request or from a protocol translation event, the route table similarly determines which X.25 service and destination should be used.

You create the X.25 route table and add route entries to it. You can optionally specify the order of the entries in the table, the criteria to match against the VC information, and whether to modify the destination or source addresses. Each entry must specify the disposition of the VC (that is, what is done with the VC). Each route can also specify XOT keepalive options.

The route table is used as follows:

- VC information is matched against selection criteria specified for each route.
- The table is scanned sequentially from the top.
- The first matching route determines how the VC is handled.
- Once a matching entry is found, the call addresses can be modified and the call disposed of (forwarded or cleared) as instructed by the entry.

Each application can define special conditions if a route will not be used or what occurs if no route matches. For instance, switched X.25 will skip a route if the disposition interface is down and clear a call if no route matches. X.25 PAD and PAD-related applications, such as protocol translation using X.25, will route the call to the default X.25 interface, which is the first X.25 interface configured.

To configure an X.25 route (thus adding the route to the X.25 routing table), use the following command in global configuration mode:

Command	Purpose
`Router(config)# x25 route [#position] [selection-options] [modification-options] disposition-options [xot-keepalive-options]`	Configures an X.25 route.

The following options offer versatility and flexibility when you use the **x25 route** command:

- *#position*—Position in the table. You can use the optional *#position* element to indicate the number of the entry in the route able. For example, #9 indicates the ninth entry from the top. The route table is always searched sequentially from the top, and the first match found will be used.
- *selection-options*—Criteria to define to which VCs the route will apply. You can match against zero to four of the following optional *selection* elements:
  - *destination-pattern*
  - **source** *source-pattern*
  - **dest-ext** *nsap-destination-pattern*
  - **cud** *user-data-pattern*
- *modification-options*—Modifications to the source or destination address for address translation. You can use neither, one, or both of the following optional *modification* elements to change the source or destination address before forwarding the call to the destination:
  - **substitute-source** *rewrite-source*
  - **substitute-dest** *rewrite-destination*

Note You must include a selection option or a modification option in an **x25 route** command.

- *disposition-options*—Where the VC will be forwarded or whether it will be cleared. You are required to use one of the following *disposition* elements:
  - **interface** *serial-interface*

    A route to a specific *serial-interface* will send the VC to an X.25 service on a synchronous serial interface.
  - **interface** *cmns-interface* **mac** *mac-address*

    A route to a broadcast interface will send the VC to a CMNS partner reachable on a broadcast medium at a specified MAC address. The CMNS interface can be an Ethernet, Token Ring, or FDDI interface.
  - **xot** *ip-address* [*ip2-address* [...[*ip6-address*]]] [**xot-source** *interface*]

    A route to an **xot** destination (formerly called a *remote* or *tunneled* configuration) will send the VC to the XOT service for establishment of a TCP connection across which the XOT VC packets will travel. An **xot** disposition may specify alternate destinations to try if a TCP connection cannot be established for all preceding destinations.
  - **clear**

    A route to a **clear** destination will deny further service to the VC by shutting down the connection.
- *xot-keepalive-options*—You can use neither, one, or both of the following optional *xot-keepalive* elements:
  - **xot-keepalive-period** *seconds*
  - **xot-keepalive-tries** *count*

Configuring a PVC Switched Between X.25 Interfaces

You can configure an X.25 PVC in the X.25 switching software. As a result, DTE devices that require permanent circuits can be connected to a router acting as an X.25 switch and have a properly functioning connection. X.25 resets will be sent to indicate when the circuit comes up or goes down. Both interfaces must define complementary locally switched PVCs.

To configure a locally switched PVC, use the following command in interface configuration mod:

Command	Purpose
`Router(config-if)# x25 pvc number1 interface type number pvc number2 [option]`	Configures a locally switched PVC.

The command options are **packetsize** *in out* and **windowsize** *in out*; they allow the flow control values of a PVC to be defined if they differ from the interface defaults.

For an example of configuring a locally switched PVC, see the section "PVC Switching on the Same Router Example" later in this chapter.

To ensure that TCP sessions remain connected in the absence of XOT traffic, use the following command in global configuration mode :

Command	Purpose
`Router(config)# service tcp-keepalives-in`	Enables received keepalives for TCP sessions to ensure timely detection of a connection failure.
`Router(config)# service tcp-keepalives-out`	Enables sent keepalives for TCP sessions to ensure timely detection of a connection failure.

TCP keepalives also inform a router when an XOT SVC session is not active, thus freeing router resources.

For examples of enabling keepalives, see the sections "Simple Switching of a PVC over XOT Example" and "PVC Switching over XOT Example" later in this chapter.

Configuring X.25 Switching Between PVCs and SVCs

In order for PVC to SVC switching to be configured between two serial interfaces, both interfaces must already be configured for X.25. In addition, X.25 switching must be enabled using the **x25 routing** global configuration command. The PVC interface must be a serial interface configured with X.25 encapsulation. (The SVC interface may use X.25, XOT, or CMNS.)

To configure X.25 switching between PVCs and SVCs, use the following command in interface configuration mode. X.25 switching must already be configured on the interace.

Command	Purpose
`Router(config-if)# x25 pvc` *number1* `svc` *x121-address* [*flow-control-options*] [*call-control-options*]	Configures PVC traffic to be forwarded to an SVC.

To display information about the switched PVC to SVC circuit, use the following command in EXEC mode:

Command	Purpose
`Router(config)# show x25 vc` [*lcn*]	Displays information about the active SVCs and PVCs.

For an example of configuring switching between a PVC and SVC, see the section "X.25 Switching Between PVCs and SVCs Example" later in this chapter.

Configuring Additional X.25 Routing Features

To configure additional X.25 routing features, perform the tasks in the following sections:

- Configuring X.25 Load Balancing
- Configuring XOT to Use Interface Default Flow Control Values
- Configuring Calling Address Interface-Based Insertion and Removal
- Substituting Addresses in an X.25 Route
- Configuring XOT Alternate Destinations

Configuring X.25 Load Balancing

X.25 load balancing was created to solve the problem that arises when the number of users accessing the same host causes an overload on Internet service provider (ISP) application resources.

In the past, in order to increase the number of users they could support, ISPs had to increase the number of X.25 lines to the host. To support a large number of VCs to a particular destination, they had to configure more than one serial interface to that destination. When a serial interface is configured to support X.25, a fixed number of VCs is available for use. However, the X.25 allocation method for VCs across multiple serial lines filled one serial line to its VC capacity before utilizing the second line at all. As a result, the first serial line was frequently carrying its maximum data traffic before it ran out of VCs.

Using a facility called *hunt groups,* the X.25 Load Balancing feature causes a switch to view a pool of X.25 lines going to the same host as one address and assign VCs on an idle logical channel basis. With this feature, X.25 calls can be load-balanced among all configured outgoing interfaces to fully use and balance performance of all managed lines. X.25 load balancing allows two load-balancing distribution methods—rotary and vc-count—utilizing multiple serial lines.

The rotary method sends every call to the next available interface, regardless of line speed and the number of available VCs on that interface.

The vc-count method sends calls to the interface that has the largest number of available logical channels. This method ensures a good load balance when lines are of equal speed. If the line speeds are unequal, the vc-count method will favor the line with the higher speed. To distribute calls equally among interfaces regardless of line speed, configure each interface with the same number of VCs. In cases where interfaces have the same line speed, the call is sent to the interface that is defined earliest in the hunt group.

With the vc-count distribution method, if a hunt group does not contain an operational interface, the call is forwared to the next route if one has been specified. An interface is considered unoperational if that interface is down or full. If a session is terminated on an interface within the hunt group, that interface now has more available VCs, and it will be chosen next.

Note XOT cannot be used in hunt groups configured with the vc-count distribution method. XOT does not limit the number of calls that can be sent to a particular destination, so the method of selecting the hunt group member with the largest number of available VCs will not work. XOT can be used in hunt groups configured with the rotary distribution method.

Only one distribution method can be selected for each hunt group, although one interface can participate in one or more hunt groups. Reconfiguration of hunt groups does not affect functionality, but distribution methods are limited to rotary and vc-count only.

Before enabling X.25 load balancing, you must activate the X.25 routing software and configure the interfaces participating in the hunt group for X.25 encapsulation. To configure X.25 load balancing, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# x25 routing`	Activates X.25 routing software.
Step 2	`Router(config)# encapsulation x25`	Specifies X.25 encapsulation on each hunt group interface.

	Command	Purpose
Step 3	`Router(config)# x25 hunt-group` *name* `{rotary | vc-count}`	Creates the hunt group.
Step 4	`Router(config)# x25 route [#`*position*`] [`*selection-options*`] [`*modification-options*`]` *disposition-options* `[`*xot-keepalive-options*`]`	Adds the hunt group to the routing table.

For examples of configuring X.25 load balancing, see the section "X.25 Load Balancing Examples" later in this chapter.

Verifying X.25 Load Balancing

To verify X.25 load balancing, use the following command in EXEC mode:

Command	Purpose
`Router# show x25 hunt-group`	Displays hunt groups and detailed interface statistics and distribution methods.

Configuring XOT to Use Interface Default Flow Control Values

When a connection is set up, the source and destination XOT implementations must cooperate to determine the flow control values that apply to the SVC. The source XOT ensures cooperation by encoding the X.25 flow control facilities (the window sizes and maximum packet sizes) in the X.25 Call packet; the XOT implementation of the far host can then correctly negotiate the flow control values at the destination interface and, if needed, indicate the final values in the X.25 Call Confirm packet.

When XOT receives a call that leaves one or both flow control values unspecified, it supplies the values. The values supplied are a window size of 2 packets and maximum packet size of 128 bytes; according to the standards, any SVC can be negotiated to use these values. Thus when XOT receives a call from an older XOT implementation, it can specify in the Call Confirm packet that these flow control values must revert to the lowest common denominator.

The older XOT implementations required that the source and destination XOT router use the same default flow control values on the two X.25 interfaces that connect the SVC. Consequently, connections with mismatched flow control values were created when this assumption was not true, which resulted in mysterious problems. In the Cisco IOS Release 12.2 XOT implementation, the practice of signalling the values used in the Call Confirm packet avoids these problems.

Occasionally the older XOT implementation will be connected to a piece of X.25 equipment that cannot handle modification of the flow control parameters in the Call Confirm packet. These configurations should be upgraded to use a more recent version of XOT; when upgrade is not possible, the behavior of XOT causes a migration problem. In this situation, you may configure the Cisco IOS software to cause XOT to obtain unspecified flow control facility values from the default values of the destination interface.

To configure this behavior, use the following command when enabling X.25 routing in global configuration mode:

Command	Purpose
`Router(config)# x25 routing [tcp-use-if-defs]`	Enables X.25 routing and optionally modifies XOT source of unencoded flow control values.

Configuring Calling Address Interface-Based Insertion and Removal

This feature describes a modification to the **x25 route** command that allows interface-based insertion and removal of the X.121 address in the X.25 routing table.

This capability allows Cisco routers running X.25 to conform to the standard that specifies that X.25 DCE devices should not provide the X.25 calling address, but instead that it should be inserted by the X.25 DTE based on interface. This calling address insertion and removal feature was designed for all routers performing X.25 switching and requiring that an X.121 address be inserted or removed by the X.25 DTE based on the interface.

This feature does not support XOT to X.25 routing using the **input-interface** keyword introduced by the Calling Address Insertion and Removal feature.

To configure an input interface-based route statement into the X.121 address routing table, use either of the following commands beginning in global configuration command mode:

Command	Purpose
`Router(config)# x25 route input-interface` *interface* `source` *source-pattern* `substitute-source` *rewrite-source* `[continue]` or `Router(config)# x25 route input-interface` *interface disposition*	Inserts an input interface-based route statement into the routing table. Inserts simplest input interface-based statement into the routing table.

The **continue** keyword is optional. It performs address substitution without address forwarding. That is, it executes the address substitution instructions in each statement, but then stops short of actual call switching, thereby postponing the actual switching process until a matching route statement with a disposition other than **continue** is reached. The **continue** keyword is most useful when you switch calls among four or more routes. If your network has three or fewer routes, the **continue** keyword will not save any steps.

For examples of configuring interface-based call address insertion and removal, see the sections "Inserting and Removing X.121 Addresses As Calls Are Routed Example" and "Forwarding Calls Using the continue Keyword Example" later in this chapter.

Verifying Interface-Based Calling Address Insertion

To display the routes assigned by the **x25 route** command, use the **show x25 route** command in EXEC mode. A sample display follows.

```
Router# show x25 route
  #  Match                              Substitute          Route to
  1  dest ^01 input-int Serial0    Sub-dest \1        Sub-source 00\0 Serial1
```

Substituting Addresses in an X.25 Route

When interconnecting two separate X.25 networks, you must sometimes provide address substitution for routes. The **x25 route** command supports modification of X.25 source and destination addresses.

To modify addresses, use either or both of the following commands in global configuration mode:

Command	Purpose
`Router(config)# x25 route [#position] destination-pattern {source source-pattern | substitute-source rewrite-source} interface interface number`	Modifies the X.25 source address.
`Router(config)# x25 route [#position] destination-pattern {source source-pattern | substitute-dest rewrite-dest} interface interface number`	Modifies the X.25 destination address.

Address substitution is available for all applications of X.25 routes.

Configuring XOT Alternate Destinations

Routes to XOT hosts can be configured with alternate destination hosts. On routing a call, XOT will try each XOT destination host in sequence; if the TCP connection attempt fails, the next destination will be tried. Up to six XOT destination addresses can be entered.

To configure an XOT route with alternate destinations (thus adding it to the X.25 routing table), use the following command in global configuration mode (the sequence of alternate destination XOT host addresses is added to the **x25 route** command using the *xot keepalive-options*):

Command	Purpose
`Router(config)# x25 route [#position] destination-pattern xot ip-address [ip-address2... [ip-address6]]`	Configures an XOT route. Optionally defines alternate XOT destination hosts.

Note Because of TCP timings, it can take up to 50 seconds to try an alternate route.

For an example of constructing the routing table, see the section "X.25 Routing Examples" later in this chapter.

Configuring DNS-Based X.25 Routing

Managing a large TCP/IP network requires accurate and up-to-date maintenance of IP addresses and X.121 address mapping information on each router database in the network. Because these IP addresses are constantly being added and removed in the network, the routing table of every router needs to be updated, which is a time consuming and error-prone task. This process has also been a problem for mnemonics (an easy-to-remember alias name for an X.121 address).

X.25 has long operated over an IP network using XOT. However, large networks and financial legacy environments experienced problems with the amount of route configuration that needed to be done manually, as each router switching calls over TCP needed every destination configured. Every destination from the host router needed a static IP route statement, and for larger environments, these destinations could be as many as several thousand per router. Until the release of Domain Name System (DNS)-based X.25 routing, the only way to map X.121 addresses and IP addresses was on a one-to-one basis using the **x25 route** *x121address* **xot** *ipaddress* command.

The solution was to centralize route configurations that routers could then access for their connectivity needs. This centralization is the function of DNS-based X.25 routing, because the DNS server is a database of all domains and addresses on a network.

DNS-based X.25 routing scales well with networks that have multiple XOT routers, simplifies maintenance of routing table and creation of new routes, and reduces labor-intensive tasks and the possibility of human error during routing table maintainance. You must have DNS activated and X.25 configured for XOT to enable DNS-based X.25 routing.

DNS has the following three components:

- Domain name space or resource records—Define the specifications for a tree-structured domain name space.
- Name servers—Hold information about the domain tree structure.
- Resolvers—Receive a client request and return the desired information in a form compatible with a local hosts data formats.

You need to maintain only one route statement in the host router to connect it to the DNS. When DNS is used, the following rules apply:

- You must use Cisco IOS name server configuration commands.
- X.28 mnemonic restrictions apply (for example, not using **-**, **.**, **P**, or **D** in the mnemonic).
- You cannot specify any **x25 route** command options on the DNS. These options must be configured within the **x25 route** command itself.
- Names must consist of printable characters.
- No embedded white space is permitted.
- Periods must separate subdomains.
- Names are case sensitive.
- You must append any domain configured for the router to the user-specified name format.
- The total length of the name must not exceed 255 characters.

For more information on configuring the DNS, see the chapter “Configuring the DNS Service” in the *Cisco DNS/DHCP Manager Administrator’s Guide*.

See the following sections for details about address and mnemonic resolution and verification of this feature:

- Address Resolution
- Mnemonic Resolution
- Verifying DNS-Based X.25 Routing
- Verifying DNS-Based X.25 Mnemonic Resolution

Note This feature should not be used in the public Internet. It should be used only for private network implementations because in the Internet world the DNS has conventions for names and addresses with which DNS-based X.25 routing does not comply.

Address Resolution

With DNS-based X.25 routing, managing the X.121-to-IP addressing correlation and the mnemonic-to-X.121 addressing correlation is easy. Instead of supplying the router multiple route statements to all destinations, it may be enough to use a single wildcard route statement that covers all addresses in the DNS.

The **x25 route** *disposition* **xot** command option has been modified to include the **dns** *pattern* argument after the **xot** keyword, where *pattern* is a rewrite element that works in the same way that address substitution utilities work (see the *Cisco IOS Wide-Area Networking Command Reference* for further details).

The wildcard **^.*** characters and **\0** pattern of the modified **x25 route ^.* xot dns \0** command give the command more universality and effectiveness and make DNS-based X.25 routing simple and easy to use. These characters and pattern already exist and are explained in detail under the **x25 route** command in the *Cisco IOS Wide-Area Networking Command Reference*. This command functions only if the DNS route table mapping has been configured in a method recognized and understood by X.25 and the DNS server.

The following example is a setup from a DNS route table showing which X.121 address relates to which IP address:

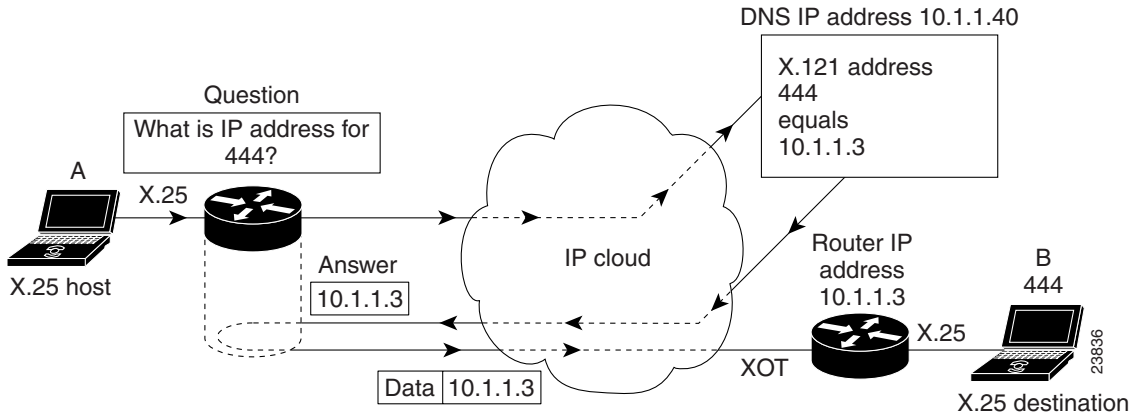
```
222 IN      A       172.18.79.60
444 IN      A       10.1.1.3
555 IN      A       10.1.1.2 10.1.2.2 10.1.3.2 10.1.4.2 10.1.5.7 10.1.6.3
```

The command line **x25 route 444 xot dns \0** shown in the DNS-based X.25 routing configuration example is what extracts the IP address from the DNS. The **\0** pattern replaces itself with 444. The 444 is then used as the index into the DNS route table to generate the IP address 10.1.1.3. Other characters can be combined with the pattern; for example, **A-\0**. In the DNS database, the index would appear as A-444.

As the example in Figure 4 shows, a call sent by the router goes to the DNS. The DNS checks its route table and identifies the X.121 address 444 and its related IP address 10.1.1.3. The DNS returns the IP address to the host router, which then creates a route statement and forwards the data to the IP address of the destination router (10.1.1.3).

If the DNS-based X.25 routing configuration example included the command **x25 route 555 xot dns \0**, then a call to the X.121 address 555 would also go to the DNS. Since multiple IP addresses have been configured in the domain name space records, all of the IP addresses for that domain name would be returned to the router. Each address would be tried in sequence, just as if the X.25 routing configuration had been **x25 route 555 xot 10.1.1.2 10.1.2.2 10.1.3.2 10.1.4.2 10.1.5.7 10.1.6.3**. The router will accept up to 6 IP addresses from DNS for the domain name. If there are more than six, there will be an error message, and the list will be trucated to the first six received.

Figure 4 DNS-Based X.25 Routing Using XOT over an IP Cloud



Mnemonic Resolution

DNS-based X.25 routing can be used for mnemonic resolution with or without use of XOT routing. For more information on mnemonic addressing, refer to the chapter "Configuring the Cisco PAD Facility for X.25 Connections" chapter in the *Cisco IOS Terminal Services Configuration Guide*.

When mnemonics are used with XOT, the same communication with the DNS occurs, except that the router needs to contact the DNS twice—first to get the X.121 address using the mnemonic, and then to get the IP address using the X.121 address. However, there is no substantial performance issue because the process happens very quickly.

The following example is a setup from the DNS route table showing a mnemonic and its related X.121 address ("destination_host" represents 222). The **X25** keyword ensures that this line will be recognized by DNS-based X.25 routing in the DNS server.

```
destination_host IN      X25     222
```

Using X.28 to retrieve this address, you would enter the following commands:

```
Router# x28
*destination_host
Translating "destination_host"...domain server (10.1.1.40)
```

Notice the output line requesting mnemonic resolution from the DNS server with IP address 10.1.1.40.

If you were using PAD, you would need to enter only the mnemonic name, as in the following example:

```
Router# pad destination_host
```

Caution You must remove any permanent entry for X.25 located in the host table of the router that has been duplicated in the DNS route table (as part of the enabling process for DNS-based X.25 routing). Otherwise, DNS-based X.25 routing will be overridden by the host table entries of the router.

To configure DNS-based X.25 routing, use the following command in global configuration mode. This task assumes that you already have XOT and DNS configured and enabled and that the route table in the DNS server has been correctly organized.

Command	Purpose
`Router(config)# x25 route x121address xot dns pattern`	Configures XOT routing to search for IP addresses in DNS.

For an example of configuring DNS-based X.25 routing, see the section "DNS-Based X.25 Routing Example" later in this chapter.

Verifying DNS-Based X.25 Routing

To verify that the DNS-Based X.25 Routing feature is configured, use the **show x25 route** command in EXEC mode:

```
Router# show x25 route
  #  Match                          Substitute          Route to
  1  dest 444                                           xot dns \0
  2  dest 555                                           xot dns \0
```

If DNS-based X.25 routing is not functioning correctly, check that your DNS is configured properly and operating correctly as follows:

- Use the **show hosts** command to display temporary entries cached by DNS at the router.
- Use **debug x25 events** and **debug domain** commands to display current data flow. See the *Cisco IOS Debug Command Reference* for more information.

Verifying DNS-Based X.25 Mnemonic Resolution

To verify DNS-based X.25 mnemonic resolution, use the **show hosts** command in EXEC mode. All permanent (perm) entries of type X.121 should be removed from the route table for DNS-based X.25 routing to work.

In the following example, the mnemonic "destination_host" is showing itself to be a permanent entry:

```
Router# show hosts
Default domain is home.com
Name/address lookup uses domain service
Name servers are 10.1.1.40

Host                    Flags      Age Type   Address(es)
destination_host        (perm, OK)  1  X.121  222
```

Configuring X.25 over Frame Relay (Annex G)

Annex G (X.25 over Frame Relay) facilitates the migration of traffic from an X.25 backbone to a Frame Relay backbone by permitting encapsulation of X.25 traffic within a Frame Relay connection. With Annex G, transporting X.25 over Frame Relay has been simplified by allowing direct and transparent X.25 encapsulation over a Frame Relay network. Annex G is supported only on Frame Relay main interfaces (not subinterfaces) and over Frame Relay PVCs. However, X.25 PVC connections are not supported, but only X.25 SVC connections.

X.25 profiles make Annex G easy to configure for both X.25 and LAPB because they consist of bundled X.25 and LAPB commands. Once created and named, X.25 profiles can be simultaneously associated with more than one DLCI connection, using just the profile name. This process means that you need not enter the same X.25 or LAPB commands for each DLCI you are configuring. Multiple Annex G DLCIs can use the same X.25 profile, but the DLCIs can be configured for only one Frame Relay service at a time. The creation of X.25 profiles allows the specification of X.25 and LAPB configurations without the need to allocate hardware interface data block (IDB) information. X.25 profiles do not support IP encapsulation.

Annex G provides multiple logical X.25 SVCs per Annex G link, and modulo 8 and 128 are supported. X.25 Layers 2 and 3 are transparently supported over Annex G. LAPB treats the Frame Relay network like an X.25 network link and passes all of the data and control messages over the Frame Relay network. Before enabling Annex G connections you must establish a Frame Relay connection.

To configure an Annex G connection (assuming you have already configured a Frame Relay connection on your router), use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# x25 profile name`	Creates the X.25 profile.
Step 2	`Router(config)# interface type number`	Configures an interface.
Step 3	`Router(config-if)# encapsulation frame-relay`	Activates Frame Relay encapsulation on each interface that will be using Annex G connections.
Step 4	`Router(config-if)# frame-relay interface-dlci`	Configures the Frame Relay DLCI.
Step 5	`Router(config-fr-dlci)# x25-profile name`	Assigns the named X.25 profile to the DLCI.
Step 6	`Router(config)# x25 routing`	(Optional) Enables X.25 routing of outgoing calls.
Step 7	`Router(config)# x25 route number interface serial-interface dlci number`	(Optional) Assigns an X.25 route for the DLCI on that interface. Required if you want the router to accept switched calls, as well as originating them.

For an example of configuring an Annex G (X.25 over Frame Relay) connection, see the section "X.25 over Frame Relay (Annex G) Example" later in this chapter.

Configuring CMNS Routing

CMNS provides a mechanism through which X.25 services can be extended to nonserial media through the use of packet-level X.25 over frame-level logical link control (LLC2).

Note For information about configuring LLC2 parameters, refer to the chapter "Configuring SDLC and LLC2 Parameters" in the *Cisco IOS Bridging and IBM Networking Configuration Guide*.

The Cisco CMNS implementation permits most X.25 services to be extended across a LAN, although datagram encapsulation and QLLC operations are not available. For example, a DTE host and a Sun workstation can be interconnected via the router's LAN interfaces *and* to a remote OSI-based DTE through a WAN interface to an X.25 packet-switched network (PSN).

To implement CMNS routing, perform the tasks in the following sections:

- Enabling CMNS on an Interface

- Configuring a Route to a CMNS Host

Enabling CMNS on an Interface

To enable CMNS on a nonserial interface, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# cmns enable`	Enables CMNS.

For an example of enabling CMNS on an interface, see the section "CMNS Switching Example" later in this chapter.

Configuring a Route to a CMNS Host

Once CMNS is enabled on a nonserial interface, the router can forward calls over that medium by configuring **x25 route** commands that define the MAC address of each CMNS host that can be reached.

To define routes to CMNS hosts, use the following command—plus pattern and character match options for the **x25 route** command—in interface configuration mode:

Command	Purpose
`Router(config)# x25 route pattern-character match options interface cmns-interface mac mac-address`	Defines route to CMNS host.

Configuring Priority Queueing or Custom Queueing for X.25

Two types of output queueing are available for X.25:

- Priority queueing—Classifies packets on the basis of certain criteria and then assigns the packets to one of four output queues, with high, medium, normal, or low priority.
- Custom queueing—Classifies packets, assigns them to one of 16 output queues, and controls the percentage of available bandwidth for an interface that is used for a queue.

Output queueing for X.25 interfaces differs subtly from its use with other protocols because X.25 is a strongly flow-controlled protocol. Each X.25 VC has an authorized number of packets it can send before it must suspend transmission to await acknowledgment of one or more of the packets that were sent.

Queue processing is also subject to a VC's ability to send data; a high priority packet on a VC that cannot send data will not stop other packets from being sent if they are queued for a VC that can send data. In addition, a datagram that is being fragmented and sent may have its priority artificially promoted if higher-priority traffic is blocked by the fragmentation operation.

Both priority queueing and custom queueing can be defined, but only one method can be active on a given interface.

To configure priority queueing and custom queueing for X.25, perform the following steps:

Step 1 Perform the standard priority and custom queueing tasks *except* the task of assigning a priority or custom group to the interface, as described in the chapters "Configuring Priority Queueing" and "Configuring Custom Queueing" in the *Cisco IOS Quality of Service Solutions Configuration Guide*.

Step 2 Perform the standard X.25 encapsulation tasks, as specified in the section "Configuring an X.25 Datagram Transport" earlier in this chapter.

Step 3 Assign either a priority group or a custom queue to the interface, as described in the chapters "Configuring Priority Queueing" and "Configuring Custom Queueing" in the *Cisco IOS Quality of Service Solutions Configuration Guide*.

Note Connection-oriented VCs (for example, QLLC, PAD, and switched X.25) will use the default queue of the interface. To maintain the correct order, all connection-oriented VCs use a single output queue for sending data.

Configuring X.25 Closed User Groups

A closed user group (CUG) is a collection of DTE devices for which the network controls access between two members and between a member and a nonmember. An X.25 network can support up to 10,000 CUGs (numbered from 0 to 9999), each of which can have any number of member DTE devices. An individual DTE becomes a member of a specific network CUG by subscription. The subscription data includes the local number the DTE will use to identify the network CUG (which may or may not be the same as the network number, as determined by network administration and the requirements of the DTE device), and any restriction that prohibits the DTE from placing a call within the CUG or, conversely, prohibits the network from presenting a call within the CUG to the DTE device.

The X.25 DCE interfaces of the router can be configured to perform the standard CUG access controls normally associated with a direct attachment to an X.25 network POP. The DCE interface of the router acts as the boundary between the DTE and the network, and CUG use ensures that only those incoming and outgoing SVCs consistent with the configured CUG subscriptions are permitted. X.25 CUG configuration commands on the router are specified at every POP, and CUG security decisions are made solely from those commands. However, CUG service is not supported on XOT connections.

CUG security depends on CUG decisions made by the two POPs used to connect an SVC through the network, so CUG security depends on the collective configuration of all POPs that define the network boundary. The standalone interface configuration determines if the POP will permit user access for a given incoming or outgoing call within the authorized CUG.

CUGs are a network service designed to allow various network subscribers (DTE devices) to be segregated into private subnetworks with limited incoming or outgoing access. This means that a DTE must obtain membership from its network service (POP) for the set of CUGs it needs access to. A DTE may subscribe to zero, one, or several CUGs at the same time. A DTE that does not require CUG membership for access is considered to be in the open part of the network. Each CUG typically permits subscribing users to connect to each other, but precludes connections with nonsubscribing DTE devices.

However, CUG behavior is highly configurable. For instance, a CUG configuration may subscribe a DTE to a given CUG, but bar it from originating calls within the CUG or, conversely, bar it from receiving calls identified as being within the CUG. CUG configuration can also selectively permit the DTE to originate calls to a DTE on the open network, or permit the DTE to receive calls from a DTE on the open network.

CUG access control is first applied when the originating DTE places a call to the POP, and again when the POP of the destination DTE device receives the call for presentation. Changes to the POP CUG subscriptions will not affect any SVCs that have already been established.

When a DTE belongs to more than one CUG, it must specify its preferential CUG, unless a call is specifically aimed at devices outside the CUG network. However, the number of CUGs to which a DTE can belong depends on the size of the network. Unsubscribing from one CUG or the overall CUG service will not result in the termination of the SVC connections.

CUG behavior is a cooperative process between two network devices. The DCE offers this service to the connecting subscribers via the DTE device. There is no global database regarding CUG membership; therefore, the Cisco router uses information configured for the various X.25 devices and the encoded CUG information in the outgoing and incoming packets.

X.25 CUGs are used for additional X.25 access protection and security. In a setup where DTE devices are attached to a PDN, you can derive a private subnetwork by subscribing your DTE devices to a set of CUGs, which allows closer control of your DTE devices, such as permitting or restricting which DTE can talk to other DTE devices and for what particular purpose. For example, a distinct CUG can be defined to handle each of the different modes of connectivity, such as the following:

- Datagram encapsulation operation among all company sites
- PAD services for customers seeking public information
- PAD services for system administration internal access to consoles
- QLLC access restricted to the company financial centers

One site could have different CUG subscriptions, depending on connectivity requirements. These sites could all have restrictions regarding which other company devices can be reached (within a CUG), whether a device is permitted to call the open network for a given function, and whether a public terminal can access the device for a given function.

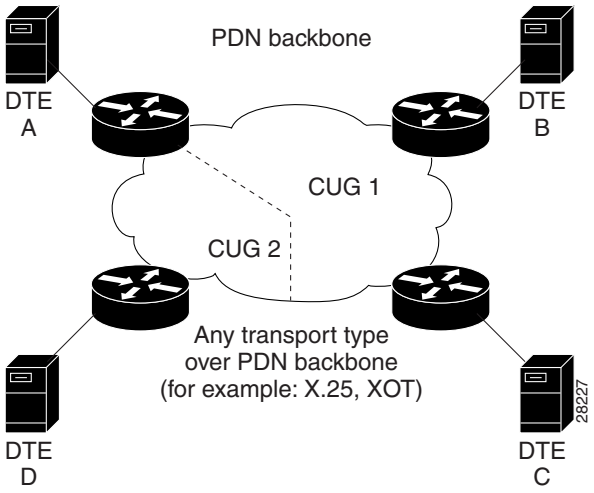
By default, no CUG behavior is implemented. Therefore, in order to observe CUG restrictions, all users attached to the network must be subscribed to CUG behavior (CUG membership) even if they are not subscribed to a specific CUG.

Figure 5 shows two CUGs (CUG 1 and CUG 2). DTE devices A, B, and C are members of CUG 1. They can initiate and receive calls only from the other members of CUG 1. They are therefore members of a private subnet with no access to other DTE devices. DTE A is also a member of CUG 2 with DTE D, but DTE D cannot send calls to or receive calls from DTE B or DTE C. The router checks each received call to determine if it is intended for their CUG. If not, the router rejects the call.

You can subscribe to multiple CUGs per interface, but each CUG that is permitted must be specifically configured. All CUGs are sorted by their local identifier. The main limitation to the number of CUGs configured is the amount of nonvolatile memory to store the configuration. Having subscribed to a CUG, the DTE indicates which CUG is being called. If the DTE does not indicate a CUG, its DCE determines which CUG is used and if the call should be allowed.

Note CUG service is implemented at the DCE interface, which means that it specifies a network function. For a summary of DCE operations, refer to *ITU-T 1996 Recommendation X.301* tables 7-6 and 7-8.

Figure 5 DTE Devices A, B, C, and D Connecting to CUGs 1 and 2 over a PDN



Understanding CUG Configuration

Answering the following questions will help you set up your CUG service and CUGs:

- Do you want to permit incoming public access to the DTE device?

  If so, configure the **x25 subscribe cug-service incoming-access** command on the DCE so that the CUG service from the open network allows incoming calls to the DTE device.

- Do you want to permit outgoing public access for the DTE device?

  If so, configure the **x25 subscribe cug-service outgoing-access** command on the DCE so that the CUG service allows public outgoing calls from the DTE to the open network.

- Will the CUG users require restricted access to the PDN?

  If so, configure the **x25 subscribe local-cug** command for mapping the local CUG to the network CUG for the same CUG entity. To obtain full access to the PDN, the CUG service will need to be subscribed to by both incoming and outgoing access.

  If you want a secure CUG with no access to the PDN, subscribe the CUG to no incoming or outgoing access, and configure it to communicate only with other attachments within CUGs that it has defined.

  After establishing that you want PDN CUG access, you must then answer the following questions:

  – Can the user place calls within the CUG?

    The default is set for users to be able to place calls. If you do not want this setting, use the **no-outgoing** keyword.

  – Can the user receive calls within the CUG?

    The default is set for users to be able to receive calls. If you do not want this setting, use the **no-incoming** keyword.

  – Do you want a subscribed CUG to be assumed when a CUG member places a call without specifying a CUG?

    If so, use the **preferential** keyword.

Point of Presence

X.25 is not a POP by default, and POP behavior does not automatically enforce CUG security. Within PDNs, all devices are connected by POPs, which are open entry points into a network and, as such, pose a potential security risk.

When you enable X.25 CUG service, you are configuring your network like a PDN, and so for every POP with attachments in the network you must configure CUG security. CUG security is particularly important on those POPs that do not subscribe to CUGs, because they could act as a "back door" into your CUGs.

Note If you do not configure CUG security on your network POPs, you are creating a security risk for your network. Configuration must be done manually for every POP in your network.

CUG Membership Selection

CUG membership selection occurs from the calling DTE in an outgoing (call request) packet to specify the CUG membership selected for the call. CUG membership selection is requested or received by a DTE only after the DTE has subscribed to one or more of the following facilities:

- Relevant CUG service
- Outgoing access CUG, which allows the source DTE to identify the CUG within which it is placing the call
- Incoming access CUG, which allows the destination DTE to identify the CUG to which both DTE devices belong

See the following sections for details of different types of CUG membership:

- Preferential CUGs
- Incoming and Outgoing Access CUGs
- Incoming and Outgoing Calls Barred Within a CUG

Preferential CUGs

A DTE that subscribes to more than one CUG (and permits neither incoming nor outgoing access from or to the open network) must designate a preferential CUG. Its use is assumed when no CUG selection is enabled in the outgoing call (call request) or incoming call. Using a preferential CUG achieves a higher level of security. Preferential CUG designation is for DTE devices meant to operate without requiring a CUG selection facility in every outgoing call, or for DTE devices not capable of encoding a CUG selection.

Preferential CUG designation options are as follows:

- If no preferential CUG has been designated and a CUG member presents a call without specifying a receiving CUG, the call will be rejected, unless incoming access from the open network is configured.
- If a preferential CUG has been designated and the user presents a call without specifying a CUG, the call will be directed to the preferential CUG.
- If outgoing access is permitted on your CUG and you present an outgoing call without designating a preferential CUG, then your CUG assumes the call is meant either for the open network or for the preferential CUG.
- A single CUG specified at a DCE interface is treated as the preferential CUG.

Incoming and Outgoing Access CUGs

CUG service with incoming access allows you to receive incoming calls from the open part of the network and from DTE devices belonging to other outgoing access CUGs. If the DTE does not subscribe to incoming access, any incoming call without the CUG membership selection facility will not be accepted.

A CUG with outgoing access allows you to make outgoing calls to the open part of the network and to DTE devices with incoming access capability. Subscribing to the outgoing access CUG allows a DTE to belong to one or more CUGs and to originate calls to DTE devices in the open part of the network (DTE devices not belonging to any CUGs) and to DTE devices belonging to incoming access CUGs. If the DTE has not subscribed to outgoing access, the outgoing packets must contain a valid CUG membership selection facility. If a CUG membership selection facility is not present, the local DCE defaults to the preferential CUG, or rejects the call if a preferential CUG is not specified.

Incoming and Outgoing Calls Barred Within a CUG

When a DTE wishes to initiate only outgoing calls, it specifies "incoming calls barred." With this CUG option subscribed to, a subscriber DTE is permitted only to originate calls and not to receive calls within the CUG. The DCE will clear an incoming call before it reaches the DTE.

If a DTE subscribes to the "outgoing calls barred" option, it is permitted to receive calls but not to originate calls within the CUG. An attempted outgoing call will be cleared by the DCE, which in turn will notify the DTE of its actions.

X.25 CUG Configuration Task List

To configure X.25 CUGs, perform the tasks in the following sections. Each section is identified as required or optional.

- Configuring X.25 CUG Service, Access, and Properties (Required)
- Configuring a POP with No CUG Access (Optional)
- Configuring a POP with Access Restricted to One CUG (Optional)
- Configuring a POP with Multiple CUGs and No Public Access (Optional)
- Configuring a POP with Multiple CUGs and Public Access (Optional)
- Configuring CUG Selection Facility Suppression (Optional)
- Verifying X.25 CUG Service (Optional)
- Troubleshooting Tips for X.25 CUG Service (Optional)

Configuring X.25 CUG Service, Access, and Properties

Note If you do not want to enable the **x25 subscribe cug-service** command, you will be subscribed to CUG service automatically the first time you subscribe to a CUG (using the **x25 subscribe local-cug** command), with CUG service default settings of no incoming and no outgoing access.

You must establish X.25 DCE encapsulation and X.25 CUG service on the interface to enable this feature. Within the **x25 subscribe cug-service** command, establish the type of CUG public access (incoming or outgoing) you want. If you do not enter this command, the default will be enabled.

To set up the individual CUGs, use the **x25 subscribe local-cug** command to specify each local CUG and map it to a network CUG, setting the access properties of the local CUG—no-incoming, no-outgoing, preferential, all, or none—at the same time.

To configure X.25 CUG service, access, and properties, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface number`	Selects the interface to be configured.
Step 2	`Router(config-if)# encapsulation x25 dce`	Enables X.25 DCE network operation.
Step 3	`Router(config-if)# x25 subscribe cug-service [incoming-access | outgoing-access]`	Enables and controls standard CUG behavior on an X.25 DCE interface.
Step 4	`Router(config-if)# x25 subscribe local-cug number network-cug number [no-incoming | no-outgoing | preferential]`	Maps the desired local CUG number to its corresponding network CUG.

For an example of configuring X.25 CUG service, access and properties, see the section "X.25 CUG Service, Access, and CUG Properties Example" at the end of this chapter.

Configuring a POP with No CUG Access

Caution This configuration is critical to enforcing full CUG security on your network. You must conduct this configuration on every POP in your network. If you do not configure this for all POPs in your network, you will not have a secure network, and a security breach could occur.

With the POP configuration of no individual CUG subscriptions, the POP is a member of the open network. Even though it does not have a CUG attached, you must configure CUG security on it to ensure that the rest of your network remains secure. The POP has CUG incoming access and outgoing access permitted—the least restrictive setting. The POP will allow calls that do not require CUG authorization to and from the open network, but it will refuse any CUG-specified calls because the POP does not belong to a CUG. A call from an intranetwork connection with no CUG selected is permitted as incoming access from the open network, but a call that requires CUG access will be refused.

To configure a POP with no CUG access, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface number`	Selects the interface to be configured.
Step 2	`Router(config-if)# encapsulation x25 dce`	Enables X.25 DCE network operation.
Step 3	`Router(config-if)# x25 subscribe cug-service incoming-access outgoing-access`	Permits incoming and outgoing CUG access on an X.25 DCE interface.

For an example of configuring a POP with no CUG access, see the section "POP with No CUG Access Example" at the end of this chapter.

Configuring a POP with Access Restricted to One CUG

In the POP configuration with one CUG subscribed, it is important to have no public access permitted on it. You do this by configuring the default setting (no incoming and no outgoing access) for the **x25 subscribe cug-service** command. When an outgoing call not specifying a CUG is made, the POP assumes the call to be for its one subscribed CUG. An incoming call that does not specify that CUG is rejected. This single CUG configuration assumes the CUG to be the preferential CUG.

To configure a POP with access restricted to one CUG, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface number`	Selects the interface to be configured.
Step 2	`Router(config-if)# encapsulation x25 dce`	Enables X.25 DCE network operation.
Step 3	`Router(config-if)# x25 subscribe cug-service`	Sets default behavior on an X.25 DCE interface.
Step 4	`Router(config-if)# x25 subscribe local-cug number network-cug number [no-incoming | no-outgoing | preferential]`	Maps the desired local CUG number to its corresponding network CUG.

For an example of configuring a POP with access restricted to one CUG, see the section "POP with Access Restricted to One CUG Example" at the end of this chapter.

Configuring a POP with Multiple CUGs and No Public Access

With the POP configuration of multiple CUGs and no public access permitted, the only difference from the POP configuration with one subscribed CUG is that one of the CUGs must be chosen as preferential. If you do not specify a preferential CUG, no calls can be made or accepted. Notice the omission of the keywords from the **x25 subscribe cug-service** command. This omission enables the default settings of no incoming and no outgoing access.

To configure a POP with multiple CUGs and no public access, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface number`	Selects the interface to be configured.
Step 2	`Router(config-if)# encapsulation x25 dce`	Enables X.25 DCE network operation.
Step 3	`Router(config-if)# x25 subscribe cug-service`	Sets default CUG behavior on an X.25 DCE interface.
Step 4	`Router(config-if)# x25 subscribe local-cug number network-cug number [no-incoming | no-outgoing | preferential]`	Maps the desired local CUG number to its corresponding network CUG.
Step 5	`Router(config-if)# x25 subscribe local-cug number network-cug number [no-incoming | no-outgoing | preferential]`	Configures another CUG interface.

For an example of configuring a POP with multiple CUGs and no public access, see the section "POP with Multiple CUGs and No Public Access Example" at the end of this chapter.

Configuring a POP with Multiple CUGs and Public Access

The least restrictive POP configuration is a POP configured to allow public access to members of several CUG and to originate and receive calls from the open network (that is, to or from users that do not subscribe to one of the CUGs to which this POP subscribes). Configuring the POP with multiple CUGs and public access is achieved using the **x25 subscribe cug-service** command with the addition of the keywords **incoming-access** and **outgoing-access** to allow calls to be made and received to and from outside hosts not in the specified CUG network.

To set up the individual CUGs, use the **x25 subscribe local-cug** command to specify each local CUG and map it to a network CUG, setting the access properties of the local CUG—no-incoming, no-outgoing, preferential, all, or none—at the same time.

An outgoing call may select any of the local CUGs or not. When no CUG is selected, it is assumed that the call is intended for the open network. The call will be refused if it specifies a local CUG different from the one to which the POP is subscribed. An incoming call may or may not select related network CUGs. If no CUG is selected, the call is accepted as coming from the open network. A call that requires access to a different CUG will be refused.

To configure a POP with multiple CUGs and public access, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface number`	Selects the interface to be configured.
Step 2	`Router(config-if)# encapsulation x25 dce`	Enables X.25 DCE network operation.
Step 3	`Router(config-if)# x25 subscribe cug-service incoming-access outgoing-access`	Permits incoming and outgoing CUG access on an X.25 DCE interface.
Step 4	`Router(config-if)# x25 subscribe local-cug number network-cug number [no-incoming | no-outgoing | preferential]`	Maps the desired local CUG number to its corresponding network CUG.
Step 5	`Router(config-if)# x25 subscribe local-cug number network-cug number [no-incoming | no-outgoing | preferential]`	Configures another CUG interface.

For an example of configuring a POP with multiple CUGs and public access, see the section "POP with Multiple CUGs and Public Access Example" at the end of this chapter.

Configuring CUG Selection Facility Suppression

A CUG selection facility is a specific encoding element that can be presented in a call request or an incoming call. A CUG selection facility in a call request allows the source DTE to identify the CUG within which it is placing the call. A CUG selection facility in an incoming call allows the destination DTE to identify the CUG to which both DTEs belong.

You can configure an X.25 DCE interface or X.25 profile with a DCE station type to selectively remove the CUG selection facility before presenting an incoming call packet to a subscribed DTE. The CUG selection facility can be removed from incoming call packets destined for the preferential CUG only or for all CUGs. You can also remove the selection facility from a CUG with outgoing access (CUG/OA). The CUG selection facility suppression mechanism does not distinguish between CUGs and CUG/OAs.

Note The CUG Selection Facility Suppress Option feature will not in any way compromise CUG security.

CUG selection facility supression is supported by X.25 over Frame Relay (Annex G). If Annex G is being used, you must configure CUG selection facility supression in an X.25 profile.

To configure X.25 CUG selection facility suppression, perform the tasks in the following sections:

- Configuring CUG Selection Facility Suppression on an Interface
- Configuring CUG Selection Facility Suppression on an X.25 Profile

Configuring CUG Selection Facility Suppression on an Interface

To configure X.25 CUG selection facility suppression on an interface, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface type number`	Configures an interface type and enters interface configuration mode.
Step 2	`Router(config-if)# encapsulation x25 dce`	Specifies that a serial interface will operate as an X.25 DCE device.
Step 3	`Router(config-if)# x25 subscribe cug-service [incoming-access | outgoing-access] [suppress preferential | suppress all]`	Enables and controls standard CUG behavior on an X.25 DCE interface.

For an example of CUG selection facility suppression on an interface, see the section "CUG Selection Facility Suppression for the Preferential CUG Example" later in this chapter.

Configuring CUG Selection Facility Suppression on an X.25 Profile

To configure X.25 CUG selection facility suppression on an X.25 profile, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# x25 profile name dce`	Configures an X.25 profile and specifies a DCE station type.
Step 2	`Router(config-x25)# x25 subscribe cug-service [incoming-access | outgoing-access] [suppress preferential | suppress all]`	Enables and controls standard CUG behavior on an X.25 DCE interface.

For an example of CUG selection facility suppression on an X.25 profile, see the section "CUG Selection Facility Suppression for All CUGs Example" later in this chapter.

Verifying X.25 CUG Service

To show current settings of the X.25 CUGs feature, use the **show x25 cug** (either keyword **local-cug** or **network-cug** must be designated) command in EXEC mode. In the following example local CUGs 100, 200, 300, and 5000 are shown mapped to their related network CUGs 11, 22, 33, and 55, respectively, all with incoming and outgoing public access, and with network CUG 55 being set as the preferential:

```
Router# show x25 cug local-cug
X.25 Serial0, 4 CUGs subscribed with incoming and outgoing public access
  local-cug 100 <-> network-cug 11
  local-cug 200 <-> network-cug 22
```

```
local-cug 300 <-> network-cug 33
local-cug 5000 <-> network-cug 55, preferential
```

Troubleshooting Tips for X.25 CUG Service

You can use **debug x25 events** command to verify if and when CUG calls are being made and how the CUGs are behaving. The following example shows messages concerning a rejection of a call by a DCE because CUG 40 is not configured at the DCE interface, either by design or by administrative mistake:

```
Router# debug x25 events
00:48:33:Serial1:X.25 I R1 Call (14) 8 lci 1024
00:48:33:  From (3):111 To (3):444
00:48:33:  Facilities:(2)
00:48:33:    Closed User Group (basic):40
00:48:33:  Call User Data (4):0x01000000 (pad)
00:48:33:X.25 Incoming Call packet, Closed User Group (CUG) protection, selected network
CUG not subscribed
00:48:33:Serial1:X.25 O R1 Clear (5) 8 lci 1024
00:48:33:  Cause 11, Diag 65 (Access barred/Facility code not allowed)
```

Configuring DDN or BFE X.25

The Defense Data Network (DDN) X.25 protocol has two versions: Basic Service and Standard Service. Cisco System's X.25 implementation supports only the Standard Service which also includes Blacker Front End (BFE).

DDN X.25 Standard Service requires that the X.25 data packets carry IP datagrams. The DDN packet switching nodes (PSNs) can extract the IP datagram from within the X.25 packet and pass data to another Standard Service host.

The DDN X.25 Standard is the required protocol for use with DDN PSNs. The Defense Communications Agency (DCA) has certified Cisco Systems' DDN X.25 Standard implementation for attachment to the Defense Data Network. As part of the certification, Cisco IOS software is required to provide a scheme for dynamically mapping Internet addresses to X.121 addresses. See the section "Understanding DDN X.25 Dynamic Mapping" that follows for details on that scheme.

To enable DDN X.25 service, refer to the following sections:

- Understanding DDN X.25 Dynamic Mapping
- Enabling DDN X.25
- Defining IP Precedence Handling

To enable BFE X.25 service, perform the task in the following section:

- Configuring Blacker Front End (BFE) X.25

Understanding DDN X.25 Dynamic Mapping

The DDN X.25 standard implementation includes a scheme for dynamically mapping all classes of IP addresses to X.121 addresses without a table. This scheme requires that the IP and X.121 addresses conform to the formats shown in Figure 6 and Figure 7. These formats segment the IP addresses into network (N), host (H), logical address (L), and PSN (P) portions. For the BFE encapsulation, the IP address is segmented into Port (P), Domain (D), and BFE ID number (B). The DDN algorithm requires that the host value be less than 64.

Figure 6 DDN IP Address Conventions

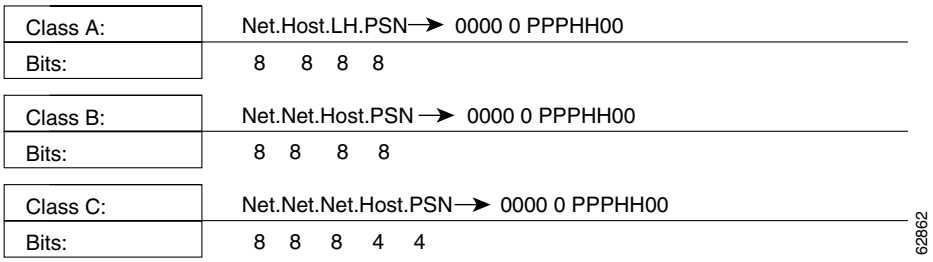
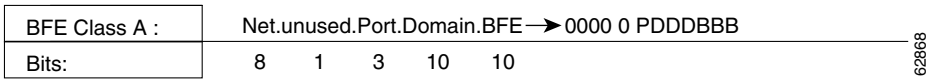


Figure 7 BFE IP Address Conventions



The DDN conversion scheme uses the host and PSN portions of an IP address to create the corresponding X.121 address. The DDN conversion mechanism is limited to Class A IP addresses; however, the Cisco IOS software can convert Class B and Class C addresses as well. As indicated, this method uses the last two octets of a Class B address as the host and PSN identifiers, and the upper and lower four bits in the last octet of a Class C address as the host and PSN identifiers, respectively. The BFE conversion scheme requires a Class A IP address.

The DDN conversion scheme uses a physical address mapping if the host identifier is numerically less than 64. (This limit derives from the fact that a PSN cannot support more than 64 nodes.) If the host identifier is numerically larger than 64, the resulting X.121 address is called a *logical address*. The DDN does not use logical addresses.

The format of physical DDN X.25/X.121 addresses is ZZZZFIIIHHZZ(SS). Each character represents a digit, described as follows:

- ZZZZ represents four zeros.
- F is zero to indicate a physical address.
- III represents the PSN octet from the IP address padded with leading zeros.
- HH is the host octet from the IP address padded with leading zeros.
- ZZ represents two zeros.
- (SS) represents the optional and unused subaddress.

The physical and logical mappings of the DDN conversion scheme always generate a 12-digit X.121 address. Subaddresses are optional; when added to this scheme, the result is a 14-digit X.121 address. The DDN does not use subaddressing.

Packets using routing and other protocols that require broadcast support can successfully traverse X.25 networks, including the DDN. This traversal requires the use of network protocol-to-X.121 maps, because the router must know explicitly where to deliver broadcast datagrams. (X.25 does not support broadcasts.) You can mark network protocol-to-X.121 map entries to accept broadcast packets; the router then sends broadcast packets to hosts with marked entries. For DDN or BFE operation, the router generates the interface X.121 addresses from the interface IP address using the DDN or BFE mapping technique.

Enabling DDN X.25

Both DCE and DTE operation causes the Cisco IOS software to specify the Standard Service facility in the Call Request packet, which notifies the PSNs to use Standard Service.

To enable DDN X.25, use one of the following commands in interface configuration mode, as appropriate for your network:

Command	Purpose
`Router(config-if)# encapsulation x25 ddn`	Sets DDN X.25 DTE operation.
`Router(config-if)# encapsulation x25 dce ddn`	Sets DDN X.25 DCE operation.

For an example of enabling DDN X.25, see the section "DDN X.25 Configuration Example" later in this chapter.

Defining IP Precedence Handling

Using Standard Service, the DDN can be configured to provide separate service for datagrams with high precedence values. When IP precedence handling is enabled, the router uses a separate X.25 SVC to handle each of four precedence classes of IP traffic—routine, priority, immediate, and other. An IP datagram is transmitted only across the SVC that is configured with the appropriate precedence.

By default, the DDN X.25 software opens one VC for all types of service values. To enable the precedence-sensitivity feature, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# x25 ip-precedence`	Allows a new VC based on the type of service (TOS) field.

Verify that your host does not send nonstandard data in the TOS field. Nonstandard data can cause multiple, wasteful VCs to be created.

Configuring Blacker Front End (BFE) X.25

For environments that require a high level of security, the Cisco IOS software supports attachment to Defense Data Network (DDN) Blacker Front End (BFE) equipment.

BFE encapsulation operates to map between Class A IP addresses and the X.121 addresses expected by the BFE encryption device.

To set BFE encapsulation on the router attached to a BFE device, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# encapsulation x25 bfe`	Sets BFE encapsulation on the router attached to a BFE device.

For an example of configuring Blacker Front End, see the section "Blacker Front End Example" at the end of this chapter.

Configuring X.25 Remote Failure Detection

X.25 remote failure detection is important because after a primary link failure, the router can establish a secondary link and continue sending data. The router detects a call failure and uses a secondary route to send subsequent packets to the remote destination, at the same time making periodic attempts to reconnect to its primary link. The number of these attempts and the interval between such attempts is controlled using the **x25 retry** command. The failed link is marked up again when any of the following occurs:

- An attempt to reestablish the link via the retry mechanism is successful.
- An incoming call is received on the subinterface.
- The X.25 packet layer on the interface is restarted.

X.25 remote failure detection needs to be manually configured on each intended subinterface. However, because it is a per-destination configuration rather than a per-user configuration, you need it enabled only on the subinterface requiring the retry option—typically your primary interface. This feature is not automatically enabled and only responds to failed outgoing call attempts. The feature applies only to point-to-point subinterfaces and works only on SVCs. It is not necessary if you are running IP routing, because IP routing already implements alternate routing. This feature is targeted at environments that have static IP routing across an X.25 network, where these static IP routes currently need to be manually added to the route tables.

The **x25 retry** command is activated by a call failure notification. Retry occurs only with calls initiated on a subinterface configured with the **x25 retry** command. This command works only when no VCs are up. When reconnection occurs, traffic begins to reuse the primary interface. This resetting of the line protocol to up is the last activity that the **x25 retry** command conducts. Issuing the **clear x25** command on the remote failure detection configured interface, or receiving a call during retry, will disable the **x25 retry** and the subinterface will be marked "up." An incoming call can be conducted in a way similar to how the **ping** command is used to check connectivity (by definition, a successful incoming call indicates that connectivity is functioning). Also, if the router reaches its retry attempts limit, the **x25 retry** command will discontinue and the subinterface will remain down.

X.25 remote failure detection is designed to work with any network layer routed protocol. However, the feature depends on the ability of the protocol to handle more than one static route to the same destination at the same time. Currently, only IP can accomplish this multistatic route handling.

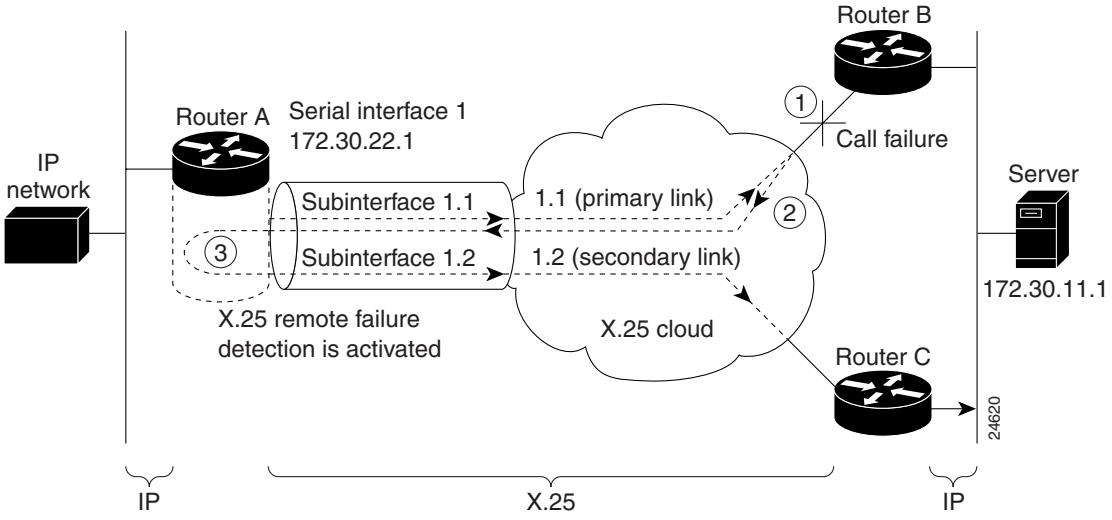
Alternatively, X.25 remote failure detection can be used to activate a backup link should the subinterface configured for retry be marked down via the retry mechanism. See the "X.25 Remote Failure Detection and the Backup Interface" configuration tasks for further details.

The benefits of this feature are network cost savings because IP routing updates (requiring dynamic but costly network connectivity) are not necessary; improved responsiveness and versatility of X.25 primary and alternate links; and more robust networking options for data transmission.

Figure 8 shows how X.25 remote failure detection works:

1. The data cannot reach its destination using its primary route.
2. A call failure notification is sent to the transmitting router.
3. The **x25 retry** command is activated, and IP then activates the preassigned secondary route in its route table and begins sending data. The **x25 retry** command also shuts down subinterface 1.1 and begins its retry attempts on this link.

Figure 8 X.25 Remote Failure Detection in Action over an X.25 Cloud



For examples of configuring remote failure detection, see the "X.25 Remote Failure Detection Examples" section, containing the "X.25 Remote Failure Detection with IP Static Routes Example" and "X.25 Remote Failure Detection and the Backup Interface Example," later in this chapter.

X.25 Remote Failure Detection with IP Static Routes

To configure X.25 remote failure detection with IP static routes, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface number`	Enters specified interface configuration mode.
Step 2	`Router(config-if)# encapsulation x25`	Enables X.25 encapsulation on the interface.
Step 3	`Router(config-if)# x25 address x121-address`	Sets X.121 address of the network interface.
Step 4	`Router(config-if)# interface subinterface number point-to-point`	Enters specified subinterface and enables point-to-point for it.
Step 5	`Router(config-subif)# ip address address mask`	Creates IP address and mask for the subinterface.
Step 6	`Router(config-subif)# x25 map ipaddress x121address`	Maps IP address to an X.121 address.
Step 7	`Router(config-subif)# x25 retry interval seconds attempts count`	Enables the X.25 retry option on the subinterface.
Step 8	`Router(config)# ip route address mask serial subinterface number weight`	Configures static route from point-to-point interface specified to a destination.
Step 9	`Router(config)# ip route address mask serial nextsubinterface number weight`	Configures static route from next point-to-point interface specified for the same destination.

X.25 Remote Failure Detection and the Backup Interface

To configure X.25 remote failure detection and create a backup interface, use the following commands beginning in global configuration mode. Note that IP static routes need not be configured because this backup route is being only configured as a secondary route.

	Command	Purpose
Step 1	`Router(config)# interface number`	Enters specified interface configuration mode.
Step 2	`Router(config-if)# encapsulation x25`	Enables X.25 encapsulation on the interface.
Step 3	`Router(config-if)# x25 address x121-address`	Sets X.121 address of the network interface.
Step 4	`Router(config)# interface subinterface number point-to-point`	Enters specified subinterface and configures point-to-point for it.
Step 5	`Router(config-subif)# ip address address mask`	Creates IP address and mask for the subinterface.
Step 6	`Router(config-subif)# x25 map ipaddress x121address`	Maps IP address to an X.121 address.
Step 7	`Router(config-subif)# x25 retry interval seconds attempts count`	Enables the X.25 retry option on the subinterface.
Step 8	`Router(config-subif)# backup interface serial number`	Configures specified interface as the backup.
Step 9	`Router(config)# interface number`	Enters specified interface configuration mode to configure the backup.
Step 10	`Router(config-if)# encapsulation x25`	Enables X.25 encapsulation on the interface.
Step 11	`Router(config-if)# x25 address x121-address`	Sets X.121 address of the network interface.
Step 12	`Router(config-if)# ip address address mask`	Creates IP address and mask for the subinterface.
Step 13	`Router(config-if)# x25 map ipaddress x121address`	Maps IP address to an X.121 address.

For examples of configuring remote failure detection, see the section "X.25 Remote Failure Detection Examples," containing the "X.25 Remote Failure Detection with IP Static Routes Example" and "X.25 Remote Failure Detection and the Backup Interface Example," later in this chapter. For verification, see the section "Verifying X.25 Remote Failure Detection," next.

Verifying X.25 Remote Failure Detection

To verify X.25 remote failure detection, use the **show interfaces serial** command on the interface with the **x25 retry** command configured. The last line in the following output shows the X.25 retry mechanism currently in action on subinterface 1.1, which is currently down—as indicated by the "(retry in progress)" statement—and which has "tried" one out of its possible 100 retry attempts.

```
Router# show interfaces serial1
Serial1 is up, line protocol is up
  Hardware is QUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation X25, loopback not set
  X.25 DTE, address 11111, state R1, modulo 8, timer 0
      Defaults:idle VC timeout 0
        cisco encapsulation
        input/output window sizes 2/2, packet sizes 128/128
      Timers:T20 180, T21 200, T22 180, T23 180
      Channels:Incoming-only none, Two-way 1-1024, Outgoing-only none
```

```
      RESTARTs 2/0 CALLs 0+0/0+0/0+0 DIAGs 0/0
Interface Serial1.1:retry-interval 5, attempts 100, tried 1 (retry in progress)
```

To verify which route is currently in use by IP, use the **show ip route** command.

The **debug x25 events** command can be also activated, so that you can see a call being attempted by the X.25 retry mechanism every configured interval.

Creating X.29 Access Lists

Protocol translation software supports access lists, which make it possible to limit access to the access server from X.25 hosts. Access lists take advantage of the message field defined by Recommendation X.29, which describes procedures for exchanging data between two PADs or between a PAD and a DTE device.

To create and enable access lists, perform the tasks in the following sections:

- Creating an X.29 Access List
- Applying an Access List to a Virtual Terminal Line

When configuring protocol translation, you can specify an access list number with each **translate** command. When translation sessions result from incoming PAD connections, the corresponding X.29 access list is used. Refer to the *Cisco IOS Dial Technologies Command Reference* for more information about the **translate** command.

For an example of defining an X.29 access list, see the section "X.29 Access List Example" at the end of this chapter.

Creating an X.29 Access List

To specify the access conditions, use the following command beginning in global configuration mode:

Command	Purpose
`Router(config)# x29 access-list access-list-number {deny | permit} x121-address`	Restricts incoming and outgoing connections between a particular vty (into a Cisco access server) and the addresses in an access list.

An access list can contain any number of lines. The lists are processed in the order in which you type the entries. The first match causes the permit or deny condition. If an X.121 address does not match any of the entries in the access list, access is denied.

Applying an Access List to a Virtual Terminal Line

To apply an access list to a virtual line, use the following command in line configuration mode:

Command	Purpose
`Router(config)# access-class access-list-number in`	Restricts incoming and outgoing connections between a particular vty (into a Cisco access server) and the addresses in an access list.

The access list number is used for incoming TCP access, for incoming local-area transport (LAT) access, and for incoming PAD access. For TCP access, the protocol translator uses the defined IP access lists. For LAT access, the protocol translator uses the defined LAT access list. For incoming PAD connections, the protocol translator uses an X.29 access list. If you want to have access restrictions only on one of the protocols, you can create an access list that permits all addresses for the other protocol.

Creating an X.29 Profile Script

You can create an X.29 profile script for use by the **translate** command. When an X.25 connection is established, the protocol translator then acts as if an X.29 Set Parameter packet had been sent that contained the parameters and values set by this command.

To create an X.29 profile script, use the following command beginning in global configuration mode:

Command	Purpose
`Router(config)# x29 profile {default | name} parameter:value [parameter:value]`	Creates an X.29 profile script.

For an example of a profile script, see the section "X.29 Profile Script Example" at the end of this chapter.

Monitoring and Maintaining LAPB and X.25

To monitor and maintain X.25 and LAPB, use any of the following commands in EXEC mode:

Command	Purpose
`Router# clear x25 {serial number | cmns-interface mac-address} [vc-number]`	Clears an SVC, restarts an X.25 or CMNS service, or resets a PVC.
`Router# clear xot remote ip-address port local ip-address port`	Clears an XOT SVC or resets an XOT PVC.
`Router# show cmns [type number]`	Displays CMNS information.
`Router# show interfaces serial number`	Displays operation statistics for an interface.
`Router# show llc2`	Displays CMNS connections over LLC2.
`Router# show x25 interface [serial number | cmns-interface mac mac-address]`	Displays information about VCs on an X.25 interface (a serial interface) or a CMNS interface (an Ethernet, Token Ring, or FDDI interface).
`Router# show x25 map`	Displays the protocol-to-X.121 address map.
`Router# show x25 remote-red`	Displays the one-to-one mapping of the IP addresses of the host and the IP addresses of the remote BFE device.
`Router# show x25 route`	Displays routes assigned by the **x25 route** command.
`Router# show x25 services`	Displays information about X.25 services.
`Router# show x25 vc [lcn]`	Displays details of active VCs.
`Router# show x25 xot [local ip-address [port port]] [remote ip-address [port port]]`	Displays information for all XOT VCs or, optionally, for VCs that match a specified set of criteria.

Note See the appendix "X.25 Cause and Diagnostic Codes" in the *Cisco IOS Debug Command Reference* for a description of PVC states that can appear in these **show command** displays.

X.25 and LAPB Configuration Examples

The following sections provide examples to help you understand how to configure LAPB and X.25 for your network:

- Typical LAPB Configuration Example
- Transparent Bridging for Multiprotocol LAPB Encapsulation Example
- Typical X.25 Configuration Example
- VC Ranges Example
- X.25 Failover Example
- PVC Switching on the Same Router Example
- X.25 Route Address Pattern Matching Example
- X.25 Routing Examples
- PVC Used to Exchange IP Traffic Example
- Point-to-Point Subinterface Configuration Example
- Simple Switching of a PVC over XOT Example
- PVC Switching over XOT Example
- X.25 Load Balancing Examples
- X.25 Switching Between PVCs and SVCs Example
- Inserting and Removing X.121 Addresses As Calls Are Routed Example
- Forwarding Calls Using the continue Keyword Example
- DNS-Based X.25 Routing Example
- X.25 over Frame Relay (Annex G) Example
- CMNS Switching Example
- CMNS Switching over a PDN Example
- CMNS Switched over Leased Lines Example
- Configuring Local Acknowledgment Example
- Setting Asymmetrical Window and Packet Sizes Flow Control Never Example
- Configuring Flow Control Always Example
- X.25 CUGs Examples
- DDN X.25 Configuration Example
- Blacker Front End Example
- X.25 Ping Support over Multiple Lines Example
- Booting from a Network Server over X.25 Example
- X.25 Remote Failure Detection Examples

- X.29 Access List Example
- X.29 Profile Script Example

Typical LAPB Configuration Example

In the following example, the frame size (N1), window size (k), and maximum retransmission (N2) parameters retain their default values. The **encapsulation** interface configuration command sets DCE operation to carry a single protocol, IP by default. The **lapb t1** interface configuration command sets the retransmission timer to 4,000 milliseconds (4 seconds) for a link with a long delay or slow connecting DTE device.

```
interface serial 3
 encapsulation lapb dce
 lapb t1 4000
```

Transparent Bridging for Multiprotocol LAPB Encapsulation Example

The following example configures transparent bridging for multiprotocol LAPB encapsulation:

```
no ip routing
!
interface Ethernet 1
 no ip address
 no mop enabled
 bridge-group 1
!
interface serial 0
 no ip address
 encapsulation lapb multi
 bridge-group 1
!
bridge 1 protocol ieee
```

Typical X.25 Configuration Example

The following example shows the complete configuration for a serial interface connected to a commercial X.25 PDN for routing the IP protocol. The IP subnetwork address 172.25.9.0 has been assigned for the X.25 network.

Note When you are routing IP over X.25, you must treat the X.25 network as a single IP network or subnetwork. Map entries for routers that have addresses on subnetworks other than the one on which the IP address of the interface is stored are ignored by the routing software. Additionally, all routers using the subnet number must have map entries for all other routers. Moreover, using the broadcast option with dynamic routing can result in significantly larger traffic loads, requiring a larger hold queue, larger window sizes, or multiple VCs.

```
interface serial 2
 ip address 172.25.9.1 255.255.255.0
!
 encapsulation X25
!
! The "bandwidth" command is not part of the X.25
! configuration; it is especially important to understand that it does not
```

```
! have any connection with the X.25 entity of the same name.
! "bandwidth" commands are used by IP routing processes (currently only IGRP)
! to determine which lines are the best choices for traffic.
! Since the default is 1544 Kbaud, and X.25 service at that rate is not generally
! available, most X.25 interfaces that are being used with IGRP in a
! real environment will have "bandwidth" settings.
!
! This is a 9.6 Kbaud line:
!
 bandwidth 10
! You must specify an X.121 address to be assigned to the X.25 interface by the PDN.
!
 x25 address 31370054065
!
! The following Level 3 parameters have been set to match the network.
! You generally need to change some Level 3 parameters, most often
! those listed below. You might not need to change any Level 2
! parameters, however.
!
 x25 htc 32
!
! These Level 3 parameters are default flow control values; they need to
! match the PDN defaults. The values used by an SVC are negotiable on a per-call basis:
!
 x25 win 7
 x25 wout 7
 x25 ips 512
 x25 ops 512
!
!
! The following commands configure the default behavior for our encapsulation
! SVCs
!
 x25 idle 5
 x25 nvc 2
!
! The following commands configure the X.25 map. If you want to exchange
! routing updates with any of the routers, they would need
! "broadcast" flags.
! If the X.25 network is the only path to the routers, static routes are
! generally used to save on packet charges. If there is a redundant
! path, it might be desirable to run a dynamic routing protocol.
!
 x25 map IP 172.25.9.3 31370019134 ACCEPT-REVERSE
! ACCEPT-REVERSE allows collect calls
 x25 map IP 172.25.9.2 31370053087
!
! If the PDN cannot handle fast back-to-back frames, use the
!"transmitter-delay" command to slow down the interface.
!
 transmitter-delay 1000
```

VC Ranges Example

The following example sets the VC ranges of 5 to 20 for incoming calls only (from the DCE to the DTE) and 25 to 1024 for either incoming or outgoing calls. It also specifies that no VCs are reserved for outgoing calls (from the DTE to the DCE). Up to four permanent VCs can be defined on VCs 1 through 4.

```
x25 lic 5
x25 hic 20
x25 ltc 25
```

X.25 Failover Example

In the following example, X.25 failover is configured on a network that is also configured for Annex G. If data-link connection identifier (DLCI) 13 or DLCI 14 on serial interface 1/0 goes down, dialer interface 1 will serve as the secondary interface. After DLCI 13 or 14 comes back up and remains up for 20 seconds, dialer interface 1 will reset, sending all calls back to the primary interface.

```
interface serial1/0
 encapsulation frame-relay
 frame-relay interface-dlci 13
 x25-profile frame1
  exit
 frame-relay interface-dlci 14
 x25-profile frame1
  exit
!
interface dialer1
 encapsulation x25
   exit

x25 route ^1234 interface serial1/0 dlci 13
x25 route ^1234 interface serial1/0 dlci 14
x25 route ^1234 interface dialer1
!
x25 profile frame1 dte
 x25 fail-over 20 interface dialer1
  exit
!
```

PVC Switching on the Same Router Example

In the following example, a PVC is connected between two serial interfaces on the same router. In this type of interconnection configuration, the destination interface must be specified along with the PVC number on that interface. To make a working PVC connection, two commands must be specified, each pointing to the other.

```
interface serial 0
 encapsulation x25
 x25 ltc 5
 x25 pvc 1 interface serial 1 pvc 4
!
interface serial 1
 encapsulation x25
 x25 ltc 5
 x25 pvc 4 interface serial 0 pvc 1
```

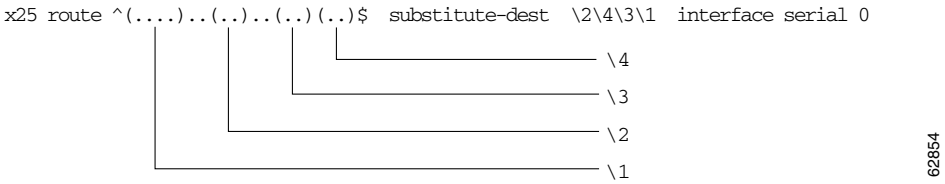
X.25 Route Address Pattern Matching Example

The following example shows how to route X.25 calls with addresses whose first four Data Network Identification Code (DNIC) digits are 1111 to interface serial 3. This example also shows how to change the DNIC field to 2222 in the addresses presented to equipment connected to that interface. The *1* in the rewrite pattern indicates the portion of the original address matched by the digits following the 1111 DNIC.

```
x25 route ^1111(.*) substitute-dest 2222\1 interface serial 3
```

Figure 9 shows a more contrived command intended to illustrate the power of the rewriting scheme.

Figure 9 X.25 Route Address Pattern Matching Example



The command in Figure 9 causes all X.25 calls with 14-digit called addresses to be routed through interface serial 0. The incoming DNIC field is moved to the end of the address. The fifth, sixth, ninth, and tenth digits are deleted, and the thirteenth and fourteenth are moved before the eleventh and twelfth.

X.25 Routing Examples

The following examples illustrate how to enable the X.25 switch service and how to configure a router on a Tymnet/PAD switch to accept and forward calls.

The first example shows enabling X.25 switching and entering routes in the X.25 routing table:

```
! Enable X.25 forwarding
x25 routing
! Enter routes into the table. Without a positional parameter, entries
! are appended to the end of the table
x25 route ^100$ interface serial 0
x25 route 100 cud ^pad$ interface serial 2
x25 route 100 interface serial 1
x25 route ^3306 interface serial 3
x25 route .* ip 10.2.0.2
```

The routing table forwards calls for X.121 address 100 out interface serial 0. Otherwise, calls are forwarded onto serial 1 if the X.121 address contains 100 anywhere within it and contains no call user data (CUD), or if the CUD is not the string “pad.” If the X.121 address contains the digits 100 and the CUD is the string “pad,” the call is forwarded onto serial 2. All X.121 addresses that do not match the first three routes are checked for a DNIC of 3306 as the first four digits. If they do match, they are forwarded over serial 3. All other X.121 addresses will match the fifth entry, which is a match-all pattern and will have a TCP connection established to the IP address 10.2.0.2. The router at 10.2.0.2 will then handle the call according to its configuration.

This second example configures a router that sits on a Tymnet/PAD switch to accept calls and have them forwarded to a DEC VAX system. This feature permits running an X.25 network over a generalized existing IP network, thereby making another physical line for one protocol unnecessary. The router positioned next to the DEC VAX system is configured with X.25 routes, as follows:

```
x25 route vax-x121-address interface serial 0
x25 route .* ip cisco-on-tymnet-ipaddress
```

These commands route all calls to the DEC VAX X.121 address out to serial 0, where the VAX is connected running PSI. All other X.121 addresses are forwarded to the “cisco-on-tymnet” address through its IP address. As a result, all outgoing calls from the VAX are sent to “cisco-on-tymnet” for further processing.

On the router named “cisco-on-tymnet”, you enter these commands:

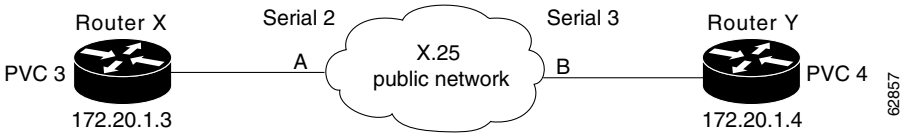
```
x25 route vax-x121-address ip cisco-on-vax
x25 route .* interface serial 0
```

These commands force all calls with the VAX X.121 address to be sent to the router that has the VAX connected to it. All other calls with X.121 addresses are forwarded out to Tymnet. If Tymnet can route them, a Call Accepted packet is returned, and everything proceeds normally. If Tymnet cannot handle the calls, it clears each call and the Clear Request packet is forwarded back toward the VAX.

PVC Used to Exchange IP Traffic Example

The following example, illustrated in Figure 10, demonstrates how to use the PVC to exchange IP traffic between router X and router Y.

Figure 10 Establishing an IP Encapsulation PVC Through an X.25 Network



Configuration for Router X

```
interface serial 2
 ip address 172.20.1.3 255.255.255.0
 x25 pvc 4 ip 172.20.1.4
```

Configuration for Router Y

```
interface serial 3
 ip address 172.20.1.4 255.255.255.0
 x25 pvc 3 ip 172.20.1.3
```

In this example, the PDN has established a PVC through its network, connecting PVC number 3 of access point A to PVC number 4 of access point B. On router X, a connection is established between router X and router Y's IP address, 172.20.1.4. On router Y, a connection is established between router Y and router X's IP address, 172.20.1.3.

Point-to-Point Subinterface Configuration Example

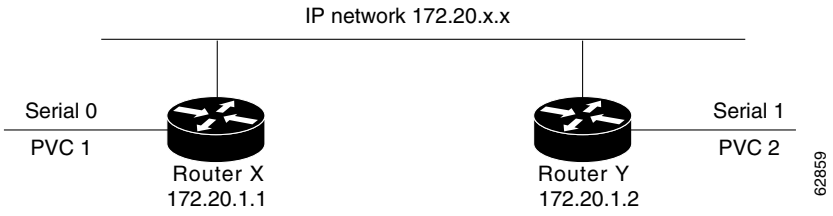
The following example creates a point-to-point subinterface, maps IP and AppleTalk to a remote host, and creates an encapsulating PVC for DECnet to the same remote host, identified by the X.121 address in the commands:

```
interface Serial0.1 point-to-point
 x25 map ip 172.20.170.90 170090 broadcast
 x25 map appletalk 4.50 170090 broadcast
 x25 pvc 1 decnet 1.2 170090 broadcast
```

Simple Switching of a PVC over XOT Example

In the following simple example, a connection is established between two PVCs across a LAN. Because the connection is remote (across the LAN), the XOT service is used. This example establishes a PVC between router X, serial 0, PVC 1 and router Y, serial 1, PVC 2. Keepalives are enabled to maintain connection notification. Figure 11 provides a visual representation of the configuration.

Figure 11 X.25 PVC Connection



Configuration for Router X

```
service tcp-keepalives-in
service tcp-keepalives-out
interface serial 0
 x25 pvc 1 xot 172.20.1.2 interface serial 1 pvc 2
```

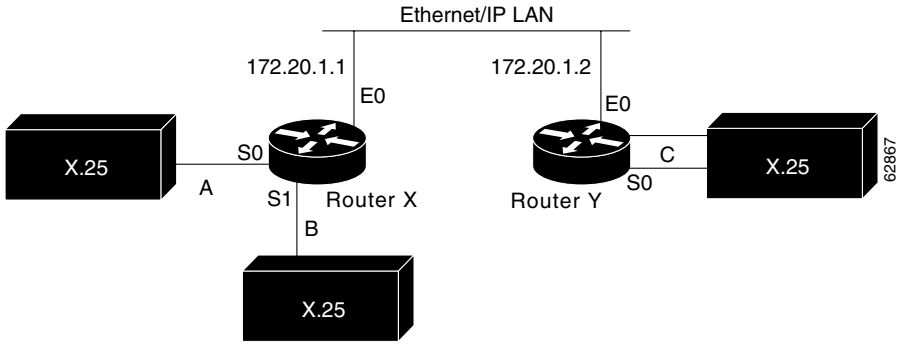
Configuration for Router Y

```
service tcp-keepalives-in
service tcp-keepalives-out
interface serial 1
 x25 pvc 2 xot 172.20.1.1 interface serial 0 pvc 1
```

PVC Switching over XOT Example

In the more complex example shown in Figure 12, the connection between points A and B is switched, and the connections between point C and points A and B are made using XOT. Keepalives are enabled to maintain connection notification.

Figure 12 PVC Switching over XOT



Configuration for Router X

```
service tcp-keepalives-in
service tcp-keepalives-out
interface ethernet 0
 ip address 172.20.1.1 255.255.255.0
!
interface serial 0
 x25 ltc 5
 x25 pvc 1 interface serial 1 pvc 1
 x25 pvc 2 xot 172.20.1.2 interface serial 0 pvc 1
```

```
!
interface serial 1
 x25 ltc 5
 x25 pvc 1 interface serial 0 pvc 1
 x25 pvc 2 xot 172.20.1.2 interface serial 0 pvc 2
```

Configuration for Router Y

```
service tcp-keepalives-in
service tcp-keepalives-out
interface ethernet 0
 ip address 172.20.1.2 255.255.255.0
!
interface serial 0
 x25 ltc 5
 x25 pvc 1 xot 172.20.1.1 interface serial 0 pvc 2
 x25 pvc 2 xot 172.20.1.1 interface serial 1 pvc 2
```

X.25 Load Balancing Examples

For examples of X.25 load balancing, see the following sections:

- X.25 Load Balancing Using VC-Count Distribution Method Example
- X.25 Load Balancing with Multiple Hunt Groups Example

X.25 Load Balancing Using VC-Count Distribution Method Example

In the following example, the vc-count distribution method is used on two serial interfaces that have different numbers of VCs. Assuming that no sessions are being terminated at this time, the first 450 calls will be sent to Serial1, and subsequent calls will alternate between Serial0 and Serial1 until the interfaces are full.

```
!
interface serial0
  description 56k link supporting 50 virtual circuits
  x25 htc 50
!
interface serial1
  description T1 line supporting 500 virtual circuits
  x25 htc 500
!
x25 hunt-group hg-vc vc-count
  interface serial0
  interface serial1
!
```

X.25 Load Balancing with Multiple Hunt Groups Example

The following example enables X.25 encapsulation on relevant serial interfaces and configures serial interfaces 1 and 2 to participate in X.25 hunt group “HG1,” and serial interfaces 0 and 3 to participate in X.25 hunt group “HG2.” Serial interfaces 1 and 2 and XOT IP addresses 172.17.125.54 and 172.17.125.34 are then associated with hunt group “HG1” (with rotary distribution assigned); and serial interfaces 0 and 3 are associated with hunt group “HG2” (with vc-count distribution assigned). These hunt groups are then added to the routing table, where X.25 route 1111 will use “HG1” and X.25 route 1112 will use “HG2”.

```
x25 routing
```

```
interface serial 0
 encapsulation x25
 interface serial 1
 encapsulation x25
 interface serial 2
 encapsulation x25
 interface serial 3
 encapsulation x25
!
x25 hunt-group HG1 rotary
 interface serial 1
 interface serial 2
 xot 172.17.125.54
 xot 172.17.125.34
 exit
!
x25 hunt-group HG2 vc-count
 interface serial0
 interface serial3
 exit
!
x25 route 1111 hunt-group HG1
x25 route 1112 hunt-group HG2
```

X.25 Switching Between PVCs and SVCs Example

The following example allows X.25 switching between a PVC on the first interface and an SVC on the second interface. X.25 traffic arriving on PVC 20 on serial interface 0 will cause a call to be placed to 000000160100, if one does not already exist.

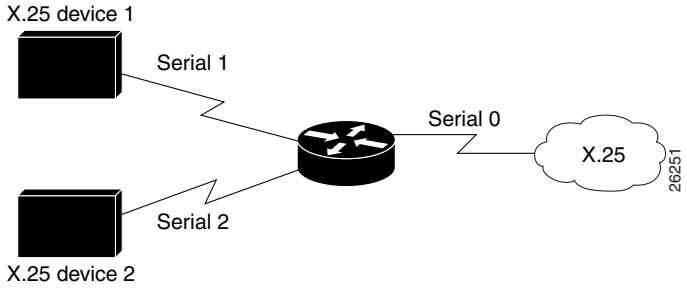
```
x25 routing
interface serial0
 encapsulation x25
 x25 address 000000180100
 x25 ltc 128
 x25 pvc 20 svc 000000160100 packetsize 128 128 windowsize 2 2

interface serial2
 encapsulation x25 dce
 x25 route ^000000160100$ interface Serial2
 x25 route ^000000180100$ interface Serial0
```

The **x25 route** command adds the two X.121 addresses to the X.25 routing table. Data traffic received on PVC 20 on serial interface 0 will cause a call to be placed with a Called (destination) Address of 000000160100; this call will be routed to serial interface 2. Alternatively, an X.25 call received with a Called Address of 000000180100 and a Calling Address of 000000160100 will be associated with PVC 20 on serial interface 0. In either case, subsequent X.25 traffic on either the SVC or the PVC will be forwarded to the other circuit. Because no idle timeout has been specified for the interface or for the circuit, the router will not clear the call.

Inserting and Removing X.121 Addresses As Calls Are Routed Example

The following example shows insertions and removals in the X.121address as calls from the X.25 network get routed to X.25 devices. Figure 13 shows the topology for this example.

Figure 13 Typical X.25 Network Configuration



Example Configuration

```
x25 route ^2(.*) input-interface serial1 substitute-dest \1 interface serial2
x25 route input-interface serial2 source .* substitute-source 2\0 interface serial0
```

For a call coming from interface serial 1 with a called address starting with 2, the 2 is stripped off the called address and the call forwarded to serial interface 2.

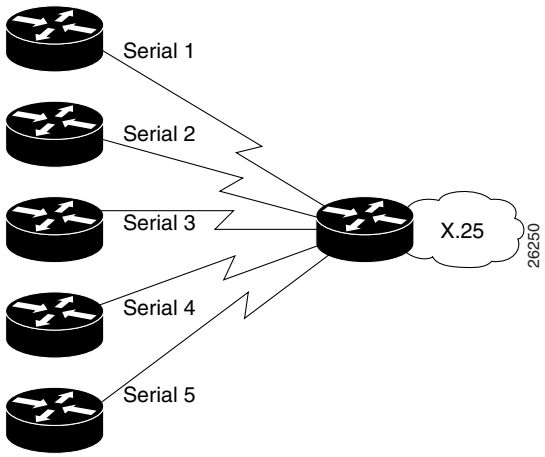
For a call coming from interface serial 2 with any calling address, a 2 will be inserted to its calling address and the call forwarded to serial interface 0.

Forwarding Calls Using the continue Keyword Example

This section provides two examples of the same configuration. Both examples show how to forward calls among a number of local X.25 devices; however, the second example shows how the **continue** keyword reduces the number of routing statements. (Keep in mind that the **continue** keyword is most useful when you will be switching calls among four or more routes.)

Figure 14 illustrates the network topology for both examples.

Figure 14 X.25 Network with Multiple Interfaces



X.25 Routing Statements Before continue Keyword

The following example shows how to forward calls among a number of local X.25 devices without using the **continue** keyword:

```
x25 route ^02 input-interface serial 1 substitute-source 01\0 substitute-dest \1 interface serial 2
x25 route ^03 input-interface serial 1 substitute-source 01\0 substitute-dest \1 interface serial 3
x25 route ^04 input-interface serial 1 substitute-source 01\0 substitute-dest \1 interface serial 4
x25 route ^05 input-interface serial 1 substitute-source 01\0 substitute-dest \1 interface serial 5
!
x25 route ^01 input-interface serial 2 substitute-source 02\0 substitute-dest \1 interface serial 1
x25 route ^03 input-interface serial 2 substitute-source 02\0 substitute-dest \1 interface serial 3
x25 route ^04 input-interface serial 2 substitute-source 02\0 substitute-dest \1 interface serial 4
x25 route ^05 input-interface serial 2 substitute-source 02\0 substitute-dest \1 interface serial 5
!
x25 route ^02 input-interface serial 3 substitute-source 03\0 substitute-dest \1 interface serial 2
x25 route ^01 input-interface serial 3 substitute-source 03\0 substitute-dest \1 interface serial 1
x25 route ^04 input-interface serial 3 substitute-source 03\0 substitute-dest \1 interface serial 4
x25 route ^05 input-interface serial 3 substitute-source 03\0 substitute-dest \1 interface serial 5
!
x25 route ^02 input-interface serial 4 substitute-source 04\0 substitute-dest \1 interface serial 2
x25 route ^03 input-interface serial 4 substitute-source 04\0 substitute-dest \1 interface serial 3
x25 route ^01 input-interface serial 4 substitute-source 04\0 substitute-dest \1 interface serial 1
x25 route ^05 input-interface serial 4 substitute-source 04\0 substitute-dest \1 interface serial 5
!
x25 route ^02 input-interface serial 5 substitute-source 05\0 substitute-dest \1 interface serial 2
x25 route ^03 input-interface serial 5 substitute-source 05\0 substitute-dest \1 interface serial 3
x25 route ^04 input-interface serial 5 substitute-source 05\0 substitute-dest \1 interface serial 4
x25 route ^01 input-interface serial 5 substitute-source 05\0 substitute-dest \1 interface serial 1
```

Same X.25 Network Configuration with continue Keyword

The following example shows how to forward calls among a number of local X.25 devices using the **continue** keyword:

```
x25 route input-interface serial 1 source .* substitute-source 01\0 continue
x25 route input-interface serial 2 source .* substitute-source 02\0 continue
x25 route input-interface serial 3 source .* substitute-source 03\0 continue
x25 route input-interface serial 4 source .* substitute-source 04\0 continue
x25 route input-interface serial 5 source .* substitute-source 05\0 continue
x25 route ^01(.*) substitute-dest \1 interface serial 1
x25 route ^02(.*) substitute-dest \1 interface serial 2
x25 route ^03(.*) substitute-dest \1 interface serial 3
x25 route ^04(.*) substitute-dest \1 interface serial 4
x25 route ^05(.*) substitute-dest \1 interface serial 5
```

DNS-Based X.25 Routing Example

The following example shows XOT switch configuration for XOT switching via the DNS:

```
Router(config)# ip tcp synwait-time 5
Router(config)# ip name-server 10.1.1.40
Router(config)# x25 routing
Router(config)# service pad to-xot
Router(config)# service pad from-xot
Router(config)# ip domain-name home.com
Router(config)# ip domain-list home.com
Router(config)# ip domain-lookup
Router(config)# interface Ethernet1
Router(config-if)# ip address 10.1.1.2 255.255.255.0
```

```
Router(config-if)# exit
Router(config)# interface Serial0
Router(config-if)# encapsulation x25 dce
Router(config-if)# exit
Router(config)# x25 route 444 xot dns \0
Router(config)# x25 route 555 xot dns \0
```

X.25 over Frame Relay (Annex G) Example

The following example configures X.25 profile "NetworkNodeA" (using the X.25 commands **x25 htc**, **x25 idle**, **x25 accept-reverse** and **x25 modulo**) on DLCI interfaces 20 and 30; and X.25 profile "NetworkNodeB" (using the X.25 command **x25 address**) on DLCI interface 40; all on serial interface 1. The example shows the final step of assigning your X.25 profile to the DLCI interface by using the **frame-relay interface-dlci** command, and then assigning X.25 routes to DLCIs 20, 30, and 40 using the **x25 route** command.

The new **x25 profile** command mode (config-x25) can be seen in this example. This mode is used for configuring the parameters of your X.25 profile. For a complete description of this command and mode, refer to the **x25 profile** command section in the chapter "X.25 and LAPB Commands" in the *Cisco IOS Wide-Area Networking Command Reference*.

This example assumes that you already have Frame Relay enabled on your router.

```
Router(config)# x25 routing
Router(config)# x25 profile NetworkNodeA dce
Router(config-x25)# x25 htc 128
Router(config-x25)# x25 idle 5
Router(config-x25)# x25 accept-reverse
Router(config-x25)# x25 modulo 128
Router(config-x25)# end
Router(config)# x25 profile NetworkNodeB dce
Router(config-x25)# x25 address 1111
Router(config-x25)# end
Router(config)# interface serial1
Router(config-if)# encapsulation frame-relay
Router(config-if)# frame-relay interface-dlci 20
Router(config-fr-dlci)# x25-profile NetworkNodeA
Router(config-fr-dlci)# end
Router(config)# interface serial1
Router(config-if)# frame-relay interface-dlci 30
Router(config-fr-dlci)# x25-profile NetworkNodeA
Router(config-fr-dlci)# end
Router(config)# interface serial1
Router(config-if)# frame-relay interface-dlci 40
Router(config-fr-dlci)# x25-profile NetworkNodeB
Router(config-fr-dlci)# end
Router(config)# x25 route 2000 interface serial1 dlci 20
Router(config)# x25 route 3000 interface serial1 dlci 30
Router(config)# x25 route 4000 interface serial1 dlci 40
```

CMNS Switching Example

The following example illustrates enabling CMNS and configuring X.25 routes to the available CMNS host and the PDN connectivity:

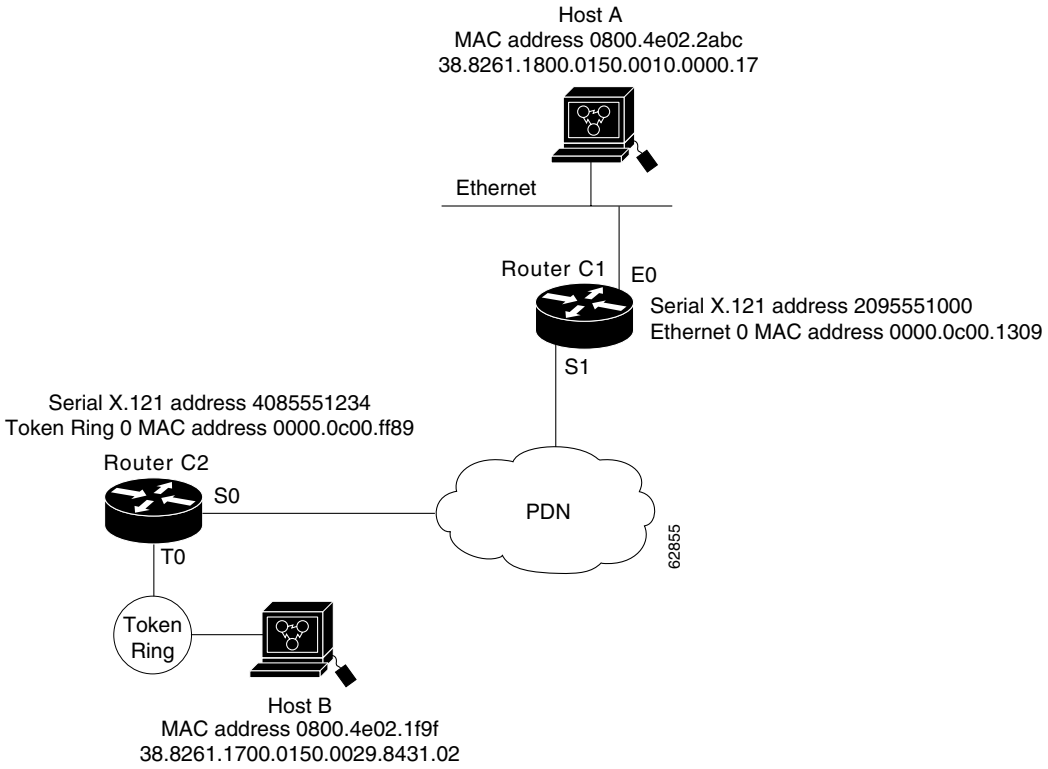
```
interface ethernet 0
 cmns enable
!
interface serial 0
```

```
 encapsulation x25
!
interface serial 1
 encapsulation x25
!
x25 route dest-ext ^38.8261.1000.0150.1000.17 interface Ethernet0 mac 0000.0c00.ff89
! Above maps NSAP to MAC-address on Ethernet0
!
x25 route dest-ext ^38.8261.1000.0150.1000.18 substitute-dest 3110451 interface Serial0
! Above maps NSAP to X.121-address on Serial0 assuming the link is over a PDN
!
x25 route dest-ext ^38.8261.1000.0150.1000.20 interface Serial1
! Above specifies cmns support for Serial1
! assuming that the link is over a leased line
```

CMNS Switching over a PDN Example

The following example depicts switching CMNS over a packet-switched PDN. Figure 15 illustrates the general network topology for a CMNS switching application where calls are being made between resources on opposite sides of a remote link to Host A (on an Ethernet) and Host B (on a Token Ring), with a PDN providing the connection.

Figure 15 Example Network Topology for Switching CMNS over a PDN



The following configuration listing allows resources on either side of the PDN to call host A or host B. This configuration allows traffic intended for the remote NSAP address specified in the **x25 route** commands (for the serial ports) to be switched through the serial interface for which CMNS is configured.

Configuration for Router C2

```
interface token 0
 cmns enable
!
interface serial 0
 encapsulation x25
 x25 address 4085551234
!
x25 route dest-ext ^38.8261.17 interface Token0 mac 0800.4e02.1f9f
!
! The line above specifies that any traffic from any other interface
! intended for any NSAP address with NSAP prefix 38.8261.17 will be
! switched to MAC address 0800.4e02.1f9f through Token Ring 0
!
x25 route dest-ext ^38.8261.18 substitute-dest 2095551000 interface Serial0
!
! The line above specifies that traffic from any other interface
! on Cisco Router C2 that is intended for any NSAP address with
! NSAP-prefix 38.8261.18 will be switched to
! X.121 address 2095551000 through Serial 0
```

Configuration for Router C1

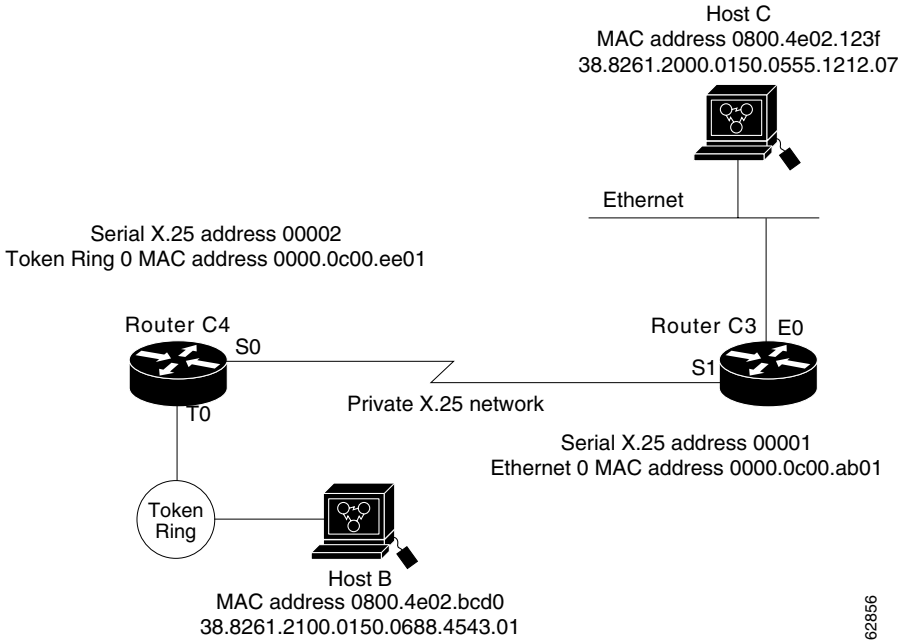
```
interface ethernet 0
 cmns enable
!
interface serial 1
 encapsulation x25
 x25 address 2095551000
!
x25 route dest-ext ^38.8261.18 interface Ethernet0 mac 0800.4e02.2abc
!
! The line above specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.18
! will be switched to MAC address 0800.4e02.2abc through Ethernet 0
!
x25 route dest-ext ^38.8261.17 substitute-dest 4085551234 interface Serial1
!
! The line above specifies that traffic from any other interface
! on Cisco Router C1 that is intended for any NSAP address with
! NSAP-prefix 38.8261.17 will be switched to X.121 address
! 4085551234 through Serial 1
```

CMNS Switched over Leased Lines Example

The following example illustrates switching CMNS over a leased line. Figure 16 illustrates the general network topology for a CMNS switching application where calls are being made by resources on the opposite sides of a remote link to host C (on an Ethernet) and host B (on a Token Ring), with a dedicated leased line providing the connection.

The following configuration listing allows resources on either side of the leased line to call host C or host B. This configuration allows traffic intended for the remote NSAP address specified in the **x25 route** commands (for the serial ports) to be switched through the serial interface for which CMNS is configured.

Figure 16 Example Network Topology for Switching CMNS over a Leased Line



A key difference for this configuration compared with the previous example is that with no PDN, the substitution of the destination X.121 address in the **x25 route** command is not necessary. The specification of an X.25 address also is not needed, but it is included for symmetry with the previous example.

Configuration for Router C4

```
interface token 0
 cmns enable
!
interface serial 0
 encapsulation x25
 x25 address 4085551234
!
x25 route dest-ext ^38.8261.17 interface Token0 mac 0800.4e02.1f9f
!
! The line above specifies that any traffic from any other interface
! intended for any NSAP address with NSAP prefix 38.8261.17 will be
! switched to MAC address 0800.4e02.1f9f through Token Ring 0
!
x25 route dest-ext ^38.8261.18 interface Serial0
!
! The line above specifies that traffic from any other interface
! on Cisco Router C2 that is intended for any NSAP address with
! NSAP-prefix 38.8261.18 will be switched to
! X.121 address 2095551000 through Serial 0
```

Configuration for Router C3

```
interface ethernet 0
 cmns enable
!
interface serial 1
 encapsulation x25
 x25 address 2095551000
```

```
!
x25 route dest-ext ^38.8261.18 interface Ethernet0 mac 0800.4e02.2abc
!
! The line above specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.18
! will be switched to MAC address 0800.4e02.2abc through Ethernet 0
!
x25 route dest-ext ^38.8261.17 interface Serial1
!
! The line above specifies that traffic from any other interface
! on Cisco Router C1 that is intended for any NSAP address with
! NSAP-prefix 38.8261.17 will be switched to X.121 address
! 4085551234 through Serial 1
```

Configuring Local Acknowledgment Example

The following example shows X.25 local acknowledgment being configured on the router:

```
Router(config)# x25 routing acknowledge local
```

Setting Asymmetrical Window and Packet Sizes Flow Control Never Example

The following example shows asymmetrical window and packet sizes being set on the router on serial interfaces 0 and 1, with local acknowledgment enabled globally, and flow control disabled on both interfaces to allow asymmetrical flow control to occur:

```
Router(config)# interface serial0
Router(config-if)# x25 win 2
Router(config-if)# x25 wout 3
Router(config-if)# x25 ips 256
Router(config-if)# x25 ops 512
Router(config-if)# x25 ops 512
Router(config-if)# exit
Router(config)# interface serial1
Router(config-if)# x25 win 4
Router(config-if)# x25 wout 5
Router(config-if)# x25 ips 128
Router(config-if)# x25 ops 512
Router(config-if)# exit
Router(config)# x25 routing acknowledge local
Router(config)# interface serial 0
Router(config-if)# encapsulation x25 dte
Router(config-if)# x25 subscribe flow-control never
Router(config-if)# exit
Router(config)# interface serial 1
Router(config-if)# encapsulation x25 dte
Router(config-if)# x25 subscribe flow-control never
```

Configuring Flow Control Always Example

The following example shows X.25 routing with local acknowledgment being enabled globally and flow control negotiation being enabled on serial interface 1/4. Window size ranges are set at a permitted rate of 1 (minimum) and 7 (maximum) and target rate of 2 (minimum) and 4 (maximum).

Packet size ranges are set at a permitted rate of 64 (minimum) and 1024 (maximum), and target rate of 128 (minimum) and 1024 (maximum).

```
Router(config)# x25 routing acknowledge local
```

```
Router(config)# interface serial 1/4
Router(config-if)# encapsulation x25 dte
Router(config-if)# x25 subscribe flow-control always
Router(config-if)# x25 subscribe windowsize permit 1 7 target 2 4
Router(config-if)# x25 subscribe packetsize permit 64 1024 target 128 1024
```

You do not have to configure window and packet size ranges because their default settings are appropriate for most configurations. The following example shows X.25 routing with local acknowledgment being enabled globally and flow control negotiation being enabled on serial interface 1/4 with default window and packet size settings:

```
Router(config)# interface serial 1/4
Router(config-if)# encapsulation x25 dte
Router(config-if)# x25 subscribe flow-control always
```

X.25 CUGs Examples

The following sections provide examples of different X.25 closed user groups (CUGs) configurations:

- X.25 CUG Service, Access, and CUG Properties Example
- POP with No CUG Access Example
- POP with Access Restricted to One CUG Example
- POP with Multiple CUGs and No Public Access Example
- POP with Multiple CUGs and Public Access Example
- CUG Selection Facility Suppression for the Preferential CUG Example
- CUG Selection Facility Suppression for All CUGs Example

X.25 CUG Service, Access, and CUG Properties Example

In the following example, X.25 CUG service is being subscribed to on serial 0, which then permits the subscription to local CUGs (5000, 100, 200, and 300). Subscription to local CUGs cannot be achieved without subscription to X.25 CUG service (although this occurs automatically—with CUG service default settings of no incoming and no outgoing access—the first time you subscribe to a specific CUG using the **x25 subscribe local-cug** command).

Local CUG 5000 has been designated as the preferential CUG, which means that it will be used when a call with no CUG membership selection is made. These local CUGs all belong to different network identifiers (IDs) (local 5000 = network 55; local 100 = network 11; local 200 = network 22; local 300 = network 33), but they could also subscribe to the same network ID if desired.

```
Router(config)# interface serial0
Router(config-if)# encapsulation x25 dce
Router(config-if)# x25 subscribe cug-service incoming-access outgoing-access
Router(config-if)# x25 subscribe local-cug 5000 network-cug 55 preferential
Router(config-if)# x25 subscribe local-cug 100 network-cug 11
Router(config-if)# x25 subscribe local-cug 200 network-cug 22
Router(config-if)# x25 subscribe local-cug 300 network-cug 33
```

POP with No CUG Access Example

In the following example, serial interface 0 is being configured as a POP for a user that has no access to any of the CUGs in the network, but full public access (incoming and outgoing access)—the least restrictive setting:

```
Router(config)# interface serial0
Router(config-if)# encapsulation x25 dce
Router(config-if)# x25 subscribe cug-service incoming-access outgoing-access
```

POP with Access Restricted to One CUG Example

In the following example, serial interface 0 is configured as a POP with access only to members of its own CUG and no public access. The POP is being configured for CUG service security using the most restrictive settings (the default) of the **x25 subscribe cug-service** command—no incoming and no outgoing access permitted. Local CUG 5000, which is associated with network 55, is being subscribed to this POP.

An outgoing call from the DTE may select local CUG 5000 or not. Because there is only one CUG subscribed to, its use is implicit. CUG 5000 will always select its related network CUG 55. An outgoing call that specifies a different local CUG will be refused. An incoming call must specify network CUG 55; otherwise the call will be refused.

```
Router(config)# interface serial0
Router(config-if)# encapsulation x25 dce
Router(config-if)# x25 subscribe cug-service
Router(config-if)# x25 subscribe local-cug 5000 network-cug 55
```

POP with Multiple CUGs and No Public Access Example

In the following example, serial interface 0 is being configured as a POP with access to members of several CUGs, using the most restrictive settings (the default) of the **x25 subscribe cug-service** command—no incoming and no outgoing access permitted. Local CUGs (5000, 100, 200, and 300) are then subscribed to this POP. Local CUG 5000 has been designated as the preferential CUG, which means that it will be used when a call with no CUG membership selection was made.

These local CUGs all belong to different networks (local 5000 = network 55; local 100 = network 11; local 200 = network 22; local 300 = network 33), but they could also subscribe to the same network if desired.

An outgoing call from the DTE may select any of the local CUGs (5000, 100, 200, and 300) or not. Because there is a preferential CUG (5000), its use will be implicit when no CUG is specified. The related network CUG (55) will be selected when switched to an intranetwork connection. A call specifying a different local CUG will be refused. An incoming call must select one of the network CUGs (55, 11, 22, or 33); otherwise the call will be refused.

```
Router(config)# interface serial0
Router(config-if)# encapsulation x25 dce
Router(config-if)# x25 subscribe cug-service
Router(config-if)# x25 subscribe local-cug 5000 network-cug 55 preferential
Router(config-if)# x25 subscribe local-cug 100 network-cug 11
Router(config-if)# x25 subscribe local-cug 200 network-cug 22
Router(config-if)# x25 subscribe local-cug 300 network-cug 33
```

POP with Multiple CUGs and Public Access Example

In the following example, serial interface 0 is being configured as a POP with public access to members of several CUGs and the means to originate and receive calls from the open network (that is, to or from users that do not subscribe to one of the CUGs to which this POP subscribes).

An outgoing call from the DTE may select any of the local CUGs (1, 2, 3, or 4) or not. When no CUG is selected, it is assumed that the call is intended for the open network. When a CUG is selected, the related network CUG will be selected when the call is switched to an intranetwork connection. The call will be refused if it specifies a different local CUG from the one to which the POP is subscribed.

An incoming call to the DTE from an intra network connection may select related network CUGs (101, 202, 303, or 404) or no CUG. If no CUG is selected, the call is accepted as coming from the open network. A call that requires access to a different CUG will be refused.

```
Router(config)# interface serial0
Router(config-if)# encapsulation x25 dce
Router(config-if)# x25 subscribe cug-service incoming-access outgoing-access
Router(config-if)# x25 subscribe local-cug 1 network-cug 101
Router(config-if)# x25 subscribe local-cug 2 network-cug 202
Router(config-if)# x25 subscribe local-cug 3 network-cug 303
Router(config-if)# x25 subscribe local-cug 4 network-cug 404
```

CUG Selection Facility Suppression for the Preferential CUG Example

In the following example, CUG selection facility suppression is configured for the preferential CUG only on serial interface 0:

```
interface serial0
  encapsulation x25 dce
  x25 subscribe cug-service suppress preferential
  x25 subscribe local-cug 0 network-cug 10 preferential
  x25 subscribe local-cug 50 network-cug 500
```

CUG Selection Facility Suppression for All CUGs Example

In the following example, CUG selection facility suppression and incoming access are configured for all CUGs, including the preferential CUG on the X.25 profile:

```
x25 profile CUG-SUPRS-ALL dce
  x25 subscribe cug-service incoming-access suppress all
  x25 subscribe local-cug 0 network-cug 10 preferential
  x25 subscribe local-cug 20 network-cug 202
  x25 subscribe local-cug 40 network-cug 40
```

DDN X.25 Configuration Example

The following example illustrates how to configure a router interface to run DDN X.25:

```
interface serial 0
 ip address 192.31.7.50 255.255.255.240
 encapsulation x25 ddn
 x25 win 6
 x25 wout 6
 x25 ips 1024
 x25 ops 1024
 x25 t20 10
 x25 t21 10
 x25 t22 10
 x25 t23 10
 x25 nvc 2
 x25 map IP 192.31.7.49 000000010300 BROADCAST
```

Blacker Front End Example

In the following example, interface serial 0 is configured to attach to the DDN X.25 network via a Blacker Front End.

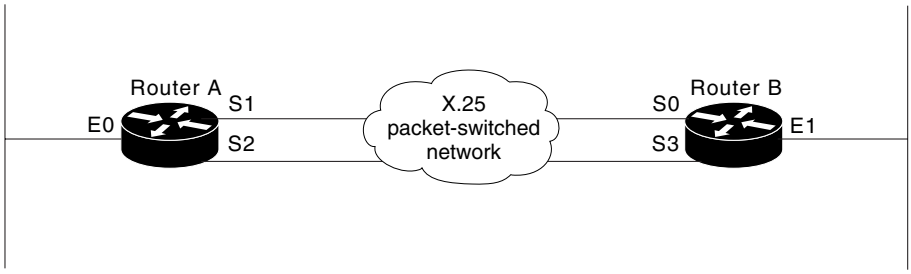
```
interface serial 0
 ip address 21.0.0.2 255.0.0.0
 encapsulation x25 bfe
```

X.25 Ping Support over Multiple Lines Example

For **ping** commands to work in an X.25 environment (when load sharing is occurring over multiple serial lines), you must include entries for all adjacent interface IP addresses in the **x25 map** command for each serial interface. The following example illustrates this point.

Consider two routers, router A and router B, communicating with each other over two serial lines via an X.25 PDN (see Figure 17) or over leased lines. In either case, all serial lines must be configured for the same IP subnet address space. The configuration that follows allows for successful **ping** commands. A similar configuration is required for the same subnet IP addresses to work across X.25.

Figure 17 Parallel Serial Lines to an X.25 Network



Note All four serial ports configured for the two routers in the following configuration example must be assigned to the same IP subnet address space. In this case, the subnet is 172.20.170.0.

Configuration for Router A

```
interface serial 1
 ip 172.20.170.1 255.255.255.0
 x25 address 31370054068
 x25 alias ^31370054069$
 x25 map ip 172.20.170.3 31370054065
 x25 map ip 172.20.170.4 31370054065
!
interface serial 2
 ip 172.20.170.2 255.255.255.0
 x25 address 31370054069
 x25 alias ^31370054068$
 x25 map ip 172.20.170.4 31370054067
 x25 map ip 171.20.170.3 31370054067
! allow either destination address
```

Configuration for Router B

```
interface serial 0
```

```
 ip 172.20.170.3 255.255.255.0
 x25 address 31370054065
 x25 alias ^31370054067$
 x25 map ip 172.20.170.1 31370054068
 x25 map ip 172.20.170.2 31370054068
!
interface serial 3
 ip 172.20.170.4 255.255.255.0
 x25 address 31370054067
 x25 alias ^31370054065$
 x25 map ip 172.20.170.2 31370054069
 x25 map ip 172.20.170.1 31370054069
! allow either destination address
```

Booting from a Network Server over X.25 Example

You cannot boot a router over an X.25 network using broadcasts. Instead, you must boot from a specific host. Also, an **x25 map** command must exist for the host that you boot from. The **x25 map** command maps an IP address to an X.121 address. The **x25 map** command must match the IP address given on the **boot system** command line. The following is an example of such a configuration:

```
boot system gs3-k.100 172.18.126.111
interface Serial 1
 ip address 172.18.126.200 255.255.255.0
 encapsulation X25
 x25 address 10004
 x25 map IP 172.18.126.111 10002 broadcast
 lapb n1 12040
 clockrate 56000
```

In this case, 10002 is the X.121 address of the remote router that can get to host 172.18.126.111. The remote router must have the following **x25 map** entry for the remote router to return a boot image from the host to the router booting over X.25.

```
x25 map IP 172.18.126.200 10004 broadcast
```

X.25 Remote Failure Detection Examples

You must have X.25 encapsulation activated for X.25 remote failure detection to function. See the section "Configuring X.25 Encapsulation" for further details. You must also have IP static routes or a backup link configured for X.25 encapsulation.

These examples show the **x25 retry** command being used only with a secondary route. However, the **x25 retry** command can be configured for as many subinterfaces that require an alternative route. Use either one of the following examples to configure X.25 remote failure detection:

- X.25 Remote Failure Detection with IP Static Routes Example
- X.25 Remote Failure Detection and the Backup Interface Example

X.25 Remote Failure Detection with IP Static Routes Example

The following is an example of X.25 remote failure detection being configured on subinterfaces 1.1 and 1.2 using the **x25 retry** command. Subinterface 1.1 has been set at a retry every 60 seconds up to a maximum of 10 attempts.

Observe the weighting of 100 on subinterface 1.1 over 200 on subinterface 1.2 in the **ip route** command, because subinterface 1.1 is the primary route and 1.2 is the secondary route. The latter becomes activated only when subinterface 1.1 is unable to function. Weights make for predictable routing events and therefore promote the concept of primary and secondary routes.

```
Router(config)# interface serial1
Router(config-if)# encapsulation x25
Router(config-if)# x25 address 11111
Router(config-if)# exit
Router(config)# interface serial1.1 point-to-point
Router(config-subif)# ip address 172.30.22.1 255.255.255.0
Router(config-subif)# x25 map ip 172.30.22.2 22222
Router(config-subif)# x25 retry interval 60 attempts 10
Router(config-subif)# exit
Router(config)# interface serial1.2 point-to-point
Router(config-subif)# ip address 172.30.22.1 255.255.255.0
Router(config-subif)# x25 map ip 172.30.22.4 44444
Router(config-subif)# exit
Router(config)# ip route 172.30.11.1 255.255.255.0 serial1.1 100
Router(config)# ip route 172.30.11.1 255.255.255.0 serial1.2 200
```

X.25 Remote Failure Detection and the Backup Interface Example

The following configuration example is an alternative to the method previously described. X.25 remote failure detection is configured on subinterface 1.1, and interface 2 is made the backup interface. The **x25 retry** command has been set with an interval of 50 seconds up to a maximum of 20 attempts. In this example, there is no need to configure any IP static routes (as is done with the above configuration) because the backup interface is functioning as the secondary route. In other situations, there may be a need for static IP routes, depending on how the backup interface is configured.

For more details about backup, see the **backup interface** command in the chapter in the *Cisco IOS Dial Technologies Command Reference*.

```
Router(config)# interface serial1
Router(config-if)# encapsulation x25
Router(config-if)# x25 address 11111
Router(config-if)# exit
Router(config)# interface serial1.1 point-to-point
Router(config-subif)# ip address 172.30.22.1 255.255.255.0
Router(config-subif)# x25 map ip 172.30.22.2 22222
Router(config-subif)# x25 retry interval 50 attempts 20
Router(config-subif)# backup interface serial2
Router(config-subif)# exit
Router(config)# interface serial2
Router(config-if)# encapsulation x25
Router(config-if)# x25 address 11111
Router(config-if)# ip address 172.30.22.1 255.255.255.0
Router(config-if)# x25 map ip 172.30.22.3 33333
Router(config-if)# exit
```

X.29 Access List Example

The following example illustrates an X.29 access list. Incoming permit conditions are set for all IP hosts and LAT nodes that have specific characters in their names. All X.25 connections to a printer are denied. Outgoing connections are list restricted.

```
!Permit all IP hosts and LAT nodes beginning with "VMS".
!Deny X.25 connections to the printer on line 5.
!
```

```
access-list 1 permit 0.0.0.0 255.255.255.255
 lat access-list 1 permit ^VMS.*
 x29 access-list 1 deny .*
!
line vty 5
 access-class 1 in
!
!Permit outgoing connections for other lines.
!
!Permit IP access with the network 172.30
 access-list 2 permit 172.30.0.0 0.0.255.255
!
!Permit LAT access to the boojum/snark complexes.
 lat access-list 2 permit ^boojum$
 lat access-list 2 permit ^snark$
!
!Permit X.25 connections to Infonet hosts only.
 x29 access-list 2 permit ^31370
!

line vty 0 16
 access-class 2 out
```

X.29 Profile Script Example

The following profile script turns local edit mode on when the connection is made and establishes local echo and line termination upon receipt of a Return. The name *linemode* is used with the **translate** command to effect use of this script.

```
x29 profile linemode 2:1 3:2 15:1
translate tcp 172.30.1.26 x25 55551234 profile linemode
```

The X.3 PAD parameters set in the profile file and the **translate** command are described in the chapter "Configuring Protocol Translation and Virtual Asynchronous Devices" in the *Cisco IOS Terminal Services Configuration Guide*.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.

Appendixes

X.25 Facility Handling

This appendix provides reference material describing how X.25 facilities are handled by the Cisco IOS software.

X.25 Facility Handling in Datagram Transport Virtual Circuits

A router either originates or accepts datagram transport (encapsulation) switched virtual circuits (SVCs) to transport LAN traffic through an X.25 network.

When the router originates a call for LAN traffic encapsulation, the facilities in the call are controlled by the facilities configured for the interface and the map statement that specifies the LAN and X.25 encapsulation. Because a router can be attached to a public data network (PDN), the interface and map configurations allow a number of facilities to be specified in outgoing calls. These facilities are specified in all originated calls relating to the given interface and map, with one exception: the incoming and outgoing maximum packet sizes proposed are lowered if the LAPB cannot support the specified data packet size.

When the router accepts an encapsulation call, many facilities are simply ignored. The maximum packet sizes are lowered if the LAPB cannot support the sizes proposed. A reverse-charge call is cleared if neither the interface nor the map allows it. A call that specifies a network user identification (NUID) is cleared if the user authentication fails.

If an interface is configured as a DCE that is subscribed to closed user group (CUG) services, datagram encapsulation calls that originate and terminate on the interface will be subject to the requirements of CUG security.

X.25 Facility Handling in Switching Virtual Circuits

As a general rule, the X.25 switch services will forward facilities encoded in Call, Call Confirm, Clear, and Clear Confirm packets. This handling, however, is subject to the following restrictions:

- The facilities must be valid for the X.25-class service on which they were received and must be consistent with the other information presented in the packet and any prior signaling for the SVC.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

© 2007 Cisco Systems, Inc. All rights reserved.

- The facilities must be valid for the X.25-class service to which they are forwarded and must be consistent with the other information being encoded in the packet and any prior signaling for the SVC. Some limited amount of modification of facility values may be performed to meet various standard requirements; for example, some facility values have restrictions based on the station identity of the DTE/DCE sending the packet.
- Some facilities are subject to modification, insertion, or deletion by specific features configured for the incoming or outgoing X.25-class service, or by the X.25 switch service itself.

X.25 Standard Facilities

Table 1 describes how X.25 standard facilities are treated when a switched virtual circuit (SVC) is routed. If the facility was introduced in a recommendation later than the 1980 X.25 recommendation, the recommendation in which the facility was introduced is listed in parentheses after the facility name. By default, Cisco IOS software supports the 1984 recommendation.

Table 1 Treatment of Standard X.25 Facilities by Cisco IOS Software

Facility	Treatment When Switched by Cisco IOS Software
Flow Control Negotiation • Packet size • Window size • Extended window size (1996) **Note** The 1980 recommendation defines maximum Data packet sizes from 32 to 1024 bytes. The 1984 recommendation extends the upper limit to 4096 bytes.	Adds, removes, or changes flow control parameter values, depending on the requirements of the X.25-class services supporting the connection and the X.25 switch service. For information about flow control parameters, see the "Enabling Flow Control Parameter Negotiation" section of the "Configuring X.25 and LAPB" chapter.
Throughput Negotiation • Throughput facility, basic encoding • Throughput facility, extended encoding (1993)	Forwards incoming Throughput facilities.
Closed User Group Selection • CUG facility, basic encoding • CUG facility, extended encoding (1984) • CUG with Outgoing Access facility, basic encoding (1984) • CUG with Outgoing Access facility, extended encoding (1984) • Bilateral CUG facility	Forwards Closed User Group (CUG) selection facilities. If an interface is configured as a DCE that is subscribed to CUG services, all calls that originated and terminated on the interface will be subject to the requirements of CUG security.
Reverse Charging	Forwards the incoming Reverse Charging facility.
Fast Select	Forwards the incoming Fast Select facility.
Internetwork Call Redirection and Deflection (ICRD) Status Selection (1993)	Forwards the ICRD Status Selection facility.
Network User Identification (NUID) (1984)	Forwards the incoming NUID facility.

Table 1 Treatment of Standard X.25 Facilities by Cisco IOS Software (Continued)

Facility	Treatment When Switched by Cisco IOS Software
Charging • Charging request (1984) • Monetary report (1984) • Segment report (1984) • Duration report (1984)	Forwards Charging facilities.
ROA • ROA facility, basic encoding • ROA facility, extended encoding (1984)	Forwards ROA facilities.
Called Line Address Modified Notification (CLAMN) (1984)	Forwards the CLAMN facility. A router will insert a CLAMN facility in the call confirm if the call was routed through a hunt group.
Call Deflection Selection (1988)	Forwards the Call Deflection Selection facility.
Call Redirection or Call Deflection Notification (CRCDN) (1984)	Forwards the CRCDN facility. A router will insert a CRCDN facility in a call that is routed through a hunt group.
Transit Delay (1984)	Forwards the Transit Delay facility.
Marker Facilities, including the following: • Local and remote network marker	Forwards a block of private network facilities preceded by either a local network marker or a remote network maker, presuming that the information following the network marker can be parsed according to the X.25 rules that define encoding for Class A, B, C, and D facilities. An X.25 interface configured for DDN or BFE can encode facilities behind a local network marker. The X.25 switching service will forward these facilities.
• ITU-T Specified DTE facilities marker	An ITU-T Specified DTE facilities marker will be validated according to the facilities defined for the X.25-class service handling the packet.

ITU-T-Specified Marker Facilities

Table 2 describes how CCITT/ITU-T-specified marker facilities are treated when an SVC is routed.

Table 2 Default Treatment of ITU-T-Specified Marker Facilities

Facility	Treatment When Switched by Cisco IOS Software
Calling Address Extension (1984)	Forwards the incoming Calling Address Extension facility.
Called Address Extension (1984)	Forwards the incoming Called Address Extension facility.

Table 2 Default Treatment of ITU-T-Specified Marker Facilities

Facility	Treatment When Switched by Cisco IOS Software
Quality of Service (QoS) Negotiation • Minimum Throughput Class QoS facility, basic encoding (1984) • Minimum Throughput Class QoS facility, extended encoding (1993) • End-to-End Transit Delay QoS facility (1984) • Priority QoS (1988) • Protection QoS (1988)	Forwards the QoS facilities and their associated values.
Expedited Data Negotiation (1984)	Forwards the Expedited Data Negotiation facility.

CMNS hosts commonly use the Called Address Extension facility, which can be used to make X.25 routing decisions.

The encoding of any CCITT/ITU-T facilities is preceded by a marker, as displayed in the output of the **debug x25** command.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0711R)

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

© 2007 Cisco Systems, Inc. All rights reserved.