User Reference Manual

Revision Date: 08.04.04

9032578-27

NOTICE

Enterasys Networks reserves the right to make changes in specifications and other information contained in this document and its web site without prior notice. The reader should in all cases consult Enterasys Networks to determine whether any such changes have been made.

The hardware, firmware, or software described in this document is subject to change without notice.

IN NO EVENT SHALL ENTERASYS NETWORKS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT, WEB SITE, OR THE INFORMATION CONTAINED IN THEM, EVEN IF ENTERASYS NETWORKS HAS BEEN ADVISED OF, KNEW OF, OR SHOULD HAVE KNOWN OF, THE POSSIBILITY OF SUCH DAMAGES.

Enterasys Networks, Inc.
50 Minuteman Road
Andover, MA 01810

Part Number: 9032578-27 August 2004

CODE COPYRIGHTS AND ACKNOWLEDGEMENTS

ENTERASYS NETWORKS, INC.
FIRMWARE LICENSE AGREEMENT

BEFORE OPENING OR UTILIZING THE ENCLOSED PRODUCT, CAREFULLY READ THIS LICENSE AGREEMENT.

This document is an agreement ("Agreement") between the end user ("You") and Enterasys Networks, Inc. on behalf of itself and its Affiliates (as hereinafter defined) ("Enterasys") that sets forth Your rights and obligations with respect to the Enterasys software program/firmware installed on the Enterasys product (including any accompanying documentation, hardware or media) ("Program") in the package and prevails over any additional, conflicting or inconsistent terms and conditions appearing on any purchase order or other document submitted by You. "Affiliate" means any person, partnership, corporation, limited liability company, or other form of enterprise that directly or indirectly through one or more intermediaries, controls, or is controlled by, or is under common control with the party specified. This Agreement constitutes the entire understanding between the parties, and supersedes all prior discussions, representations, understandings or agreements, whether oral or in writing, between the parties with respect to the subject matter of this Agreement. The Program may be contained in firmware, chips or other media.

BY INSTALLING OR OTHERWISE USING THE PROGRAM, YOU REPRESENT THAT YOU ARE AUTHORIZED TO ACCEPT THESE TERMS ON BEHALF OF THE END USER (IF THE END USER IS AN ENTITY ON WHOSE BEHALF YOU ARE AUTHORIZED TO ACT, "YOU" AND "YOUR" SHALL BE DEEMED TO REFER TO SUCH ENTITY) AND THAT YOU AGREE THAT YOU ARE BOUND BY THE TERMS OF THIS AGREEMENT, WHICH INCLUDES, AMONG OTHER PROVISIONS, THE LICENSE, THE DISCLAIMER OF WARRANTY AND THE LIMITATION OF LIABILITY. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT OR ARE NOT AUTHORIZED TO ENTER INTO THIS AGREEMENT, ENTERASYS IS UNWILLING TO LICENSE THE PROGRAM TO YOU AND YOU AGREE TO RETURN THE UNOPENED PRODUCT TO ENTERASYS OR YOUR DEALER, IF ANY, WITHIN TEN (10) DAYS FOLLOWING THE DATE OF RECEIPT FOR A FULL REFUND.

IF YOU HAVE ANY QUESTIONS ABOUT THIS AGREEMENT, CONTACT ENTERASYS NETWORKS, LEGAL DEPARTMENT AT (978) 684-1000.

You and Enterasys agree as follows:

1. LICENSE. You have the non-exclusive and non-transferable right to use only the one (1) copy of the Program provided in this package subject to the terms and conditions of this Agreement.

2. RESTRICTIONS. Except as otherwise authorized in writing by Enterasys, You may not, nor may You permit any third party to:

(i) Reverse engineer, decompile, disassemble or modify the Program, in whole or in part, including for reasons of error correction or interoperability, except to the extent expressly permitted by applicable law and to the extent the parties shall not be permitted by that applicable law, such rights are expressly excluded. Information necessary to achieve interoperability or correct errors is available from Enterasys upon request and upon payment of Enterasys' applicable fee.

(ii) Incorporate the Program, in whole or in part, in any other product or create derivative works based on the Program, in whole or in part.

(iii) Publish, disclose, copy, reproduce or transmit the Program, in whole or in part.

(iv) Assign, sell, license, sublicense, rent, lease, encumber by way of security interest, pledge or otherwise transfer the Program, in whole or in part.

(v) Remove any copyright, trademark, proprietary rights, disclaimer or warning notice included on or embedded in any part of the Program.

3. APPLICABLE LAW. This Agreement shall be interpreted and governed under the laws and in the state and federal courts of the Commonwealth of Massachusetts without regard to its conflicts of laws provisions. You accept the personal jurisdiction and venue of the Commonwealth of Massachusetts courts. None of the 1980 United Nations Convention on Contracts for the International Sale of Goods, the United Nations Convention on the Limitation Period in the International Sale of Goods, and the Uniform Computer Information Transactions Act shall apply to this Agreement.

4. EXPORT RESTRICTIONS. You understand that Enterasys and its Affiliates are subject to regulation by agencies of the U.S. Government, including the U.S. Department of Commerce, which prohibit export or diversion of certain technical products to certain countries, unless a license to export the Program is obtained from the U.S. Government or an exception from obtaining such license may be relied upon by the exporting party.

If the Program is exported from the United States pursuant to the License Exception CIV under the U.S. Export Administration Regulations, You agree that You are a civil end user of the Program and agree that You will use the Program for civil end uses only and not for military purposes.

If the Program is exported from the United States pursuant to the License Exception TSR under the U.S. Export Administration Regulations, in addition to the restriction on transfer set forth in Sections 1 or 2 of this Agreement, You agree not to (i) reexport or release the Program, the source code for the Program or technology to a national of a country in Country Groups D:1 or E:2 (Albania, Armenia, Azerbaijan, Belarus, Bulgaria, Cambodia, Cuba, Estonia, Georgia, Iraq, Kazakhstan, Kyrgyzstan, Laos, Latvia, Libya, Lithuania, Moldova, North Korea, the People's Republic of China, Romania, Russia, Rwanda, Tajikistan, Turkmenistan, Ukraine, Uzbekistan, Vietnam, or such other countries as may be designated by the United States Government), (ii) export to Country Groups D:1 or E:2 (as defined herein) the direct product of the Program or the technology, if such foreign produced direct product is subject to national security controls as identified on the U.S. Commerce Control List, or (iii) if the direct product of the technology is a complete plant or any major component of a plant, export to Country Groups D:1 or E:2 the direct product of the plant or a major component thereof, if such foreign produced direct product is subject to national security controls as identified on the U.S. Commerce Control List or is subject to State Department controls under the U.S. Munitions List.

5. UNITED STATES GOVERNMENT RESTRICTED RIGHTS. The enclosed Program (i) was developed solely at private expense; (ii) contains "restricted computer software" submitted with restricted rights in accordance with section 52.227-19 (a) through (d) of the Commercial Computer Software-Restricted Rights Clause and its successors, and (iii) in all respects is proprietary data belonging to Enterasys and/or its suppliers. For Department of Defense units, the Program is considered commercial computer software in accordance with DFARS section 227.7202-3 and its successors, and use, duplication, or disclosure by the Government is subject to restrictions set forth herein.

6. DISCLAIMER OF WARRANTY. EXCEPT FOR THOSE WARRANTIES EXPRESSLY PROVIDED TO YOU IN WRITING BY ENTERASYS, ENTERASYS DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON- INFRINGEMENT WITH RESPECT TO THE PROGRAM. IF IMPLIED WARRANTIES MAY NOT BE DISCLAIMED BY APPLICABLE LAW, THEN ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO THIRTY (30) DAYS AFTER DELIVERY OF THE PROGRAM TO YOU.

7. LIMITATION OF LIABILITY. IN NO EVENT SHALL ENTERASYS OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS, PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR RELIANCE DAMAGES, OR OTHER LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM, EVEN IF ENTERASYS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS FOREGOING LIMITATION SHALL APPLY REGARDLESS OF THE CAUSE OF ACTION UNDER WHICH DAMAGES ARE SOUGHT.

THE CUMULATIVE LIABILITY OF ENTERASYS TO YOU FOR ALL CLAIMS RELATING TO THE PROGRAM, IN CONTRACT, TORT OR OTHERWISE, SHALL NOT EXCEED THE TOTAL AMOUNT OF FEES PAID TO ENTERASYS BY YOU FOR THE RIGHTS GRANTED HEREIN.

8. AUDIT RIGHTS. You hereby acknowledge that the intellectual property rights associated with the Program are of critical value to Enterasys and, accordingly, You hereby agree to maintain complete books, records and accounts showing (i) license fees due and paid, and (ii) the use, copying and deployment of the Program. You also grant to Enterasys and its authorized representatives, upon reasonable notice, the right to audit and examine during Your normal business hours, Your books, records, accounts and hardware devices upon which the Program may be deployed to verify compliance with this Agreement, including the verification of the license fees due and paid Enterasys and the use, copying and deployment of the Program. Enterasys' right of examination shall be exercised reasonably, in good faith and in a manner calculated to not unreasonably interfere with Your business. In the event such audit discovers non-compliance with this Agreement, including copies of the Program made, used or deployed in breach of this Agreement, You shall promptly pay to Enterasys the appropriate license fees. Enterasys reserves the right, to be exercised in its sole discretion and without prior notice, to terminate this license, effective immediately, for failure to comply with this Agreement. Upon any such termination, You shall immediately cease all use of the Program and shall return to Enterasys the Program and all copies of the Program.

9. OWNERSHIP. This is a license agreement and not an agreement for sale. You acknowledge and agree that the Program constitutes trade secrets and/or copyrighted material of Enterasys and/or its suppliers. You agree to implement reasonable security measures to protect such trade secrets and copyrighted material. All right, title and interest in and to the Program shall remain with Enterasys and/or its suppliers. All rights not specifically granted to You shall be reserved to Enterasys.

10. ENFORCEMENT. You acknowledge and agree that any breach of Sections 2, 4, or 9 of this Agreement by You may cause Enterasys irreparable damage for which recovery of money damages would be inadequate, and that Enterasys may be entitled to seek timely injunctive relief to protect Enterasys' rights under this Agreement in addition to any and all remedies available at law.

11. ASSIGNMENT. You may not assign, transfer or sublicense this Agreement or any of Your rights or obligations under this Agreement, except that You may assign this Agreement to any person or entity which acquires substantially all of Your stock or assets. Enterasys may assign this Agreement in its sole discretion. This Agreement shall be binding upon and inure to the benefit of the parties, their legal representatives, permitted transferees, successors and assigns as permitted by this Agreement. Any attempted assignment, transfer or sublicense in violation of the terms of this Agreement shall be void and a breach of this Agreement.

12. WAIVER. A waiver by Enterasys of a breach of any of the terms and conditions of this Agreement must be in writing and will not be construed as a waiver of any subsequent breach of such term or condition. Enterasys' failure to enforce a term upon Your breach of such term shall not be construed as a waiver of Your breach or prevent enforcement on any other occasion.

13. SEVERABILITY. In the event any provision of this Agreement is found to be invalid, illegal or unenforceable, the validity, legality and enforceability of any of the remaining provisions shall not in any way be affected or impaired thereby, and that provision shall be reformed, construed and enforced to the maximum extent permissible. Any such invalidity, illegality or unenforceability in any jurisdiction shall not invalidate or render illegal or unenforceable such provision in any other jurisdiction.

14. TERMINATION. Enterasys may terminate this Agreement immediately upon Your breach of any of the terms and conditions of this Agreement. Upon any such termination, You shall immediately cease all use of the Program and shall return to Enterasys the Program and all copies of the Program.

Contents

About This Manual

This manual provides information for configuring the Enterasys X-Pedition software. It details the procedures and provides configuration examples. If you have not yet installed the X-Pedition router, use the instructions in the *X-Pedition Getting Started Guide* to install the chassis and perform basic setup tasks, then return to this manual for more detailed configuration information.

Some features are not available with earlier firmware releases or specific hardware platforms. Review the Release Notes for your current firmware release or hardware platform to determine if a feature is supported.

What's New (June 2004)

The E10 system firmware release introduces support for IPv6, and contains an upgrade to the routing stack. The upgrade consists of a number of significant changes, including:

- BGP versions 2 and 3 are not supported. In addition, the **set-pref** option for setting peer groups and peer hosts was removed. Instead, there is a single global command, **bgp set internal-set-pref**, to configure this value.
- DVMRP changes include:
  - Per port snooping at L3 is removed.
  - Can no longer control rate through tunnel or name a tunnel.
  - Prunetime and neighbor-timeout are no longer settable per interface.
- There is one common IGMP implementation, instead of one for DVMRP and one for PIM-SM. As a result, the **pim igmp** commands are removed. Use the corresponding **igmp** commands instead.
- An interface can no longer be configured to be allowed or not allowed membership to a particular group.
- The OSPF capability to configure a totally stubby area has been removed and replaced by the Not So Stubby Area (NSSA) function.
- The only protocol now allowed for creating **aspath-export-source** is BGP.

- Auto-hashing, which determines optimal hashing algorithms for populating L2 and L3 tables, has been changed to detect overflows and adjust the hash algorithm accordingly within 3 seconds and to determine the best algorithm within 24 seconds when traffic is such that all algorithms result in overflows.
- The **mtrace** command was removed.
- The X-Pedition router now supports RFC 2328. Where the router formerly supported RFC 1583, the upgrade could result in changes to existing route tables.

There are a number of CLI command changes, which are described in the *Enterasys X-Pedition NATIVE Command Line Interface Reference Manual*.

The following lists the significant changes within each chapter.

Logging

Facility Support *on page 59*

Bridging

IPv6 and VLANs on page 90

IPV6 (new chapter)

Chapter 13, *IPv6 Configuration Guide*

VRRP

Hashing on page 213

RIPng (new chapter)

Chapter 16, *RIPng Configuration Guide*

OSPF

OSPF Overview on page 233

Configuring an OSPF Area on page 238

Configuring OSPF Area Parameters on page 239

Configuring Autonomous System External (ASE) Link Advertisements on page 240

Displaying OSPF Information on page 242

BGP

Community Attribute Example on page 269

Layer-4 Bridging and Filtering on page 479

WAN

WAN Overview on page 545

Related Documentation

The Enterasys X-Pedition documentation set includes the following items. Refer to these other documents to learn more about your product.

For Information About	See
Installing and setting up the X-Pedition router	*X-Pedition Getting Started Guide*
Syntax for CLI commands	*Enterasys X-Pedition Command Line Interface Reference Manual*

Document Conventions

Commands shown in this manual use the following conventions:

Convention	Description
boldface	Indicates commands and keywords that you enter as shown.
<*italics*>	Indicates arguments for which you supply values.
[**x**] or [<*italics*>] or [**x** <*italics*>]	Keywords and arguments within a set of square brackets are optional.
x|**y**|**z**|<*italics*> or [**x**|**y**|**z**|<*italics*>]	Keywords or arguments separated by vertical bars indicate a choice. Select one keyword or argument.
{**x**|**y**|**z**|<*italics*>}	Braces group required choices. Select one keyword or argument.

Getting Help

For additional support related to the Common CLI syntax or this document, contact Enterasys Networks using one of the following methods:

World Wide Web	www.enterasys.com
Phone	603-332-9400 1-800-872-8440 (toll-free in U.S. and Canada) For the Enterasys Networks Support toll-free number in your country: http://www.enterasys.com/support/gtac-all.html
Internet mail	support@enterasys.com To expedite your message, please type **[ROUTE]** in the subject line.
To send comments or suggestions concerning this document to the Technical Writing Department: **TechWriting@enterasys.com** To expedite your message, please type **[techwriting]** in the subject line, and include the document Part Number in the email message.	

Before contacting Enterasys Networks for technical support, have the following information ready:

- Your Enterasys Networks service contract number
- A description of the failure
- A description of any action(s) already taken to resolve the problem (for example, changing mode switches, rebooting the unit, etc.)
- The serial and revision numbers of all involved Enterasys Networks products in the network
- A description of your network environment (layout, cable type, etc.)
- Network load and frame size at the time of trouble (if known)
- The device history (for example, have you returned the device before, is this a recurring problem, etc.)
- Any previous Return Material Authorization (RMA) numbers

Chapter 1

Maintaining Configuration Files

This chapter provides information about working with configuration files for the Enterasys X-Pedition router. Discussion centers on the different types of configuration files and the procedures involved in changing, displaying, saving, and backing up these files.

Configuration Files

The *X-Pedition Getting Started Guide* introduced the following configuration files:

- **Startup** – The configuration file that the X-Pedition router uses to configure itself when the system is powered on. The Startup configuration remains even when the system is rebooted.
- **Active** – The commands from the Startup configuration file and any configuration commands that you have made active from the scratchpad. The Active configuration remains in effect until you power down or reboot the system.

Caution: The Active configuration remains in effect only during the current power cycle. If you power off or reboot the X-Pedition router without saving the Active configuration changes to the Startup configuration file, the changes are lost.

- **Scratchpad** – The configuration commands you have entered during a CLI session. These commands are temporary and do not become active until you explicitly make them part of the Active configuration.

Changing the Configuration

Before making changes to the Active configuration file, users must enter Configure mode. To enter Configure mode, enter the **configure** command in the CLI from Enable mode. The X-Pedition router places commands entered from Configure mode in the scratchpad file.

Activating the Configuration Commands in the Scratchpad

Commands in the scratchpad do not become active until you explicitly make them part of the Active configuration using the **save active** command. To activate configuration commands located in the scratchpad, do the following:

1. Enter the desired configuration command from **configure** mode on the CLI.
2. Save the command to the Active configuration by entering the following:

```
xp (config)# save active
```

3. The CLI displays the following message:

```
xp (config)# Do you want to make the changes Active? [y]
```

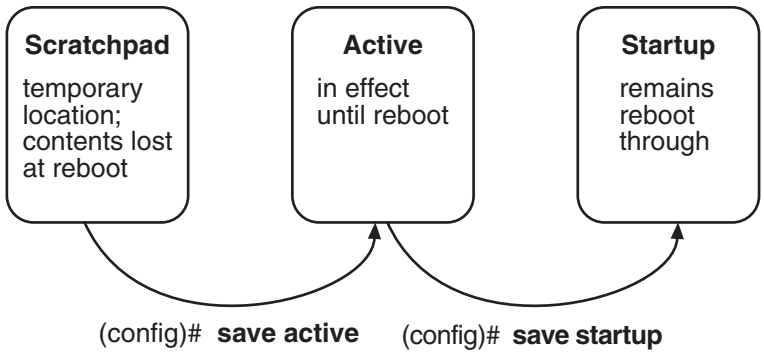
Note: If you attempt to save an empty configuration to the startup file from a Telnet session, you will terminate your connection and the session will end.

4. Type **y** to activate the changes. If you exit Configure mode by entering the **exit** command or pressing **Ctrl+Z** before activating changes the scratchpad, the CLI will ask you whether or not to make the changes active.

Because some commands depend on other commands for successful execution, the X-Pedition router scratchpad simplifies system configuration by allowing you to enter configuration commands in any order, even if dependencies exist. When you activate the commands in the scratchpad, the X-Pedition router sorts out the dependencies and executes the command in the proper sequence.

Figure 1 illustrates the configuration files and the commands you can use to save your configuration:

Figure 1. Commands to Save Configurations



Note: If you create a configuration file on a new version of firmware (e.g., E9.0.7.5 or newer), the configuration may not be 100% compatible with older firmware versions (e.g., E9.0.7.4), resulting in configuration errors.

Removing or Deactivating Configuration Commands in the Active Configuration

To remove or deactivate a configuration command from the Active configuration, enter the **negate** command and specify the line number that contains the command you wish to remove. The X-Pedition router then will insert a “no” string in front of the command and place it in the scratchpad. Entering the **save active** command from the CLI will deactivate, then delete the negated command from the Active configuration.

The Comment Facility

The X-Pedition router provides an alternative way to manipulate the active configuration. The **comment** command allows you to add lines of comments to the Active configuration file, move existing comments, negate/deactivate a command in the Active configuration file (leaving the command in the file as a comment), or reactivate a command previously deactivated by the **comment out** command.

Note: The **comment** commands take effect *immediately*—they do not require the **save active** command to activate them. Because these commands directly modify the Active configuration, use caution when commenting commands into or out of the Active configuration that are dependent upon other commands.

To add comments to the Active configuration file, enter the **comment line** command, the line number where you would like the comment to appear, and the comment string surrounded by quotation marks. For example, to add the comment “this is an ethernet port” to line 10 of the configuration, enter the following (you will note that “C” and “!!” precede all comments):

```
xp# comment line 10 “this is an ethernet port”
10C: !!this is an ethernet port
11   : interface create ip ether1 address-netmask 10.1.1.2/24 port et.4.1
```

To negate or deactivate a command in the Active configuration file and leave it in the file as a comment string, enter the **comment out** command with the line number of the command you wish to comment out. The new comment will appear on the line the command originally occupied. To reactivate a previously negated command with the **comment out** command, enter the **comment in** command with the line number of the command you wish to reactivate. The following example demonstrates how to comment a line back into the configuration that was previously commented out.

```
xp(config)# show
Running system configuration:
!
! Last modified from Console on 2003-10-20 18:08:48
!
1 : smarttrunk create st.1 protocol no-protocol
2C: !!smarttrunk add ports et.5.3-4 to st.1
!
3 : system set name SSR8-9
4 : system set idle-timeout serial 0 telnet 0

xp(config)# comment in 2
xp(config)# show
Running system configuration:
!
! Last modified from Console on 2003-10-20 18:08:48
!
1 : smarttrunk create st.1 protocol no-protocol
2 : smarttrunk add ports et.5.3-4 to st.1
!
3 : system set name SSR8-9
4 : system set idle-timeout serial 0 telnet 0
```

To move a comment string in the Active configuration from one line to another, enter the **comment move** command with the comment's current line number and the target line number where you would like to place the comment string. To remove a comment line from the Active configuration, use the **negate** command. This will remove the comment line immediately and does not require you to enter the **save active** command.

The Copy Facility

The X-Pedition router also allows you to copy entire files between the scratchpad, Active configuration, Startup configuration, TFTP server, RCP server, or URL. To copy a file, enter the **copy** command from Enable mode.

To copy configuration information from the scratchpad to the Active configuration (causing changes to take immediate effect), enter the following command from the CLI.

```
xp# copy scratchpad to active
```

To copy the file **config.john** to the PCMCIA card, slot0:config.debi:.

```
xp# copy config.john to slot0:config.debi
```

To copy the Startup configuration to a TFTP server for backup purposes, enter the following command. The CLI will then prompt the user for the TFTP server's IP address or hostname and the filename:

```
xp# copy startup to tftp-server
TFTP server? 10.136.11.1
Destination filename? my_startup.cfg
```

To copy a previously saved configuration from a TFTP server to the Startup configuration, enter the following command. Note the use of a URL to specify the TFTP server and the filename.

```
xp# copy tftp://10.1.2.3/backup/config.org to startup
```

To copy the Active configuration to a remote server using RCP, enter the following command. Note that this example uses a URL to specify the RCP user name, server, and filename.

```
xp# copy active to rcp://john@server1/config/config.dec25
```

To copy the Startup configuration from the Primary control module to the Backup control module:

```
xp# copy startup to backup-CM
```

Saving the Active Configuration to the Startup Configuration File

When you save commands from the scratchpad to the Active configuration, the X-Pedition router executes the commands and makes them active. However, unless you save the Active configuration to the Startup configuration, any changes will be lost if you power off or reboot the router. Use the following procedure when saving changes to the Startup configuration file to cause the X-Pedition router to reinstate the changes when you reboot.

Configure Mode

1. Enter the following command.

```
xp# save startup
```

2. When the CLI displays the following message, enter **yes** to save the changes:

```
xp# Are you sure you want to overwrite the Startup configuration? [n]
```

Note: If you attempt to save an empty configuration to the startup file from a Telnet session, you will terminate your connection and the session will end. You also can save active changes to the Startup configuration file from within Configure mode by entering the **save startup** command.

Enable Mode

You may obtain similar results to the previous example by entering the following command from Enable mode.

```
xp# copy active to startup
```

The new configuration saves to the Startup configuration file stored in the control modules' boot flash.

Command Summary

The following tables describe the commands you may use to manipulate configuration information.

Table 1. Commands to Change Configuration Information

Task	Command
Enable Mode:	
Copy between scratchpad, Active configuration, Startup configuration, TFTP server, RCP server, or URL.	**copy** <*source*> **to** <*destination*>

Task	Command
Configure Mode:	
Remove comments from the configuration.	**negate** <*line number*>
Comment in (reactivate) the command in the specified line number.	**comment in** <*line number*>
Comment out (deactivate and replace as a comment string) the command in the specified line number.	**comment out** <*line number*>
Place comment string in specified line number.	**comment line** <*line number*> <*string*>
Move comment string from first line number to second line number.	**comment move** <*line number*>, <*line number*>
Erase commands in scratchpad.	**erase scratchpad**
Erase Startup configuration.	**erase startup**
Negate one or more commands by line numbers.	**negate** <*line number*>
Negate commands that match a specified command string.	**no** <*string*>
Save scratchpad to Active configuration.	**save active**

Task	Command
Save Active configuration to startup.	**save startup**

Displaying Configuration Information

The following table lists the commands that are useful for displaying the X-Pedition router's configuration information.

Table 2. Commands to Display Configuration Information

Task	Command
Enable Mode:	
Show Active configuration of the system.	**system show active**
Show the non-activated configuration changes in the scratchpad.	**system show scratchpad**
Show the Startup configuration for the next reboot.	**system show startup**

Task	Command
Configure Mode:	
Show Active configuration of the system.	**show active**
Show the non-activated configuration changes in the scratchpad.	**show scratchpad**
Show the Startup configuration for the next reboot.	**show startup**
Show the running system configuration, followed by the non-activated changes in the scratchpad.	**show**
Compare activated commands with the Startup configuration file.	**diff** <*filename*> **| startup**

The show and system show commands display the commands in the order they were executed. You can change this sequence to alphabetical order by using the **system set show-config** command.

Viewing the Configuration

If you want to view the current configuration:

1. Ensure that you are in Enable mode by entering the **enable** command in the CLI.
2. Enter the following command to display the status of each command line:

```
xp# system show active-config
```

3. The CLI displays the Active configuration file with the following possible annotations:

- Commands without errors are displayed without any annotation.
- Commands with errors are annotated with an “E.”
- Comments are annotated with a “C.”
- If a particular command has been applied such that it can be expanded on additional interfaces/modules, it is annotated with a “P”. For example, if you enable STP on all ports in the current system, but the X-Pedition router contains only one module, then the command to enable STP will be applied at a later date when more modules have been added.

4. A command like **stp enable et.*.*** would be displayed as follows:

```
P: stp enable et.*.*
```

This indicates that the configuration is partially applied. If you add modules to the X-Pedition router at a later date and update the configuration file to encompass all available modules in the X-Pedition router, the “P:” portion of the command line above will disappear when you display the configuration file.

If a command that was originally configured to encompass all of the available modules on the X-Pedition router becomes only partially active (after a hot swap or some such chassis reconfiguration), then the status of that command line automatically changes to indicate a partial completion status, complete with “P:”.

Note: Commands with no annotation or that are annotated with a “P:” are not in error.

Backing up and Restoring the System Image

When you boot up the system, the X-Pedition router boots up the system image off the PCMCIA flash card. The PCMCIA flash card contains the run-time image (as of 3.1, the PCMCIA flash can store up to two images) and the Startup configuration file.

Note: When you use a PCMCIA flash card, place the card in slot 0 of the Control Module. Slot 0 is the top PCMCIA slot on an X-Pedition CM2-64/128, CM3-128, or CM4-256, and the left slot on an ER16-CM3-128 or ER16-CM4-256.

In a dual CM environment, PCMCIA cards installed in Control Modules must contain the same System Firmware version—unless a System Firmware upgrade is in progress.

It is recommended that a backup of the system image be stored on a central server in the unlikely event that the system image becomes corrupted or deleted from the PCMCIA flash card. Use the **system set bootprom** command in Configure mode to set parameters for the X-Pedition router to boot the system image remotely over a network.

system set bootprom netaddr <*IPaddr*> netmask <*IP netmask*> tftp-server <*IPaddr*> tftp-gateway <*IPaddr*>

If the X-Pedition router boots up from the PCMCIA flash card and cannot find a valid image, it goes into boot prom mode. If the en0 interface is configured and connected to a network, you can download an image to the PCMCIA flash by using the **system image add** command in Enable mode. If the en0 interface has not been configured, then you will need to configure it by specifying the following: IP address and netmask of the X-Pedition router, IP address of the TFTP server, and IP address of the default gateway. Use the following command in boot mode:

set net-addr <*IP-address*>
set netmask <*netmask*>
set boot-addr <*tftp-server address*>
set gateway <*IP-address of default gateway*>

Then, boot the X-Pedition router by specifying the following command:

boot <*directory/filename of the image file to boot from*>

Alternatively, you can use the set boot source command:

set boot source <*filename*>

Once the X-Pedition router has booted from the TFTP server image through en0, you can add the new image to the PCMCIA card by using the **system image add** command.

Additionally, you can use the following commands to display, add, and delete system images:

Copy a system software image to the X-Pedition router.	**system image add {tftp-server** <*IPaddr-or-hostname*> **filename** <*filename*>| **tftp-url** <*URL*>**} [destination {backup-cm| primary-cm| all}]**
Select a system software image.	**system image choose** <*filename*> **[backup-cm| primary-cm| all]**
List system software images on the PCMCIA flash card.	**system image list [primary-cm| backup-cm| all]**
Delete a system software image file from the PCMCIA flash card.	**system image delete** <*filename*> **[backup-cm| primary-cm| all]**

Backing up and Restoring the Startup Configuration File

When you save the Startup configuration file, the X-Pedition router stores it in up to four places (three by default): flash memory and PCMCIA card of the primary CM and, if you installed a backup CM, in its flash memory and PCMCIA card as well. Enterasys recommends that you store a backup of the Startup configuration file on a central server and on the X-Pedition router. To copy the Startup configuration to a TFTP server for backup purposes, enter the following command. The CLI will then prompt the user for the TFTP server's IP address or hostname and the filename:

```
xp# copy startup to tftp-server
TFTP server? 10.136.11.1
Destination filename? my_startup.cfg
```

Use the **file** commands to display the configuration files stored in the flash memory of the CM and, if necessary, to delete any of these files:

Display a directory of the files in the flash memory or in the PCMCIA card.	**file dir backup-cm| primary-cm {bootflash:| slot0:} [directory-name]**
Display the contents of a file in the flash memory.	**file type** <*file-name*>
Delete the specified file.	**file delete** <*file-name*>

Note: If you attempt to save an empty configuration to the startup file from a Telnet session, you will terminate your connection and the session will end.

If the startup file becomes corrupted, the X-Pedition router uses its default configuration. You can then use the copy command to copy the backup file to the startup, as shown in the following example:

```
xp# copy startup.bak to startup
```

Backup Control Module Boot Firmware Upgrade

The X-Pedition router's high-availability strategy includes the ability to perform a System and Boot Firmware upgrade. To date, it has been possible to load differing System Firmware images on the Primary and Backup Control Modules (CMs). With E9.0.0.0, it is now possible to load differing Boot Firmware images on the two CMs without a CM fail-over.

With this feature, you may upgrade the system to a new version of firmware without rebooting. While the system is operating with one version of System and Boot Firmware, the Backup CM may be loaded with a compatible but different System and Boot Firmware version. Issuing the **system failover master-cm** command causes control to be transferred from the Primary CM (running one firmware version) to the Backup CM (running another). After the Backup CM initializes, the operational firmware upgrade is complete. Future improvements in CM fail-over time will make firmware upgrades more seamless.

Note: You cannot delete a System Firmware image from the backup Control Module if the image's filename is longer than 20 characters.

In the following examples, the boot-prom image named prom-image is located in a folder in the root directory named images on the TFTP server 10.50.89.88.

To use the tftp-server option to add the prom image file to both control modules, enter the following:

```
xp# system promimage upgrade tftp-server 10.50.89.88 filename images/prom-upgrade
```

To use the tftp-url option to add the prom image file to both control modules, enter the following:

```
xp# system promimage upgrade tft-url tftp://10.50.89.88/images/prom-upgrade
```

To use the tftp-url option to add the prom image file to only the primary control module, enter the following:

```
xp# system promimage upgrade tft-url tftp://10.50.89.88/images/prom-upgrade destination
primary-cm
```

Remote File Boot

If you configure the X-Pedition router to boot from a remote file (i.e., tftp://<*file*>), the router may not be able to reboot itself automatically. This occurs if a problem exists with the remote file (i.e., a server goes down or the file doesn't exist) when the X-Pedition router attempts to reboot. To work around this problem, configure the X-Pedition router to boot from the PCMCIA card.

To configure the router to boot from a file on the PCMCIA card, do the following:

1. Boot the system.
2. To see files currently stored on the PCMCIA card enter the following:

```
xp# system image list
```

3. If the PCMCIA card already contains the file you want to boot from, enter the command below to select that file as the boot source:

```
xp# system image choose <filename>
```

4. If you wish to add a new file to the PCMCIA card, enter the syntax below to select that file as the boot source:

```
xp# system image add {tftp-server <IPaddr-or-hostname> filename <filename>|
tftp-url <URL>} [destination {backup-cm| primary-cm| all}]
```

Configuring System Settings

In addition to the initial settings described in the *Getting Started Guide*, there are additional system features you can set on the X-Pedition router.

Setting Daylight Saving Time

You can set daylight saving time (DST) on the X-Pedition router in three different ways:

- According to specific days. For example, from the first Sunday of April to the last Saturday of October.
- According to specific dates. For example, from April 1st to October 31st.
- By setting the X-Pedition router's time forward by an hour.

When you specify the **system set dst-changing** command or the **system set dst-fixed** command in the Active configuration file, the X-Pedition router automatically updates the time based on the parameters you enter. When a time change occurs, the X-Pedition router automatically sends an informational message about the change. Enter one of the following commands in Configure mode to set DST according to specific days or dates:

Set DST according to specific days.	system set dst-changing s_wk <*value*> s_dow <*value*> s_mo <*value*> s_hr <*value*> s_min <*value*> e_wk <*value*> *e_dow* <*value*> *e_mo* <*value*> *e_hr* <*value*> e_min <*value*>
Set DST according to specific dates.	system set dst-fixed s_mo <*value*> s_day <*value*> s-hr <*value*> s_min <*value*> e_mo <*value*> e_day <*value*> e_hr <*value*> e_min <*value*>

When you set DST by setting the time forward by an hour, saving it to the Active configuration file automatically activates the command and causes the time to move forward one hour. Use the **negate** command to set the time back. Enter the following command in Configure mode to move the time forward by an hour:

Set the time forward by one hour.	system set dst-manual

Configuring a Log-in Banner

Configure the X-Pedition router to display a banner when it is booted up. You can specify a text string or the name of a file on a TFTP server.

Display a log-in banner.	system set login banner [<*string*> | none | file-name <*name*>]

Chapter 2

Virtual File Systems

This chapter provides information about the Virtual File Systems used on the PCMCIA Flash Module.

PCMCIA Virtual File System (VFS)

The Virtual File System is used to format the PCMCIA Flash Module in such a way that the module can organize and store files, and enable you to change Boot and System Firmware images on the module. Enterasys refers to the original virtual file system as VFS1. With Boot Firmware version 1.1.0.8 and System Firmware version 3.0.0.6, Enterasys introduced a second virtual file system (VFS2) to improve system performance and stability. Under typical conditions, VFS1 requires 10-20 minutes to add or delete a System Firmware image—with VFS2, this operation requires a fraction of that time. Because of the myriad of possible network configurations, you may require one VFS over another; the X-Pedition router allows you to switch between VFS1 and VFS2. The following table illustrates firmware and VFS version compatibility.

Firmware Version	Supports VFS1	Supports VFS2
Boot Firmware version 1.0.0.0 and up	YES	NO
Boot Firmware version 1.1.0.8 and up	YES	YES
System Firmware version 1.0.0.0 and up	YES	NO
System Firmware version 3.0.0.6 and up	YES	YES

Note: Moving between VFS1 and VFS2 does not alter files stored on the PCMCIA module.

Changing the Virtual File System

When you move to another VFS, the files stored on the PCMCIA Flash Module will not change. Before you change to another VFS, perform the following actions:

5. Reboot the X-Pedition router. During the Boot Firmware initialization, press the Escape key to stop the System Firmware from booting. (Typically, the Boot Firmware waits two seconds for user interruption before starting to boot the System Firmware.)

6. To display the current VFS version on the PCMCIA module, enter the following command:

```
xp-boot> pcshowversion
```

7. At the Boot Firmware's CLI prompt, enter the following to change from VFS1 to VFS2:

```
xp-boot> pcmakeversion2
```

To convert VFS2 to VFS1, enter the following command:

```
xp-boot> pcmakeversion1
```

Re-initializing the VFS

The file system on the PCMCIA card will not be corrupted under normal circumstances. However, unusual circumstances such as loss of power while adding a System Firmware image may corrupt the PCMCIA card's file system. To re-initialize the PCMCIA card's file system, use the following procedure:

Note: This procedure will erase the PCMCIA card. If the user boots the System Firmware from the PCMCIA card, a copy of the System Firmware image must be placed on a TFTP server so that it may be temporarily booted over the network.

1. Reboot (or power off then on) the X-Pedition router to reach the Boot Firmware's CLI. When the Boot Firmware begins to initialize, press the **Escape** key to stop from booting the System Firmware. Typically, the Boot Firmware waits two seconds for user interruption before starting to boot the System Firmware.

2. After you press the **Escape** key, the X-Pedition router displays the Boot Firmware's CLI prompt. To unmount, erase, and mount (with initialization) the Virtual File System, enter the following commands.

```
xp-boot> pcumount    (unmounts the PCMCIA card's Virtual File System)
xp-boot> erasepcvfs  (erases the PCMCIA card's Virtual File System)
xp-boot> pcmount -i  (mounts and initializes the PCMCIA card's Virtual File System)
```

If the file system on the PCMCIA flash card is not mounted, the **pcumount** command will fail and an error message will appear—you may safely ignore this message. The PCMCIA card's file system is now initialized; however, it lacks a System Firmware image.

Note: If the PCMCIA flash card is write-protected, the **erasepcvfs** command will fail.

3. The Boot Firmware requires that you configure the netaddr, bootaddr, netmask, and/or gateway parameters in order to boot a System Firmware image from a TFTP server over the network (through the "10/100 Mgmt" port). To configure these parameters, use the **set** command as follows:

```
xp-boot> set netaddr <IPaddr-of-management-port>
xp-boot> set bootaddr <IPaddr-of-TFTP-host>
xp-boot> set gateway <IPaddr-of-default-gateway>
xp-boot> set netmask <default-netmask>
```

4. Boot (temporarily) from the System Firmware image. To boot from an image over the network, use the following command:

```
xp-boot> boot <directory> <image-file-name>
```

5. After the System Firmware boots, use the command below to copy the System Firmware image from the TFTP server to the PCMCIA:

```
xp# system image add {tftp-server <IPaddr-or-hostname> filename <filename>| tftp-url <URL>}
[destination {backup-cm | primary-cm | all}]
```

6. Configure the Boot Firmware to boot the System Firmware image added to the PCMCIA card. Enter the following command:

```
xp# system image choose <filename> [backup-cm | primary-cm | all]
```

7. Reboot the system to verify that you replaced the image.

Chapter 3

Hot Swapping

Hot swapping is the ability to replace a line card, Control Module, Switch Fabric (X-Pedition 8600 model only), or GBIC (ER16 only) without disrupting router operations. Hot swapping allows you to remove or install line cards without switching off or rebooting the X-Pedition router, and line cards that are swapped into the X-Pedition router begin functioning immediately after they are installed.

On the X-Pedition 8000 and 8600 models, you can hot swap line cards and Backup control modules. On the 8600, you can also hot swap the secondary switching fabric module. On the ER16, you can hot swap the GBICs, line cards, Backup control modules, and the secondary switch fabric module.

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

This chapter provides instructions for the following tasks:

- Hot swapping line cards
- Hot swapping Primary and Backup Control Modules
- Hot swapping the secondary Switching Fabric Module (8600 and ER16 only)
- Hot swapping a GBIC (ER16 only)

Preventing Electrostatic Damage

The X-Pedition router and line modules are easily damaged by electrostatic discharge. To prevent electrostatic damage, observe the following guidelines:

- Do not remove the module from its packaging until you are ready to install it.
- Do not touch any of the module's pins, connectors or components.
- Hold the module only by its edges or front panel.
- Wear an anti-static wristband connected to a suitable earth ground whenever handling the module.
- Store or transport this module only in appropriate anti-static packaging.

Line Cards

The procedure for hot swapping a line card consists of deactivating the line card, removing it from its slot in the X-Pedition router chassis, and installing a new line card in the slot. When removing a line card from the router, note the following:

- Whenever a user hot swaps any line card out of the router, SmartTRUNK links configured with the Huntgroup protocol may go down for a few seconds.
- All commands associated with a module that is hot swapped out of the router are marked with an "E" in the configuration.

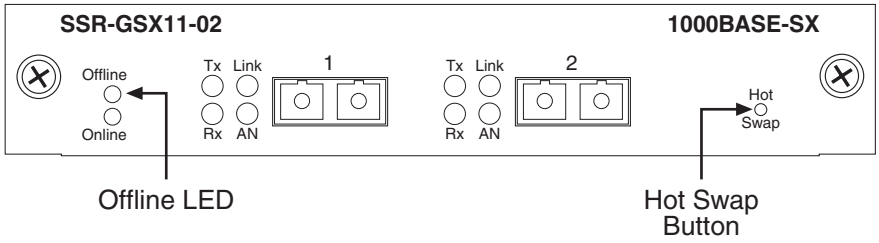
Deactivating the Line Card

To deactivate the line card, use either of the following methods:

- Press the Hot Swap button on the line card. The Hot Swap button is recessed in the line card's front panel—use the tip of a pen or similar object to reach it.

  Before pressing the Hot Swap button, only the online LED is lit. After pressing the Hot Swap button, the offline/online LEDs toggle briefly to indicate that the system has begun the process of hot swapping the line card. Once the router finishes the hot swap process, the offline LED will light and the module will be ready to remove. The amount of time required to hot swap the module depends upon the number of configuration lines that use the module and how busy the system is. Figure 2 shows the location of the Offline LED and Hot Swap button on a 1000Base-SX line card.

Figure 2. Location of Offline LED and Hot Swap Button on a 1000Base-SX Line Card



- Use the **system hotswap out** command from the CLI. For example, to deactivate the line card in slot 7, enter the following command in Enable mode:

```
xp# system hotswap out slot 7
```

After a user enters this command, the Offline LED on the line card lights and messages will appear on the console to indicate that the ports on the line card are inoperative.

Note: To reactivate a deactivated line card (whose offline LED is lit), pull the line card from its slot and push it back in. After the line card re-engages the backplane, the card is activated automatically.

Alternately, if you deactivated the line card with the **system hotswap out** command but did not remove the module from the router, you may enter the **system hotswap in** command to reactivate the module. For example, to reactivate a line card in slot 7, enter the following command from Enable mode:

```
xp# system hotswap in slot 7
```

Removing the Line Card

To remove a line card from the X-Pedition router, do the following:

1. Make sure the Offline LED on the line card is lit. Removing the line card prematurely may cause the router to crash.

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

2. Loosen the captive screws on each side of the line card.
3. Carefully remove the line card from its slot in the X-Pedition router chassis.

Installing a New Line Card

To install a new line card:

1. Slide the line card all the way into the slot, firmly but gently pressing the line card fully in place to ensure that the pins on the back of the line card are completely seated in the backplane.

   Note: Make sure the circuit card (and not the metal plate) is between the card guides. Check both the upper and lower tracks.

2. Tighten the captive screws on each side of the line card to secure it to the chassis.

Once the line card is installed, the X-Pedition router automatically recognizes it, activates it, and lights the online LED.

Control Modules

The Control Module functions as the brain of the router. The X-Pedition 8000, 8600, and ER16 can support up to two control modules (the primary and backup CMs), residing in slots 0 and 1 on the 8000 and 8600, and slots 8 and 9 on the ER16. The Primary CM is the CM in control of the router and processes packets that come through it. The backup CM remains in standby mode, receiving heartbeats from the Primary CM.

Although the backup CM is on standby, it has its PCMCIA card mounted and has booted its firmware image. The backup CM is not aware of the line cards installed in the X-Pedition router and has not processed the startup configuration—both of these operations must be performed in order for the backup CM to assume control of the X-Pedition router (i.e., a *system fail-over*). The time required for the X-Pedition router to switch control to the backup CM depends on the number of line cards installed and the length of the configuration file.

Note: In a dual Control Module configuration, both modules must be of the same type (CM2, CM3, or CM4), share the same Boot Firmware version, and should use the same size and type of memory.

The procedure for hot swapping a Control Module is similar to the procedure for hot swapping a line card. You must deactivate the Control Module, remove it from the X-Pedition router, and insert another Control Module in the slot. If dual CMs are installed on the X-Pedition router, you can hot swap the Backup CM it with another Control Module.

Single CM Environments

The X-Pedition router does not support hot swapping a Control Module in a single CM environment. To replace a Control Module in this environment, do the following:

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

1. Power down the router and make sure that *none* of the LEDs on the Control Module are lit.
2. Loosen the captive screws on each side of the Control Module and, in the case of the ER16, open the ejectors.
3. Carefully remove the Control Module from its slot in the chassis.
4. Slide the new Control Module all the way into the slot reserved for the Primary control module, firmly but gently pressing it in place to ensure that the pins on the back of the card are completely seated in the backplane. Make sure the circuit card (and not the metal plate) is between the card guides. Check both the upper and lower tracks.

Note: On an 8000 or 8600, you can install either a line card or a Control Module in slot CM/1, but you can install *only* a Control Module in slot CM. On the ER16, control modules will operate only in slots 8 and 9—slot 8 is reserved for the Primary control module.

WARNING You must install the PCMCIA card before you install the CM or while the CM is on standby after a hot swap. Do not remove a PCMCIA card while the unit is powered on.

5. Secure the Control Module. To secure the control module in an ER16, close the ejectors and use a flathead screwdriver to tighten the screw on each ejector. For all other routers, align the captive screws with the holes in the face of the router and tighten to secure the module to the chassis.

Dual CM Environments

CM Fail-Over

The problem with using only one CM is that it can fail or get into a hung state from which it may not recover. If the CM fails, you must physically remove and replace the bad CM before the X-Pedition router will be of use again. In order for the X-Pedition router to recover from a hung state, you must turn the power off and on and allow the X-Pedition router to reboot. However, with a backup CM, the X-Pedition router can recover more quickly—without human intervention.

An X-Pedition router with dual CMs operates as it normally would with one CM—until the Primary CM fails. For example, if the Primary CM (in slot 0) fails or hangs, the backup CM (in slot 1) takes control of router operations and restarts the CM in slot 0. Unless the failure was catastrophic, the CM in slot 0 will restart. Once the CM in slot 0 restarts, it will recognize that the backup CM in slot 1 has taken over and assumed the role of the primary CM (the CM in slot 0 then assumes the role of the backup CM). The CM status (primary versus backup) will remain the same until the CM in slot 1 (the current Primary CM) fails or hangs. When this occurs, the process will repeat itself and the CM in slot 0 will again be the Primary CM and the CM in slot 1 will be the Backup CM.

Note: In a dual Control Module configuration, the MAC address of the Primary Control Module in slot “CM/0” is used for both Control Modules after the system is booted. If the Control Module in slot “CM/0” is removed and not replaced after a fail-over, or if it is replaced with a new Control Module and the system is rebooted, the system will use the MAC address of the Control Module in slot 1 (i.e., the new Control Module).

When you install a backup CM, it will only come up part way. The CM will remain on standby until the heartbeat messages from the Primary CM stop coming for at least 4 seconds. Then the Backup CM will take control of the X-Pedition router.

Backup CM Status

To give users a visual indication of the Backup Control Module’s status, the X-Pedition router provides the activity indicator. The activity indicator may display either of the following:

Toggling asterisk (* , *, *, ...)
The Backup CM is receiving heartbeats from the Primary CM and is ready to assume the role of Primary CM in a fail-over situation.

Spinning line (/, –, \, |, ...)
The Backup CM is not receiving heartbeats from the Primary CM. This is common during a system boot while the Backup CM is waiting for the Primary CM to finish booting and for short periods of time during normal system operation. However, if the X-Pedition router displays this indicator for longer than the configured failover timeout value (by default, 5.6 seconds), the Backup CM will take over as Primary CM.

24-7 Networks and VRRP

In most networks, one to five minutes is too long to wait for a system to come back on line—VRRP was created for this reason. The VRRP protocol uses another X-Pedition router to take over in the case of a fail-over. If the backup VRRP router fails to receive messages from the master VRRP router for three seconds (the default), the backup VRRP router will take over. Enterasys recommends the VRRP method for 24-7 networks that use the X-Pedition router for routing.

If you want the added security of having a backup beyond that of VRRP, Enterasys recommends using a backup Control Module. If the Primary CM on the Master VRRP router fails, the backup VRRP router will take over—meanwhile, the backup CM will take over and bring the downed router back up.

Note: As a general rule, VRRP provides the quickest recovery time with layer-3 routing. If you use the X-Pedition router only as a layer-2 switch, you cannot use VRRP and must use a backup CM to provide additional reliability.

Identifying the Primary and Backup Control Modules in a Dual CM Environment

Control Modules may reside in slots CM or CM/1 on the 8000 and 8600, and slots 8 or 9 on the ER16. Except in cases where a system failover has occurred, slot CM on the 8000 and 8600 and slot 8 on the ER16 contain the Primary Control Module. To determine which Control Module is currently operating as the Primary CM and which is the Backup CM, examine the LEDs. On the Primary Control Module, the Online LED is lit—on the Backup Control Module, the Offline LED is lit.

Note: The Offline LED on the Control Module has a different function than the Offline LED on a line card. On a line card, it means that the line card has been deactivated. On a Control Module, a lit Offline LED means that the module is standing by to take over as the Primary CM if necessary; it does *not* mean that the Control Module has been deactivated.

Removing the Primary Control Module

It is important to note that users cannot remove a *Primary* CM by simply pressing the hot swap button or entering the **hot swap** command from the CLI. The Primary CM must first assume the role of Backup CM through a forced failover.

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

1. Force the failover of the Primary CM by entering the following command from the CLI:

```
xp# system failover master-cm
```

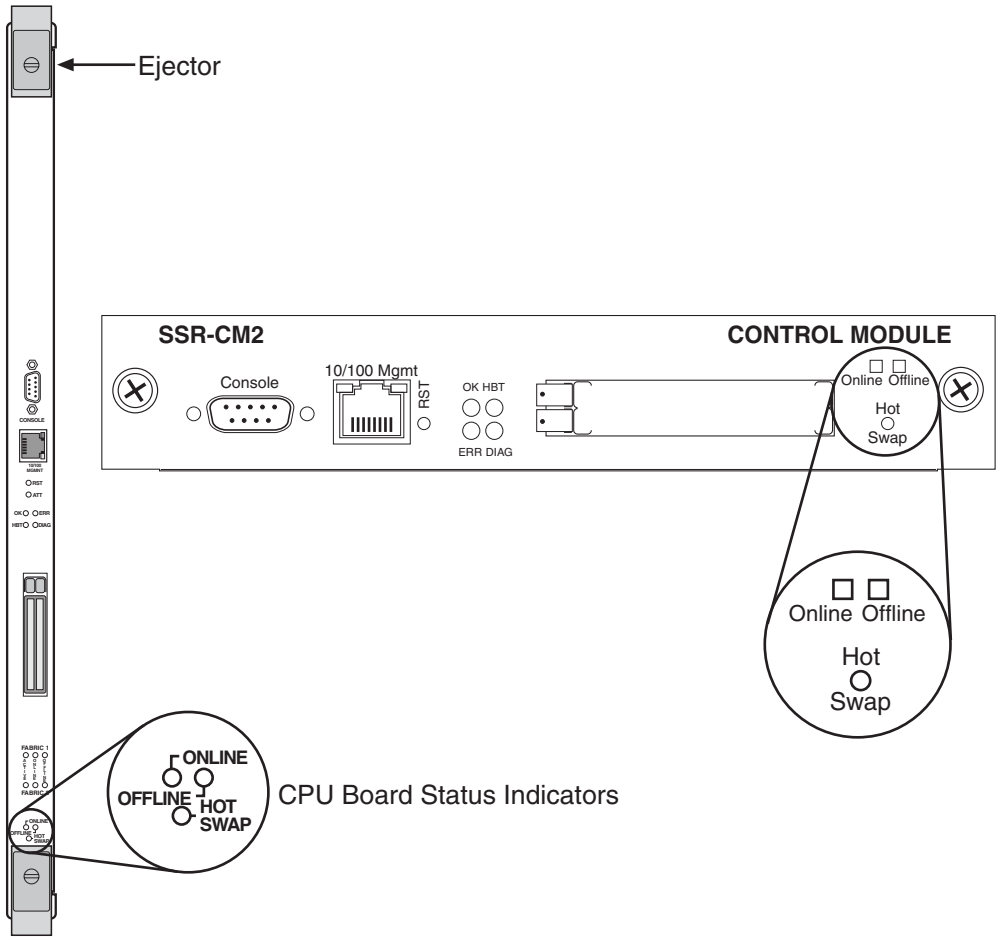
After some minor LED activity the Offline LED will light, indicating that the Control Module has assumed the role of Backup CM.

2. Press the Hot Swap button or enter the following from the CLI:

```
xp# system hotswap out slot <number>
```

3. When the module is ready to remove, the Offline LED will turn ON and all other LEDs will turn OFF. Figure 3 shows the location of the LEDs and Hot Swap buttons on Control Modules.

Figure 3. Location of Offline LED and Hot Swap button on a Control Module



Note: If you have deactivated a Control Module and want to activate it again, simply pull it from its slot and push it back in again. The module is activated automatically. If you deactivated the module with the **system hotswap out** command but did not remove the module, you can reactivate it with the **system hotswap in** command:

```
xp# system hotswap in slot 1
```

Removing the Backup Control Module

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

1. Identify the Backup Control Module.
2. Press the Hot Swap button or enter the following from the CLI:

```
xp# system hotswap out slot <number>
```

3. When the module is ready to remove, the Offline LED will turn ON and all other LEDs will turn OFF. Figure 3 shows the location of the LEDs and Hot Swap buttons on Control Modules.
4. Loosen the captive screws on each side of the Control Module and, in the case of the ER16, open the ejectors.
5. Carefully remove the Control Module from the chassis.

Note: If you deactivated a Control Module and want to activate it again, pull it from its slot and push it back in. The module is activated automatically. If you deactivated the module with the **system hotswap out** command but did not remove the module, you can reactivate it with the **system hotswap in** command:

```
xp# system hotswap in slot 1
```

Installing a Control Module

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

To install a Control Module, do the following:

1. Slide the new Control Module all the way into the slot, firmly but gently pressing it in place to ensure that the pins on the back of the card are completely seated in the backplane. Make sure the circuit card (and not the metal plate) is between the card guides. Check both the upper and lower tracks.

Note: On an 8000 and 8600, you can install either a line card or a Control Module in slot CM/1, but you can install *only* a Control Module in slot CM. On the ER16, control modules will operate only in slots 8 and 9—slot 8 is reserved for the Primary control module.

WARNING You must install the PCMCIA card before you install the CM or while the CM is on standby after a hot swap. Do not remove a PCMCIA card while the unit is powered on. Enterasys recommends using PCMCIA cards of the same size and type when operating in a Dual CM environment.

2. Secure the Control Module. To secure the control module in an ER16, close the ejectors and use a flathead screwdriver to tighten the screw on each ejector. For all other routers, align the captive screws with the holes in the face of the router and tighten to secure the module to the chassis.

Switching Fabric Module (8600 only)

The 8600 has slots for two Switching Fabric Modules. While the 8600 is operating, you can install a second Switching Fabric Module. If two Switching Fabric Modules are installed, you can hot swap one of them. When you remove one of the Switching Fabric Modules, the other goes online and stays online until it is removed or the 8600 is powered off. When the 8600 is powered on again, the Switching Fabric Module in slot "Fabric 1," if one is installed there, becomes the active Switching Fabric Module.

WARNING You can only hot swap a Switching Fabric Module if two are installed on the 8600. If only one Switching Fabric Module is installed, and you remove it, the 8600 will crash.

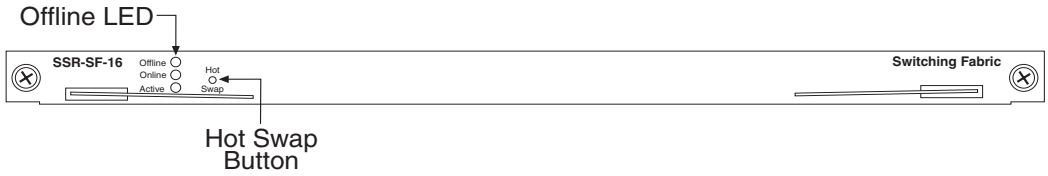
The procedure for hot swapping a Switching Fabric Module is similar to the procedure for hot swapping a line card or Control Module. You deactivate the Switching Fabric Module, remove it from the X-Pedition router, and insert another Switching Fabric Module in the slot.

Note: You cannot deactivate the Switching Fabric Module with the **system hotswap** command.

To deactivate the Switching Fabric Module:

- Press the Hot Swap button on the Switching Fabric Module you want to deactivate. The Online LED goes out and the Offline LED lights. Figure 4 shows the location of the Offline LED and Hot Swap button on a Switching Fabric Module.

Figure 4. Location of Offline LED and Hot Swap Button on a Switching Fabric Module



Removing the Switching Fabric Module

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

To remove the Switching Fabric Module:

1. Loosen the captive screws on each side of the Switching Fabric Module.
2. Pull the metal tabs on the Switching Fabric Module to free it from the connectors holding it in place in the chassis.
3. Carefully remove the Switching Fabric Module from its slot.

Installing a Switching Fabric Module

WARNING The X-Pedition router and its components are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20.

To install a Switching Fabric Module:

1. Slide the Switching Fabric Module all the way into the slot, firmly but gently pressing to ensure that the pins on the back of the module are completely seated in the backplane.

   Note: Make sure the circuit card (and not the metal plate) is between the card guides. Check both the upper and lower tracks.

2. Tighten the captive screws on each side of the Switching Fabric Module to secure it to the chassis.
3. When you install or reseat a Switching Fabric Module, use the following command (in Enable mode) to ensure proper seating and to avoid loop-back failure:

```
xp# system show switching-fabric
```

If the Primary Switch Fabric Module is seated properly, the following message will appear:

```
xp# Switching Fabric: Present, Active
```

If the Secondary Switch Fabric Module is seated properly, the following message will appear:

```
xp# Switching Fabric: Present, Standby
```

GBICs (ER16 only)

The Gigabit Ethernet line cards have slots for GBICs that can be installed at any time. You can hot swap the GBICs installed in the line cards, as well as the line cards themselves. For information on hot swapping line cards, see *Line Cards* on page 20.

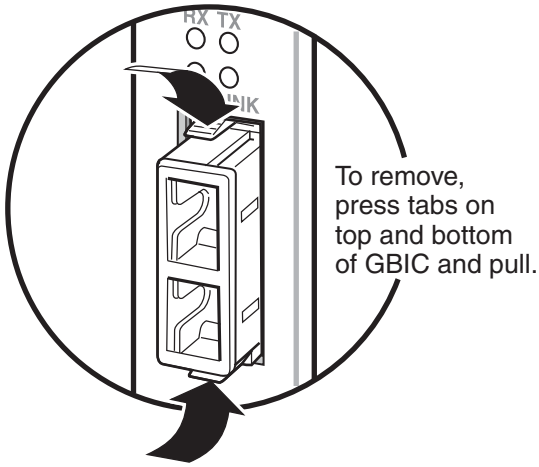
WARNING The GBIC and the host gigabit Ethernet line cards are sensitive to static discharge. To prevent electrostatic damage, observe the guidelines provided in *Preventing Electrostatic Damage* on page 20. Always leave the GBIC in the antistatic bag or an equivalent antistatic container until it is ready to be installed.

Removing a GBIC from the Line Card

To remove a GBIC from its slot on the line card:

1. Remove any cables connected to the GBIC.
2. Locate the extractor tabs on either side of the GBIC.
3. Using thumb and forefinger, compress the extractor tabs on both sides of the GBIC and pull it out of the line card. See Figure 5.
4. If storing or shipping the GBIC, insert the rubber dust protector into the GBIC to protect the fiber ports.

Figure 5. Removing a GBIC

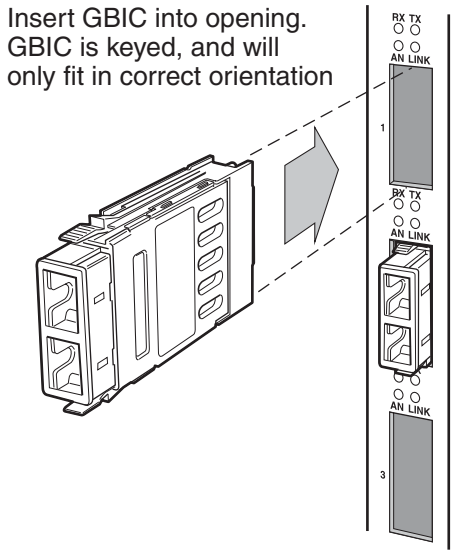


Installing a GBIC into the Line Card

Install the GBIC into the line card as follows:

1. Hold the GBIC with the network port facing away from the line card. The 20-pin connector should be facing toward the empty GBIC slot of the line card.
2. The alignment slot on the GBIC must line up with the alignment guides inside the GBIC slot. The top of the GBIC must be next to the hinged side of the GBIC slot door of the line card.

Figure 6. Installing a GBIC



3. Gently insert the GBIC module into the GBIC slot opening in the line card. The GBIC door on the line card folds in and the hinges engage the alignment slots on the sides of the GBIC module.

Note: If the GBIC module does not go in easily, do not force it. If the GBIC is not oriented properly, it will stop about one quarter of the way into the slot and it should not be forced any further. Remove and reorient the GBIC module so that it slides easily into the slot.

4. Push the GBIC module in until the connector engages the 20-pin port. The GBIC is now installed.

Chapter 4

Using the CLI

This chapter provides information about the X-Pedition Command Line Interface (CLI) used to configure and manage the X-Pedition router. In this manual, example configurations show how to use the CLI commands to configure the router.

CLI commands are grouped by subsystems. For example, the set of commands that allow you configure and display IP routing table information all start with **ip**. Within the set of **ip** commands are commands such as **set**, **show**, **start**, **stop**, **configure**, etc. The complete set of commands for each subsystem is described in the *X-Pedition Command Line Interface Reference Manual*.

Command Modes

The CLI provides access to four different command modes. Each command mode provides a group of related commands. This section describes how to access and list the commands available in each command mode and explains the primary uses for each command mode.

User Mode

After you log in to the X-Pedition router, you are automatically in User mode. The available User commands are a subset of those available in Enable mode. In general, the User commands allow you to display basic information and use basic utilities, such as ping.

The User mode command prompt consists of the X-Pedition name followed by the angle bracket (>), as shown below:

```
xp>
```

The default name is XP unless it has been changed during initial configuration. Refer to the *X-Pedition Getting Started Guide* for the procedures for changing the system name.

Enable Mode

Enable mode provides more facilities than User mode. You can display critical features within Enable mode including router configuration, access control lists, and SNMP statistics. To enter Enable mode from the User mode, enter the command **enable** (or **en**), then supply the password when prompted. If no password is configured, a warning message advising you to configure a password is displayed. If a password is configured and you do not know your password or pressing Return does not work, see the administrator for the X-Pedition router.

The Enable mode command prompt consists of the X-Pedition name followed by the pound sign(#):

```
xp#
```

To exit Enable mode and return to User mode, either type **exit** and press Return, or press Ctrl+Z.

Configure Mode

Configure mode provides the capabilities to configure all features and functions on the X-Pedition router. These include router configuration, access control lists and spanning tree. To enter Configure mode, enter the command **config** from Enable mode.

Note: As mentioned previously, up to four Telnet sessions can be run simultaneously on the X-Pedition router. All four sessions can be in Configure mode at the same time, so you should consider limiting access to the X-Pedition router to authorized users.

The Configure mode command prompt consists of the X-Pedition name followed by **(config)** and a pound sign (#):

```
xp(config)#
```

To exit Configure mode and return to Enable mode, either type **exit** and press Return, or press Ctrl+Z.

Boot PROM Mode

If your X-Pedition router does not find a valid system image on the external PCMCIA flash, the system might enter programmable read-only memory (PROM) mode. If the system is in PROM mode, you should reboot the router (enter the command **reboot** at the boot PROM prompt) to restart the system. If the system fails to reboot successfully, call Enterasys Networks Technical Support to resolve the problem.

For information on how to upgrade the boot PROM software and boot using an upgraded image, see the *X-Pedition Getting Started Guide*.

Native and Common CLI Modes

The X-Pedition router supports two standard interfaces—the Native CLI and the Common CLI. Each mode contains specific commands that are accessible only from within the particular mode. To switch between modes, enter one of the following commands.

Native to Common

```
xp> cli set common
```

Common to Native (Firmware versions E8.2.0.0 and newer)

```
xp> terminal cli native
```

Common to Native (Firmware versions E8.1.x.x and earlier)

```
xp> terminal cli ssr
```

Note: The Common CLI does not support features introduced after E8.0.0.0.

Configuring CLI Access Security

When configuring your network access security policy, Enterasys recommends that you employ at least the following:

- Minimum password length of 8 characters.
- Successful login time of 60 seconds.
- Number of failed login attempts before disabling a user's account should not exceed 6.
- New login attempts cannot be made for at least 60 minutes after disabling a user account.
- Changing administrator-assigned user password after first login (not enabled by default).
- Password lifetime of no more than 90 days.
- User password expiration warning of no more than 14 days.
- Not reusing at least the previous 5 passwords.

Passwords

The rapid growth of e-commerce and the need for customer-facing networks has businesses looking to provide more enhanced security for internal and external systems. Traditionally, a standard username and password was sufficient to prevent unauthorized access to a network—not any more. Previously “secure” password-protected accounts often fall victim to internal or external attack and network integrity is compromised. While passwords enforce rights, access, and ownership on the network when implemented properly, poor password policies contribute to network endangerment. It simply isn’t enough to rely on network users to implement a good password policy of their own.

Creating a Password

Security protocols are put in place to protect a system—however, nothing will lock you out of a system if you provide the proper username and password to access the account. Because usernames are relatively easy to identify (most are e-mail names), potential intruders have only to crack your password to gain access to your system. There are many ways for a prospective intruder to discover a user password—many people use the same code for everything. User account passwords are sacred and should not be shared with anyone—including the system administrator. However, keeping a password secret works only up to a point. When implementing a password, it is crucial that you use a *strong* password.

Weak Passwords

By their nature, network passwords are the weakest link in network security. Typical users prefer a short password that is easy to remember as opposed to one that is long and difficult to remember. This makes it easy for an intruder to guess the password or to hack into the network using a basic computer program—in contrast, a long password that is more difficult to remember is harder to guess and less likely to be compromised.

Types of passwords to avoid:

- User’s name (first or last), child's name, or the name of a pet
- Birthday or anniversary
- “Password”
- Repeated characters (e.g., “AAAAAA” or “999999")
- Sports teams or terms (such as “Bulls” or “Golfer”)
- Favorite recording artist
- Obscenities or sexual terms

In addition to avoiding bad passwords, do not employ bad security practices:

- Do NOT write down the password and post it near the terminal.
- Do NOT use the login name and password of a former employee.
- Make sure that someone besides the network administrator knows the *master account* username and password and tests them periodically. This prevents you from losing access to your network should anything happen to the employee or if the relationship with that employee deteriorates.
- Avoid using the master account for anything but administration. Using this account frequently to perform mundane network operations can lead to unnecessary accidents.
- Many networks use “guest” accounts. Disabling these accounts will help maintain appropriate access for the right people.

Computer programs used to hack passwords fall into two general categories: *dictionary attacks* and *brute force attacks*. **Dictionary attacks** involve using words from a “dictionary” or database of frequently-used terms to attempt to match the user password (e.g., names, cities, and sports teams). **Brute force attacks** focus on discovering passwords via software that generates a sequence of character combinations. For example, guessing a three-character password that uses only alphabetical characters (A - Z) would involve testing all letter combinations between AAA and ZZZ—a total of 17,576 possible combinations.

Strong Passwords

The strength of a password is proportional to its length and complexity—the longer the password, the longer it takes to generate all possible combinations. A strong password is important because password cracking tools continue to improve and the computers used to crack passwords are more powerful than ever. Network passwords that once took weeks to break are now broken in a few hours. Table 3 depicts the relative strength associated with password length and character types allowed.

Table 3. Password Combinations

Number of Characters in Password	Possible Combinations (Letters A-Z only)	Possible Combinations (Letters A-Z, with numbers 0-9)
1	26	36
2	676	1,296
3	17,576	46,656
4	456,976	1,679,616
5	11,881,376	60,466,176
6	308,915,776	2,176,782,336
7	8,031,810,176	78,364,164,096
8	208,827,064,576	2,821,109,907,456
9	5,429,503,678,976	101,559,956,668,416
10	141,167,095,653,376	3,656,158,440,062,980

Good network security includes the use of strong passwords. Although no password is infallible, a strong password equals a more secure system. For a password to be strong and hard to break, it should:

- Be at least seven characters long.
- Contain characters from each of the following groups:

Description	Examples
Letters (uppercase and lowercase)	A, B, C,...; a, b, c,...
Numerals	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Symbols	` ~ ! @ # $ % ^ & * ( ) _ + - = { } | [ ] \ : " ; ' < > ? , . /

- Have at least one symbol character in the second through sixth positions.
- Be significantly different from prior passwords.
- Not contain your name or user name.
- Not be a common word or name.

Password Policy Management

Secure access to the X-Pedition router through password protection and policies is available in both single- and multi-user modes. Global password policies are established using the **system set password-policy** command and apply to all passwords in single- or multi-user mode unless specifically overridden by one of the command options described below.

The following policy elements are configurable in single-user and multi-user modes:

- Enable or disable password policy verification (**enabled** by default)
- Minimum password length (by default, **8 characters**)
- Access management including:
  - The amount of time a user has to log in successfully (by default, **60 seconds**)
  - The number of login failures to allow before disabling a user's account (by default, **6**)
  - After an account is disabled, the amount of time that must pass before permitting another login attempt (by default, **60 minutes**)
- Password change management including:
  - Require a user to change the password following the first login (**off** by default)
  - The number of days the password is valid (by default, **90** days)
  - The number of days prior to password expiration to warn the user of the pending expiration (by default, **14 days**)
  - The number of previous passwords that cannot be re-used (by default, **5**)

Single-User Mode

Creating a Password

By default, the router operates in single-user mode with password access enabled and no passwords defined. To define a password for Login, Enable, or Configure mode, use the **system set password** command from Configure mode. The following example sets an Enable mode password:

```
xp(config)# system set password enable MyPassword
```

Note: Passwords are case *sensitive*. In other words, the X-Pedition router recognizes upper- and lower-case letters as different characters (e.g., “M” is not the same as “m”).

You must set the password for each mode individually (i.e., you may use a different password for each mode). If a password is configured for Enable mode, the X-Pedition router asks for the password when you enter the **enable** command. If no password is defined, the router will advise you to configure a password, then switch to Enable mode—from here you can access Configure mode and make configuration changes. Access to Configuration mode may be configured to require a password.

The router stores passwords in the startup configuration file. If you copy a configuration file from one X-Pedition router to another, the passwords in the file will automatically apply to the new router. When you activate a new password by copying the **system set password** command to the active configuration, the router replaces the command with a **system set hashed-password** command to hide the password text in the configuration file. This prevents others who access this file from viewing the password. To remove the Enable mode password defined above, enter the following command from Configure mode:

```
xp(config)# system set password enable none
```

Password Policies

Once a password is established, the default password policies apply unless configured otherwise. The following example requires a user to change his password after logging in for the first time. All remaining password policies use the default values.

```
xp(config)# system set password-policy change-after-first-login on
```

In the next example, the password expiration notice changes to 10 days, the login failure retry time shortens to 30 minutes, and the minimum password length changes to 6 characters. All other parameters use their default values.

```
xp(config)# system set password-policy expire-warning 10 login-failure-grace-time 30
minimum-length 6
```

Note: Passwords will appear on the console as asterisks to prevent them from being observed.

Multi-User Mode

Multi-user mode password security employs individual user accounts to grant CLI permissions on a case-by-case basis—this requires that each user log in via username and password. The X-Pedition router supports up to 256 concurrent user accounts (although you may still employ either protocol, multi-user accounting no longer requires TACACS+ or RADIUS). To define a password for Login, Enable, or Configure mode, use the **system set password** command from Configure mode. In multi-user mode, you may configure the following information for each user account:

- Username
- New password
- Password lifetime limit
- Mode or privilege where the user can gain access to the system
- Whether or not to disable the user account after too many failed login attempts.

The following example creates a new user account for Jane and grants password access to configure mode. All other options remain at their default levels. For details on specific password and privilege options associated with the **system set user** command, consult the *X-Pedition Command Line Interface Reference Manual*.

```
xp(config)# system set user Jane privilege-level config
```

Note: You must always enter a password when creating an account for a new user or when changing the access privileges for an existing user. Once users are configured, the login prompt changes to request the username and password.

Passwords are case *sensitive*. In other words, the X-Pedition router recognizes upper- and lower-case letters as different characters (e.g., “M” is not the same as “m”).

You must set the password for each mode individually (i.e., you may use a different password for each mode). If a password is configured for Enable mode, the router asks for the password when you enter the **enable** command. If no password is defined, the router will advise you to configure a password, then switch to Enable mode—from here you can access Configure mode and make configuration changes. Access to Configuration mode may be configured to require a password.

In addition to configuring multiple user accounts, you may enable the Syslog audit trail feature to track all user attempts to access the router or make configuration changes. Audit log messages record information such as the username, source IP address, and session type, and are especially useful when working with network problems and evaluating changes to system configuration over time. Although audit trails can be applied to single-user transactions, they are particularly useful in multi-user environments. For more information, see *Audit Trail* *on page 66*.

Establishing Telnet Sessions

You can establish a management connection to the X-Pedition router by connecting a terminal to the management port of the router and by establishing a telnet connection to a remote host. To establish a telnet connection, connect your network to the 10/100 MDI port on the router.

Note: When busy, the X-Pedition router may not be able to establish a Telnet connection.

The router allows up to four simultaneous telnet sessions. There are commands that allow you to monitor telnet use and to end a specific telnet session. You can also specify the number of minutes a serial or telnet connection can remain idle before the connection is terminated by the control module. The default is 5 minutes. You can disable this feature, by setting the time-out value to zero.

Display the last five connections to the router.	**system show telnet-access**
Specify time-out value for a serial or telnet connection.	**system set idle-time-out serial|telnet** <*num*>
End the specified telnet session.	**system kill telnet-session** <*session-id*>

Additionally, you can telnet to another X-Pedition router during a CLI session. To start a telnet session to another router, enter the following command in User or Enable mode.

Open a telnet session to another X-Pedition router.	**telnet** <*hostname-or-IPaddr*> [**socket** <*socket-number*>]

To end your telnet session, type “exit.”

Secure Shell (SSH) Server

Secure Shell (SSH) is a “secure” replacement for Telnet. When using Telnet, all communications, including passwords, are sent across the network in clear-text (i.e., unencrypted). This can make eavesdropping on communications a trivial task for a knowledgeable user with access to the network. SSH provides the same remote access to the router that Telnet provides, but encrypts all session data, including passwords. SSH also provides these security features:

- **Public-key authentication of the server**. This feature enables the client to validate the server’s authenticity, making it difficult for an attacker to masquerade as the server.
- **Digitally *signing* all packets**. This feature uses cryptographically strong message digests to authenticate all communications, preventing an attacker from successfully intercepting and altering information.

Note: When you enable Secure Shell server, the X-Pedition router automatically disables Telnet access to prevent security holes that might accidentally remain open after securing the network. Likewise, when you disable the SSH server or if the SSH server fails to start due to an error, the router automatically re-enables Telnet access.

Configuring Secure Shell

Although there are many ways to configure Secure Shell on the X-Pedition router, initial setup is very easy. No configuration is necessary in order to use the SSH client and, if you do not specify any of the configuration options available for SSH, the server will operate in its default configuration. This configuration is compatible with all supported protocols and algorithms, and allows the widest compatibility—however, the default configuration is not necessarily the most secure configuration.

Note: SSH client requires firmware version E9.1.0.0 or later.

To set up the SSH server, use the following steps:

1. Generate host keys.
2. Enable the SSH server.

The easiest way to generate host keys is to use the **all** option of the **ssh-server generate-host-key** command from enable mode. This command may take a few minutes to complete, but generates all of the keys necessary to interoperate with any SSH client—regardless of which version of the SSH protocol the client uses. See *SSH Protocol Versions* on page 42 for information about these protocol versions.

```
xp# ssh-server generate-host-key all
```

To enable the SSH server, enter the **ssh-server enable** command from configuration mode:

```
xp(config)# ssh-server enable
```

SSH Protocol Versions

There are two major versions of the Secure Shell protocol in use: SSH-1 and SSH-2. While the X-Pedition router supports both versions, SSH-2 is considered to be more secure than SSH-1—SSH-1 is supported for backward compatibility. The following is a brief overview of SSH protocols, highlighting the major differences between them as they relate to security.

In both protocols, the SSH server authenticates itself to the client through a *host key* (see *Host Keys* on page 43). After it verifies server authenticity, the client generates a session key to use until it disconnects from the server. The session key is kept secret through one of the following methods:

- SSH-2 uses the Diffie-Hellman key agreement method to generate a secure session key known only to the client and server. A third party who attempts to eavesdrop on communications that employ this method cannot derive the session key.
- SSH-1 clients use a combination of the host and server keys (see *Server Keys* on page 45) to encrypt the session key and send it to the server.

Once the client and server have copies of the session key, they will use the key to encrypt all further communications. In addition to encrypting each packet, both the client and server will stamp each outgoing packet with data that can be used to validate the contents of the packets.

- In SSH-2, the stamp consists of a message authentication code (MAC) created by using a secure message digest algorithm such as SHA-1 or MD5. The MAC is used by the packet recipient to authenticate the contents of each packet. If the content of the packet changes en-route, MAC authentication will fail (see *Message Authentication Codes (MACs)* on page 46).
- In SSH-1, the stamp is a cyclic redundancy check (CRC-32) of the packet contents. This type of data validation is commonly used for error detection purposes but does not hold up to cryptographic standards because it is feasible for an attacker to change the contents of the packet while maintaining a valid CRC-32 stamp. The X-Pedition router protects against this vulnerability by including a CRC compensation attack detector, but the detector cannot detect all possible CRC compensation attacks; however, it helps make SSH-1 sessions more secure.

Use the **ssh-server set protocol-version** command to specify which protocols to allow on the SSH server. By default, both SSH-1 and SSH-2 are allowed. The following example uses the **ssh-server set protocol-version** command to allow only the SSH-2 protocol:

```
xp(config)# ssh-server set protocol-version ssh2
```

The **ssh-client set protocol-version-preference** command or the **protocol-version-preference** command-line option can be used to specify which protocols to allow on the client and the preferred order in which to apply them. By default, the client allows both SSH-1 and SSH-2 protocols, but prefers SSH-2. The following example allows both protocols, but demonstrates how to change the preference to SSH-1:

Note: SSH client requires firmware version E9.1.0.0 or later.

```
xp(config)# ssh-client set protocol-version-preference ssh1 ssh2
```

This can also be accomplished on a per-session basis by using the **protocol-version-preference** command-line option:

```
xp# ssh 192.168.1.1 protocol-version-preference ssh1,ssh2
```

Host Keys

Host keys are asymmetric encryption keys commonly used in what is known as *public key cryptography*. In both versions of the Secure Shell protocol, the SSH server uses unique host keys. Each host key consists of a pair of keys, generated simultaneously—although the generated keys are related, one cannot be derived from the other. The first key of the generated pair, the *public key*, can be published freely and is used by SSH clients to securely identify the SSH server. The second key of the generated pair, the *secret key*, is stored in a safe place and should never be divulged. This key is used by the SSH server to securely identify itself to SSH clients.

The SSH-2 protocol makes two distinct types of host keys available: the Digital Signature Algorithm (DSA) and the Rivest-Shamir-Adleman (RSA) algorithm. DSA was developed for the Digital Signature Standard program administered by the National Institute of Standards and Technology (NIST). RSA, whose previous patent expired, was added to the SSH-2 protocol to widen available security options. Both DSA and RSA are NIST-approved digital signature algorithms.

The SSH-1 protocol uses only one type of host key—an RSA1 key. The RSA1 key is an RSA key stored in a format that is compatible with the SSH-1 protocol.

To generate host keys, use the **ssh-server generate-host-key** command from enable mode. The following example demonstrates how to generate a 768-bit RSA host key:

```
xp# ssh-server generate-host-key rsa bits 768
```

When selecting the bit size for the host key, keep in mind that, in general, 1,024 bits (the default) is considered very secure. Lengths greater than 1,024 bits are not considered to provide much additional security and will slow down cryptographic operations. For example, keys that are 1,024 bits or less in size take only a few minutes to generate. In contrast, keys larger than 1,024 bits may take several hours to generate (e.g., 4,096 bit keys may require several hours).

Before enabling the SSH server, at least one host key must be generated and the host key must be compatible with the protocol version used. See *SSH Protocol Versions* on page 42 and *Configuring Secure Shell* on page 42.

In order to securely identify SSH servers to which it connects, the SSH client keeps track of host keys and the servers to which they belong. The X-Pedition router stores host keys in the "**Known Hosts**" database and provides a set of commands for managing the database. Because the Known Hosts database plays a crucial role in preventing certain types of attacks, it must be kept safe from tampering by unauthorized individuals. To facilitate this, the commands used to manage the Known Hosts database can be executed only by users with configuration-level privileges—for convenience, however, the default configuration allows any user to automatically add new keys to the Known Hosts database by actually establishing an SSH connection to a new or unknown host. For installations that require the highest level of security, enable "strict" host key checking to prevent users from automatically adding to the Known Hosts database. To enable strict host key checking, add the **ssh-client set strict-host-key-checking** command to the configuration.

The Known Hosts database can be cleared by using the **ssh-client clear-known-hosts** command. This command is useful for resetting the Known Hosts database if unauthorized alterations to the database are detected. To remove old or outdated keys from the database without resetting the entire database, use the **ssh-client delete-host-keys** command. To import new host keys individually or in bulk from a text file stored on the router, use the **ssh-client import-host-keys** command. Please consult the Native CLI Reference Manual for details on importing keys from stored files and the file format used.

The Known Hosts database resides on the external PCMCIA flash card. Therefore, the added security of the Known Hosts database and the host key management functions require that a PCMCIA flash card is present in the system chassis.

Server Keys

When an SSH-1 protocol session begins, the client generates a session key to use to encrypt all session data—the type of session key used depends on the encryption algorithm enabled (see *Encryption Algorithms (Ciphers)* on page 45). Once it generates the session key, the client provides a copy of the key to the server by encrypting it with the host key and the *server key*. On the X-Pedition router, the server key is always an asymmetric 768-bit RSA key composed of a public and secret key pair. The SSH server passes the public part of the server key pair, along with the public part of the host key pair, to the client.

The server key provides *perfect forward secrecy*, meaning that the security of previous sessions will not be compromised—even if the host or session key is compromised. Since the server key regenerates periodically, the impact of a compromised key is minimal. The more often the server key regenerates, the more secure it will be; however, frequent regenerating requires more system resources. The configuration mode command, **ssh-server set server-key-lifetime** controls the amount of time that will elapse before the server key regenerates. The following example shows how to regenerate the server key every 30 minutes. By default, the server key regenerates every 60 minutes:

```
xp(config)# ssh-server set server-key-lifetime 30
```

The server key is used only for the SSH-1 protocol—if only the SSH-2 protocol is enabled, the X-Pedition router will not use the server key and it will never regenerate. The SSH-2 protocol uses a method known as Diffie-Hellman key agreement which allows the client and server to arrive at a shared session key without actually sending the key across the network. Any third party that may eavesdrop on the Diffie-Hellman message exchange will not be able to determine the session key.

Encryption Algorithms (Ciphers)

The X-Pedition router uses several encryption algorithms or *ciphers* to encrypt session data. Commonly referred to as “bulk” ciphers, these algorithms can encrypt large quantities of data and are faster than other encryption methods such as public-key encryption. However, the keys used by these bulk ciphers to formulate the session key are not asymmetric—both the server and the client will share a copy of the same key. Because each session generates its own key, if a session is compromised, other sessions remain secure.

Use the **ssh-server set encryption** command in configuration mode to select which bulk ciphers to allow. When a client connects, it will select which cipher to use from the list of ciphers specified by this command. The following example demonstrates how to enable the Blowfish and AES ciphers:

```
xp(config)# ssh-server set encryption blowfish aes128-cbc
```

Note: The list of ciphers specified by this command will apply to both SSH-2 and SSH-1 sessions. If this command is not added to the configuration, the X-Pedition router will allow all available ciphers.

Use the **ssh-client set encryption-preference** command or the **encryption-preference** command-line option to select the SSH-2 ciphers to attempt, in order of preference. The following example uses the **ssh-client set encryption preference** configuration command to set 3DES, AES, and Blowfish as the only ciphers to attempt (with AES being the most preferred cipher).

```
xp(config)# ssh-client set encryption-preference aes128-cbc blowfish 3des-cbc
```

You may also accomplish this task on a per-session basis by using the **encryption-preference** command-line option:

```
xp# ssh 192.168.1.1 encryption-preference aes128-cbc,blowfish,3des-cbc
```

For the SSH-1 protocol version, users cannot specify a preference for encryption algorithms. However, users may use the **ssh-client set ssh1-encryption** command or the **ssh1-encryption** command-line option to force use of a particular cipher. The following example uses the **ssh-client set ssh1-encryption** configuration command to force the use of the Blowfish cipher:

```
xp(config)# ssh-client set ssh1-encryption blowfish
```

This may also be accomplished on a per-session basis with the **ssh1-encryption** command-line option:

```
xp# ssh 192.168.1.1 ssh1-encryption blowfish
```

Message Authentication Codes (MACs)

In addition to the available ciphers, there are also several Message Authentication Code (MAC) algorithms available to SSH-2 sessions. These MACs are used to authenticate the contents of each packet of data exchanged between the client and server. If a third party tampers with the contents of a packet, MAC validation will fail and the tampering will be detected.

Use the **ssh-server set mac** command in configuration mode to select which MACs to allow. When a client connects, it will select which MAC to use from the list of MACs specified by this command. The following example demonstrates how to enable the HMAC-SHA1 and HMAC-SHA1-96 MAC algorithms:

```
xp(config)# ssh-server set mac hmac-sha1 hmac-sha1-96
```

Note: The list of MACs specified by this command will apply to SSH-2 sessions only. The SSH-1 protocol does not support MACs and will use only CRC-32 validation. If this command is not added to the configuration, the X-Pedition router will allow all available MACs.

Use the **ssh-client set mac-preference** command or the **mac-preference** command-line option to select the SSH-2 MACs to be attempted, in order of preference. The following example uses the **ssh-client set mac-preference** configuration command to set HMAC-SHA1 and HMAC-SHA1-96 as the only ciphers to attempt, with HMAC-SHA1-96 being the most preferred MAC.

```
xp(config)# ssh-client set mac-preference hmac-sha1-96 hmac-sha1
```

This may also be accomplished on a per-session basis with the **mac-preference** command-line option:

```
xp# ssh 192.168.1.1 mac-preference hmac-sha1-96,hmac-sha1
```

Additional Options

Each active Secure Shell session can use a fair amount of system resources. To what degree each session will impact overall system performance depends on the protocol versions, ciphers, and MACs in use, and the amount of activity in each session. To minimize the amount of system resources consumed by SSH sessions, the X-Pedition router allows users to limit the number of concurrent sessions.

The following example uses the **ssh-server set max-sessions** command from configuration mode to limit the number of concurrent SSH sessions to two. By default, the X-Pedition router allows up to four concurrent SSH sessions.

```
xp(config)# ssh-server set max-sessions 2
```

SSH servers usually listen for incoming connection requests on TCP port 22 (the default). To configure the X-Pedition router to listen for connection requests on a TCP port other than the default, use the **ssh-server set listen-port** command from the configuration mode. The following example shows how to configure the SSH server to listen on TCP port 4096:

```
xp(config)# ssh-server set listen-port 4096
```

To change the port to which you will connect on the target host for SSH client sessions, use the **ssh-client set port** command or the **port** command-line option:

```
xp(config)# ssh-client set port 4096
```

or

```
xp# ssh 192.168.1.1 port 4096
```

By default, SSH session data compression is always enabled. To disable data compression with the client, use the **ssh-client set no-compression** configuration command or specify the **no-compression** command-line option:

```
xp(config)# ssh-client set no-compression
```

or

```
xp# ssh 192.168.1.1 no-compression
```

By default, the *tilde* (~) character initiates the escape sequence. Typing "~?" immediately following a new line while in an active SSH client session will display a menu of all the available escape sequences. The escape sequence initiator character can be changed by using the **ssh-client set escape** configuration command, or by specifying the **escape** command-line option. The following example shows how to change the escape sequence initiator to Control + right square bracket:

```
xp(config)# ssh-client set escape ^]
```

or

```
xp# ssh 192.168.1.1 escape ^]
```

Under normal circumstances, the SSH client will pre-determine the username to use when authenticating with a remote SSH server by checking the name of the user currently logged-in. However, if no user accounts are configured and neither RADIUS nor TACACS+ is configured, the client will prompt the user for a username before establishing the SSH connection. Alternately, you can specify a username with the **ssh-client set username** configuration command or by specifying the **login-as** command-line option:

```
xp(config)# ssh-client set username foo
```

or

```
xp# ssh 192.168.1.1 login-as foo
```

Setting CLI Parameters

The X-Pedition router provides various commands for controlling the behavior and display of the CLI. The **cli set command completion** command controls the behavior of the CLI when you enter commands. When you turn on command completion, the CLI attempts to automatically complete a command that is partially entered. Typing enough characters of a command keyword to uniquely identify it and pressing the space bar to move to the next word causes the CLI to complete the command word and move on.

To set command completion, enter the following command in either Configure mode or Enable mode. In Configure mode, the command turns on or off command completion for the entire system. In Enable mode, the command affects the current login session of the user issuing the command.

Turn on or turn off command completion.	**cli set command completion on|off**

The **cli set history** command specifies the number of commands that will be stored in the command history buffer. Commands stored in the buffer can be recalled without having to type the complete command again. When you hit the ↑ key, the CLI displays the commands that were entered, from the most recent. To specify the number of commands stored in the command history buffer, enter the following command in User or Configure mode.

Set the size of the command history buffer.	**cli set history size** <*num*>|**default**|**maxsize**

Alternatively, you can display all the commands that were executed during a CLI session. To display the CLI commands, enter the following command in User mode.

Display command history.	**cli show history**

The CLI also provides commands for setting the terminal display. Use the following commands to set and display terminal settings.

Task	**Command**
User Mode	
Set the terminal display.	**cli set terminal rows** <*num*> **columns** <*num*>
Display terminal settings.	**cli show terminal**
Enable Mode	
Display system and diagnostic/tracing messages.	**cli terminal monitor on|off**

Getting Help with CLI Commands

The CLI provides interactive help, a search capability, and line editing.

CLI Help

Interactive help is available from CLI by entering the question mark (?) character at any time. The help is context-sensitive; the help provided is based on where in the command you are. For example, if you are at the User mode prompt, enter a question mark (?) to list the commands available in User mode.

You can also type the **?** character while entering in a command line to see a description of the parameters or options that you can enter. Once the help information is displayed, the command line is redisplayed as before but without the ? character. The following is an example of invoking help while entering a command:

```
xp(config)# load-balance create ?
 group-name              - Name of this Load Balanced group of servers
 vip-range-name          - Name of this Virtual IP range
xp(config)# load-balance create
```

If you enter enough characters of a command keyword to uniquely identify it and press the space bar, the CLI attempts to complete the command. If you do not enter enough characters or you enter the wrong characters, CLI cannot complete the command. For example, if you enter the following in Enable mode and press the spacebar as indicated:

```
xp# system show e[space]
```

CLI completes the command as follows:

```
xp# system show environmental
```

If entering several commands for the same subsystem, you can enter the subsystem name from the CLI. Then, execute individual commands for the subsystem without typing the subsystem name in each time. For example, if you are configuring several entries for the IP routing table, you can enter **ip** at the CLI Configure prompt and click enter. The prompt changes to indicate that the context for the commands to be entered has changed to that of the IP subsystem. If you type a ?, only those commands that are valid for the IP subsystem are displayed. The following is an example:

```
xp(config)# ip
xp(config)(ip)# ?
add               - Add a static route
 dos              - Configure specific denial of service features
 disable          - Disable certain IP function
 enable           - Enable certain IP function
 helper-address   - Specify IP helper address for an interface
 l3-hash          - Change IP hash variant for channel
 set              - Set ip stack properties
 Ctrl-z           - Exits to previous level
 top              - Exits to the top level
xp(config)(ip)# [Ctrl-Z]
xp(config)#
```

CLI Search

If you are unsure of a command structure but know a keyword, you can perform a search on the command line interface, including the CLI help, using the **cli search** command. The command displays where the text string was found.

Line Editing Commands

The X-Pedition router provides line editing capabilities that are similar to Emacs, a Unix text editor. For example, you can use certain line editing keystrokes to move forward or backward on a line, delete or transpose characters, and delete portions of a line. To use the line editing commands, you need to have a VT-100 terminal or terminal emulator. The line editing commands that you can use with CLI are detailed in Table 4.

Table 4. CLI Line Editing Commands

Command	Resulting Action
Ctrl-a	Move to beginning of line
Ctrl-b	Move back one character
Ctrl-c	Abort current line
Ctrl-d	Delete character under cursor
Ctrl-e	Move to end of line
Ctrl-f	Move forward one character
Ctrl-g	Abort current line

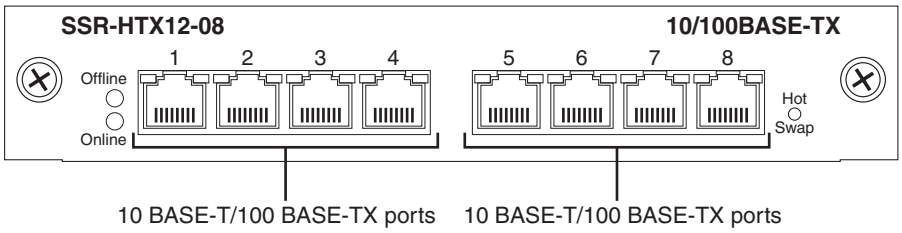
Table 4. CLI Line Editing Commands (Continued)

Command	Resulting Action
Ctrl-h	Delete character just prior to the cursor
Ctrl-i	Insert one space (tab substitution)
Ctrl-j	Carriage return (executes command)
Ctrl-k	Kill line from cursor to end of line
Ctrl-l	Refresh current line
Ctrl-m	Carriage return (executes command)
Ctrl-n	Next command from history buffer
Ctrl-o	None
Ctrl-p	Previous command from history buffer
Ctrl-q	None
Ctrl-r	Refresh current line
Ctrl-s	None
Ctrl-t	Transpose character under cursor with the character just prior to the cursor
Ctrl-u	Delete line from the beginning of line to cursor
Ctrl-v	None
Ctrl-w	None
Ctrl-x	Move forward one word
Ctrl-y	Paste back what was deleted by the previous Ctrl-k or Ctrl-w command. Text is pasted back at the cursor location
Ctrl-z	If inside a subsystem, it exits back to the top level. If in Enable mode, it exits back to User mode. If in Configure mode, it exits back to Enable mode.
ESC-b	Move backward one word
ESC-d	Kill word from cursor's current location until the first white space.
ESC-f	Move forward one word
ESC-BackSpace	Delete backwards from cursor to the previous space (essentially a delete-word-backward command)
SPACE	Attempts to complete command keyword. If word is not expected to be a keyword, the space character is inserted.
!*	Show all commands currently stored in the history buffer.
!#	Recall a specific history command. '#' is the number of the history command to be recalled as shown via the '!*' command.
"<string>"	Opaque strings may be specified using double quotes. This prevents interpretation of otherwise special CLI characters.

Port Names

The term *port* refers to a physical connector on a line card installed in the X-Pedition router. Figure 7 shows eight 10 Base-T/100 Base-TX ports on a line card.

Figure 7. 10 Base-T/100 Base-TX Ports



In the CLI, each X-Pedition port is referenced with the following syntax:

<*type*>.<*slot-number*>.<*port-number*>

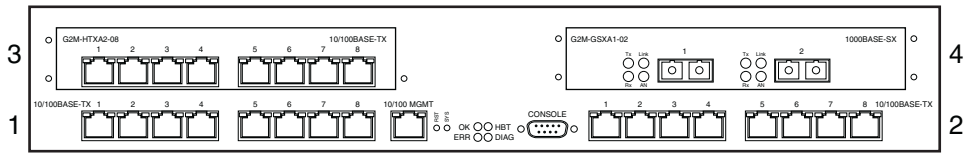
where:

<*type*> is the type of line card that can be one of the following:

Type	Line Card Description
at	Asynchronous Transfer Mode (ATM)
et	10 Base-X or 100 Base-X Ethernet
fi	Fiber Distributed Data Interface (FDDI)
gi	1000 Base-X Gigabit Ethernet
hs	Dual High Speed Serial Interface (HSSI) WAN
se	Serial WAN
so	Packet-over-Sonet (POS)
xg	10 Gigabit Ethernet

<*slot-number*> is determined by the X-Pedition model and the physical slot in which the line card is installed. On the X-Pedition 2000, 2100, and 2400 models, the slot number is printed on the side of each slot, as shown in Figure 8.

Figure 8. Slot Numbers for Models 2000, 2100, and 2400



On the X-Pedition 8000 (Figure 9) and X-Pedition 8600 (Figure 10) models, a legend printed on the fan tray shows the number of each slot. Note that Slot 1 can accommodate either a second control module or a line card. PS1 and PS2 are power supply locations, while SF1 and SF2 are the switch fabric slots.

Figure 9. Slot Numbers for the X-Pedition 8000 Model

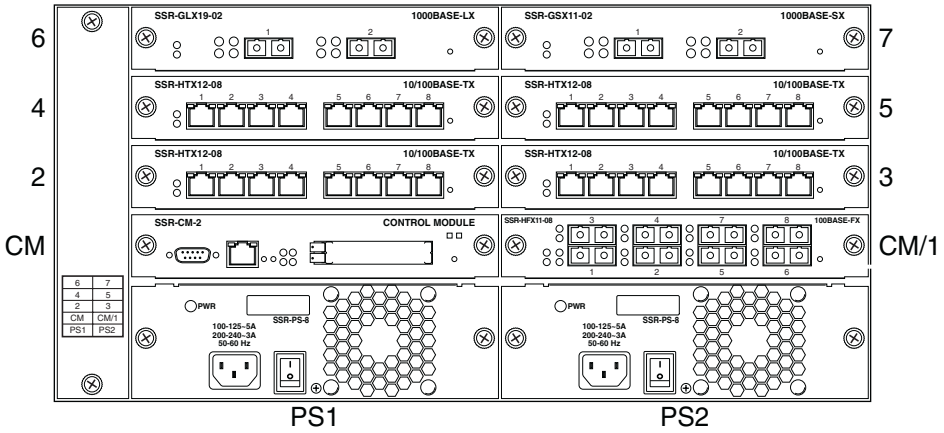
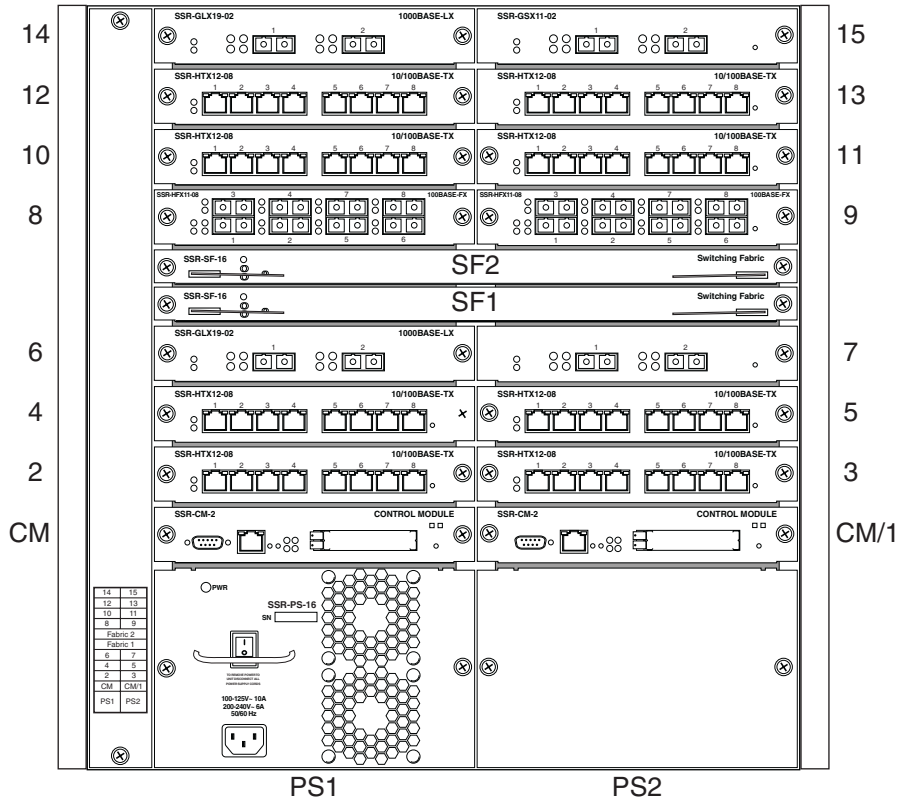
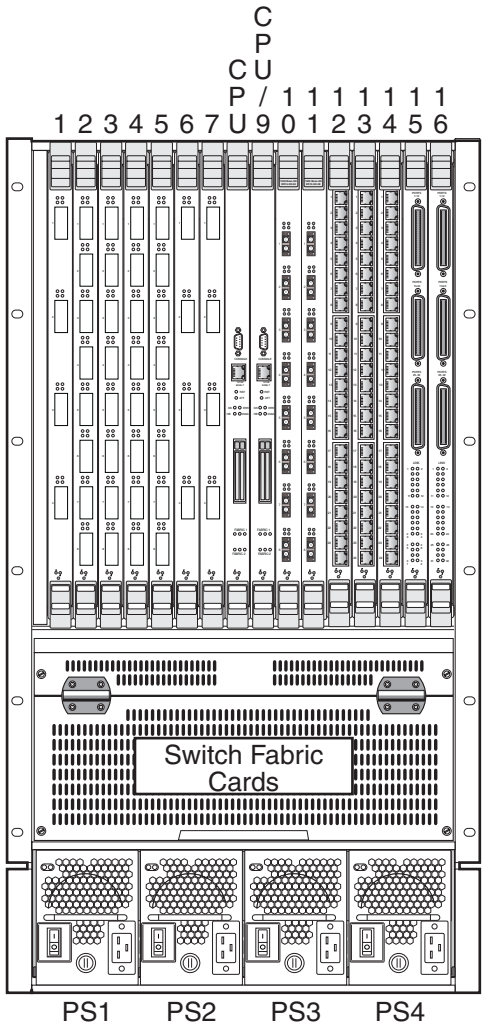


Figure 10. Slot Numbers for the X-Pedition 8600 Mode



Slot numbers for the vertically arranged line cards of the ER16 are printed just above the slot (Figure 11). ER16 slot 9 can be used for either a second control module or a line card.

Figure 11. Slot Numbers for the ER16



<*port-number*> is the number assigned to the connector on the line card. On the ER16, ports are numbered from top to bottom while on all other X-Pedition platforms the ports are numbered from left to right. Port numbers are identified on each line card. For example, the port name et.2.8 refers to the port on the Ethernet line card located in slot 2, connector 8, while port name gi.3.2 refers to the port on the Gigabit Ethernet line card located in slot 3, connector 2.

Ports with Virtual Connections

Certain WAN line cards support virtual channels, which are multiple virtual connections on a single physical connection. Port syntax for cards supporting virtual connections is of two forms: ATM and serial WAN.

ATM Virtual Connections

<type>.<slot-number>.<port-number>.<vpi>.<vci>

where:

<*type*> is the card type, or

Type	Line Card Description
at	Asynchronous Transfer Mode (ATM)

<*slot-number*> and <*port-number*> are the same as defined for the port names.

<*vpi*> is the Virtual Path Identifier, a number that identifies a particular virtual path.

<*vci*> is the Virtual Channel Identifier, a number identifying a particular virtual channel.

Serial WAN Virtual Connections

<type>.<slot-number>.<port-number>.<vc>

where:

<*type*> is the card type, or

Type	Line Card Description
hs	Dual High Speed Serial Interface (HSSI)
se	Serial WAN

<*slot-number*> and <*port-number*> are the same as defined for the port names.

<*vc*> is the Virtual Channel, a number identifying a particular virtual channel.

Pseudo Devices

In addition to the line card types, the X-Pedition router also supports two pseudo-device types: Multilink PPP and SmartTRUNK. A pseudo-device is a logical entity that uses the ports of a physical line card. Multilink PPP and SmartTRUNK are similar in that both features permit users to associate multiple physical links to form a virtual cable bundle. Ports assigned to either a Multilink PPP bundle or a SmartTRUNK inherit all the attributes of the pseudo-device.

Pseudo devices are addressed using the following CLI syntax:

<*pseudo type*>.<*index*>

where:

<*pseudo type*> is the type of pseudo-device which can be one of the following:

Type	Pseudo-Device Description
mp	Multilink PPP
st	SmartTRUNK

<*index*> is a user assignable value to uniquely identify the port.

CLI port-list Syntax

The CLI port-list capability permits users to specify a single port or multiple ports with a single expression. The following operators support multi-port expressions:

- Comma
- Dash
- Wildcard

The comma operator permits users to specify a list of slots or ports separated by commas. The following examples illustrate its usage.

Referenced Ports	Comma Operator Expression
Ports 1, 5 and 6 of an Ethernet line card in slot 2	et.2.(1,5,6)
Port 1 on the Ethernet line cards in slots 1, 3 and 4	et.(1,3,4).1
Port 2 on the Ethernet line card in slot 1 and port 5 on an Ethernet card in slot 4.	et.1.2,et.4.5

The dash operator specifies a range of slots or ports as illustrated by the following examples.

Referenced Ports	Dash Operator Expression
Ports 1 through 6 of an Ethernet line card in slot 2	et.2.(1-6)
Port 1 on the Ethernet line cards in slots 2 through 4	et.(2-4).1

The wildcard operator (*) allows all the ports in a slot or all the slots in the platform to be specified as shown below.

Referenced Ports	Wildcard Operator Expression
All ports on the Ethernet line card in slot 2	et.2.*
Port 1 on all the Ethernet line cards	et.*.1
All Ethernet ports in the platform	et.*.*

The comma, dash and wildcard operators can be used in combination to create powerful expressions, as shown in the following examples.

Referenced Ports	Mixed Operator Expression
Ports et.1.1 through et.1.8, et.2.1 through et.2.8, and et.3.1 through et.3.8	et.(1-3).(1-8)
Ports et.1.1 through et.1.8, and et.3.1 through et.3.8	et.(1,3).(1-8)
Ports et.1.1, et.1.8, et.2.1, et.2.8, et.3.1, et.3.8	et.(1-3).(1,8)
Ports et3.1, et3.3, et.3.5, et.3.6 and et.3.7	et.3.(1,3, 5-7)
All the ports in slots 1, 3 and 4	et.(1,3-4).*

Although command dependent, in general, the comma, dash and wildcard operators can be used when specifying the following:

- SmartTRUNK index
- Multilink PPP index
- ATM vpi
- ATM vci
- Serial or HSSI vc

The port-list commands allow the port list to be compressed to create a single command with a port list from multiple commands, or expanded to create separate CLI commands for each port.

Chapter 5

Logging

Introduction

The X-Pedition router uses a series of system messages to track router activity and status. These messages display helpful information to inform users of simple changes in operational status or warn users of more severe issues that may affect system operations. The areas to which these messages apply depends on the facilities used and the user-defined status type of the messages displayed.

Facility Support

A *facility* refers to a specific functional subset on the router that employs error messages, statements, and commands to support a unique protocol or feature (e.g., ATM, BGP, VLAN, SNMP, and so on). The X-Pedition router supports logging for the following facilities:

ACL	Access Control List
ACL_LOG	Access Control List Log
AGGRGEN	Aggregated/Generated Root
ARE	Advanced Routing Engine
ARP	Address Resolution Protocol
ATALK	Apple-Talk
ATM	Asynchronous Transfer Mode
ATM_DIAG	Asynchronous Transfer Mode Diagnostics
AUTH	Authentication
BGP	Border Gateway Protocol
CDP	Cabletron/Cisco Discovery Protocol
CLI	Command Line Interface

COMMON	Common Command Line Interface
CONFIG	Configuration
CONS	Console
CTRONCHASSIS	Chassis-Related
DDT	Dynamic Disassembly Tool
DHCPD	Dynamic Host Configuration Protocol
DVMRP	Distance Vector Multicast Routing Protocol
ECCMEM	Error Correcting Code Memory
ERR	Error
ETH	10Base-T Ethernet Driver
FDDI	Fiber Distributed Data Interface
GARP	Generic Attribute Registration Protocol
GATED	Gate Daemon
GVRP	GARP VLAN Registration Protocol
HBT	Control Module Heartbeat
IGMP	Internet Group Membership Protocol
INTERFACE	Interface
IP	IP Stack
IPC	The IPC facility used by WAN
IPHELPER	The IP Helper and BOOTP/DHCP Relay Agent
IPRED	IP Redundancy (VRRP)
IPV6	IP Stack Version 6
IPX	Internet Packet Exchange
L2TM	Layer-2 Table Manager
L3AGE	Layer-3 Aging facility
LOADBAL	Load Balance
MIRRORING	Mirroring
MSTP	Multiple Spanning Tree Protocol
MULTICAST	Multicast
NAT	Network Address Translation
NDISC	Neighbor Discovery
NETFLOW	NetFlow
NETSTAT	Netstat
NI	Network Interface Driver
NTP	Network Time Protocol

OSPF	Open Shortest Path First
PBR	IP Policy
PHY_POLL	Phy_Poll
PIM	Protocol Independent Multicast
PING	Ping
PING6	Ping Version 6
POLICY	Policy
PPP	Point-to-Point Protocol
PROFILE	Profile
PTY	Pseudo TTY
QOS	Quality of Service
RARPD	Reverse Address Resolution Protocol
RCP	Remote Copy Protocol
RDISC	Router Discovery
RELAY	Relay
RES	Resolver
RIP	Routing Information Protocol
RIPNG	Routing Information Protocol Next Generation
RL	Rate Limit
RMON	Remote Network Monitoring
RTADVD	Router Advertisement Daemon
SAM	Security Attack Monitor
SIO	Serial Input/Output
SNMP	Simple Network Management Protocol
SONET	Packet-Over-Sonet
SR	Temperature-Related Messages
SSH	Secure Shell
STATIC	Static Address
STATS	Statistics
STP	Spanning Tree Protocol
STRNK	SmartTRUNK
SYS	System
SYSLOG	Syslog
T1T3CLI	T1/T3 Configuration Commands
TELNETD	Telnet

TFTP	Trivial File Transfer Protocol
TR	Traceroute
TRACEROUTE6	Traceroute for IPv6 packets
UNICAST	Unicast
VLAN	Virtual Local Area Network
WAN	Wide Area Network
WC	Web Cache

Reading Messages

Every system message the X-Pedition router supports follows the same basic format:

<*timestamp*>**%**<*facility*>**-**<*severity*>**-**<*description*> <*message text*>

where:

- <*timestamp*> lists the date and time when the message appeared (e.g., 2002-08-23 14:32:00).
- <*facility*> is a code consisting of uppercase letters that indicates the facility to which the error message refers (e.g., VLAN, SNMP, WAN).
- <*severity*> is a single-letter code used to indicate the severity of the error condition. There are five severity levels:

  I – informational message (least severe)

  A – audit message

  W – warning condition

  E – error condition

  F – fatal error (most severe)

Note: Informational messages do not require user intervention.

- <*description*> is a code consisting of uppercase letters that identifies the error message.
- <*message text*> is a brief description of the error condition.

The following example shows a set of informational messages:

```
2002-10-22 13:08:51 %SYS-I-FLASHCRD, Mounting 8MB Flash card
2002-10-22 13:08:51 %SYS-I-FLASHMNTD, 8MB Flash card mounted
2002-10-22 13:08:59 %SYS-I-INITSYS, initializing system XP 8600
2002-10-22 13:08:59 %SYS-I-DSCVMOD, discovered 'Control Module' module in slot CM
2002-10-22 13:09:04 %SYS-I-INITSLOTS, Initializing system slots - please wait
2002-10-22 13:09:11 %SYS-I-MODPROBE, Detecting installed media modules - please wait
```

Logging Methods

The X-Pedition router allows you to monitor and track system activity by generating messages that report various states and events. You may view and discard this information or store it for future reference. To track these messages, the X-Pedition router uses the console and Syslog logging methods. If you are unsure whether or not logging is already enabled or which type is in use, enter the following from enable mode:

```
xp# system show syslog
```

Note: This command currently indicates only whether or not local or remote Syslog logging is enabled. To see if console logging is enabled, search the configuration file for a **system set console level** command.

There are various levels of system messages: Info, Audit, Warning, Error, and Fatal. To view which error message levels are enabled for individual facilities on the router, enter the following from the CLI:

```
xp# system show syslog levels
```

Console Logging

Console logging displays messages to the console only and allows users to view only as many messages as will fit on the screen—as new messages appear, old messages simply scroll off the console. While this is a temporary means of logging this information, it allows administrators to track very specific activities quickly and easily. Console logging is configured through the **system set console level** and **system set syslog-levels** commands and is available only while on the screen or in the screen buffer memory.

To configure console logging, do the following:

1. Set the *global* message level to display to the console. The following example tracks Fatal messages only:

```
xp(config)# system set console level fatal
```

2. To display *additional* message types for a specific facility, you may override the global level. The following example overrides the VLAN default set by the previous command to display Warning, Error, and Fatal messages:

```
xp(config)# system set syslog-levels vlan level warning
```

Note: When a user configures the router to display messages of a lower severity level, the X-Pedition router also displays all messages for that facility that are of higher severity (see following table). To override the global console message severity defined with the **system set syslog** command, use the **system set syslog-levels** command.

Fatal	Log fatal messages only.
Error	Log fatal and error messages only.
Warning	Log fatal, error, and warning messages only.
Audit	Log fatal, error, audit, and warning messages only.
Info	Log all messages.

Syslog Logging

Users may write messages locally to the internal Flash, externally to the Syslog server, or both. However, because the Syslog method does not display output on the console, it may not be readily obvious to users that this method is enabled.

Remote Syslog Server

The Syslog logging method allows users to identify the server to which the X-Pedition router should send system messages. The Syslog logging method is configured through the **system set syslog** and **system set syslog-levels** commands.

To configure Syslog logging to write to an external server, do the following:

1. Set the *global* message level to write to the Syslog server. The following example logs only Fatal messages to a server whose IP address is 10.1.43.77:

```
xp(config)# system set syslog server 10.1.43.77 level fatal
```

2. To display *additional* message types for a specific facility, you may override the global level. The following example overrides the VLAN default set by the previous command to display Warning, Error, and Fatal messages:

```
xp(config)# system set syslog-levels vlan level warning
```

Note: When a user configures the router to display messages of a lower severity level, the X-Pedition router also displays all messages for that facility that are of higher severity (see following table). To override the global console message severity defined with the **system set syslog** command, use the **system set syslog-levels** command.

Fatal	Log fatal messages only.
Error	Log fatal and error messages only.
Warning	Log fatal, error, and warning messages only.
Audit	Log fatal, error, audit, and warning messages only.
Info	Log all messages.

Local Flash

The **local** parameter of the **system set syslog** command logs Syslog messages to a local log file, **int-flash/cfg/syslog**—even if you have not configured a remote Syslog server.

Note: The local Flash is *NOT the PCMCIA card*—it is the X-Pedition router's internal memory.

Each time the router reboots and the Syslog facility initializes, the local Syslog file moves to **int-flash/cfg/syslog.bak** and a new log is created. Local logging is subject to the Syslog filtering mechanism. To display the contents of the local log files, use either of the following:

```
xp# file type syslog
```

```
xp# file type syslog.bak
```

Note: You may still use the **system show syslog buffer** command to display the buffered messages.

The X-Pedition router stores the last <*n*> messages in a local circular buffer. By default, this buffer keeps the last 50 Syslog messages; however, you can change the buffer size to hold 10–200 messages. To view the current buffer size, enter the **system show syslog buffer** command.

To configure Syslog logging to write to the internal Flash, do the following:

1. Set the *global* message level to write to the internal Flash. The following example tracks Fatal messages only:

```
xp(config)# system set syslog local level fatal
```

2. To display *additional* message types for a specific facility, you may override the global level. The following example overrides the VLAN default set by the previous command to display Warning, Error, and Fatal messages:

```
xp(config)# system set syslog-levels vlan level warning
```

Note: When a user configures the router to display messages of a lower severity level, the X-Pedition router also displays all messages for that facility that are of higher severity (see following table). To override the global console message severity defined with the **system set syslog** command, use the **system set syslog-levels** command.

Fatal	Log fatal messages only.
Error	Log fatal and error messages only.
Warning	Log fatal, error, and warning messages only.
Audit	Log fatal, error, audit, and warning messages only.
Info	Log all messages.

Audit Trail

With the X-Pedition router's ability to support multiple user accounts on the same router, it is important to be able to monitor what administrative changes are performed on the system and who performs them. The X-Pedition router collects this information and outputs it to a console, Syslog server, or Flash memory in the form of audit log messages which allow you to track information such as the username, source IP address, and session type. Whenever a user successfully executes a command, the audit trail entry will specify whether the command was added, modified, or removed from the configuration. If the command does not execute properly, the audit trail indicates a failure.

Audit Messages

Audit messages are used as an audit *trail* to aid in keeping the router secure by reporting events as they occur and information about the user who caused each event. When reporting an audit message, the X-Pedition router lists the following information prior to the message text.

<*user session type*> **user(**<*user name*>**:** <*from IP address*>**)**

Note: Values for user session type are Console, Telnet, SSH, and SNMP.

Example:

Telnet user(fred:192.168.189.52)

On the Syslog server, you can decide what to do with these messages based on their levels and the facility with which they are associated (i.e., discard the messages or write them to file). When writing this information to file, the Syslog logging method allows you to identify the server to which the X-Pedition router should send the messages. Please note that audit messages are a subset of all system messages and are available only for a limited subset of facilities.

Facility	Description
Console	Login
Telnet	Telnet Login
VLAN	VLAN Creation/Modification
ACL	ACL Creation/Modification/Application
SYS	Port Manipulation User Creation/Modification Password Manipulation
SNMP	SNMP Modifications
Syslog	Syslog Modification
SSH	Enable/Disable SSH
Configuration	Configuration File Changes
ACL_LOG	ACL denied access attempts

The X-Pedition router stores the last <*n*> messages in a local *circular buffer*. The circular buffer is a location in system memory allocated by the heap to store system messages before sending them to the Syslog server or Flash—messages remain in memory until the buffer reaches the maximum buffer size and begins to replace old messages with new ones. To view the current contents of the buffer, use the **system show syslog buffer** command. By default, this buffer keeps the last 50 Syslog messages; however, you can change the buffer size to hold 10–200 messages.

The following example depicts a sample audit trail. Note that screen output is not limited to audit messages.

```
2002-10-23 14:10:04  %SYS-A-CLI_MODE_CHANGE, Telnet user (root:134.141.135.129), CLI mode changed to (configuration).
2002-10-23 14:09:59  %CLI-A-COMMAND_EXEC, Telnet user (root:134.141.135.129), CLI command (enable) executed
2002-10-23 14:09:59  %SYS-A-CLI_MODE_CHANGE, Telnet user (root:134.141.135.129), CLI mode changed to (enabled).
2002-10-23 14:09:52  %TELNETD-A-LOGIN, Telnet user (user:134.141.135.129), Telnet user login
2002-10-23 14:09:52  %SYS-A-CLI_MODE_CHANGE, Telnet user (user:134.141.135.129), CLI mode changed to (guest).
2002-10-23 14:09:52  %SYS-W-NOPASSWD, no password for login, use 'system set password' in Config mode
2002-10-23 14:09:11  %CLI-A-COMMAND_EXEC, Console user (root), CLI command (system show syslog buffer number 20 ) executed
2002-10-23 14:08:56  %SYS-A-CLI_MODE_CHANGE, Console user (root), CLI mode changed to (enabled).
2002-10-23 14:08:33  %SNMP-I-AGENT_READY, SNMP agent is now ready to send and receive packets
2002-10-23 14:08:32  %SYS-A-CLI_MODE_CHANGE, Console user (root), CLI mode changed to (configuration).
2002-10-23 14:08:27  %CLI-A-COMMAND_EXEC, Console user (root), CLI command (enable ) executed
2002-10-23 14:08:27  %SYS-A-CLI_MODE_CHANGE, Console user (root), CLI mode changed to (enabled).
2002-10-23 14:08:20  %CONS-A-LOGIN, Console user (user), Console user login
2002-10-23 14:08:20  %SYS-A-CLI_MODE_CHANGE, Console user (user), CLI mode changed to (guest).
2002-10-23 14:08:20  %SYS-W-NOPASSWD, no password for login, use 'system set password' in Config mode
2002-10-23 14:08:14  %ACL_LOG-A-DENY, ACL [ixia] on "all-IP" UDP 10.10.1.2:63 -> 11.11.1.2:63
2002-10-23 14:08:14  %ACL_LOG-A-DENY, ACL [ixia] on "all-IP" UDP 10.10.1.2:63 -> 11.11.1.2:63
2002-10-23 14:08:14  %ACL_LOG-A-DENY, ACL [ixia] on "all-IP" UDP 10.10.1.2:63 -> 11.11.1.2:63
2002-10-23 14:08:14  %ACL_LOG-A-DENY, ACL [ixia] on "all-IP" UDP 10.10.1.2:63 -> 11.11.1.2:63
2002-10-23 14:08:14  %ACL_LOG-A-DENY, ACL [ixia] on "all-IP" UDP 10.10.1.2:63 -> 11.11.1.2:63
```

Set Up an Audit Trail on the Console

To configure an Audit Trail for **all facilities** and log to the console, do the following:

- Set the *global* message level to display to the console to Audit. This displays Audit, Warning, Error, and Fatal messages:

```
xp(config)# system set console level audit
```

Note: To override the global console message severity, reset the severity using the same command.

Set Up an Audit Trail on a Syslog Server

To configure an Audit Trail for **a specific facility** only and log to an external Syslog server, do the following:

- Set the *global* message level to write to the Syslog server. Select **Fatal** to prevent the router from logging any message that does not present a fatal condition. This will help limit the number of messages collected and make it easier to view the activity within a specific area. The following example tracks Audit messages for the System facility:

```
xp(config)# system set syslog server 10.1.43.77 level fatal
xp(config)# system set syslog-levels sys audit
```

The audit trail for this example might appear as follows:

```
2002-10-23 14:10:04  %SYS-A-CLI_MODE_CHANGE, Telnet user (root:134.141.135.129), CLI mode changed to
(configuration).
2002-10-23 14:09:59  %SYS-A-CLI_MODE_CHANGE, Telnet user (root:134.141.135.129), CLI mode changed to
(enabled).
2002-10-23 14:09:52  %SYS-A-CLI_MODE_CHANGE, Telnet user (user:134.141.135.129), CLI mode changed to
(guest).
2002-10-23 14:09:52  %SYS-W-NOPASSWD, no password for login, use 'system set password' in Config mode
2002-10-23 14:08:56  %SYS-A-CLI_MODE_CHANGE, Console user (root), CLI mode changed to (enabled).
2002-10-23 14:08:32  %SYS-A-CLI_MODE_CHANGE, Console user (root), CLI mode changed to (configuration).
2002-10-23 14:08:27  %SYS-A-CLI_MODE_CHANGE, Console user (root), CLI mode changed to (enabled).
2002-10-23 14:08:20  %SYS-A-CLI_MODE_CHANGE, Console user (user), CLI mode changed to (guest).
2002-10-23 14:08:20  %SYS-W-NOPASSWD, no password for login, use 'system set password' in Config mode
```

To configure an Audit Trail for **all facilities** and log to an external Syslog server, do the following:

- Set the *global* message level to write to the Syslog server. The following example tracks Audit, Warning, Error, and Fatal messages:

```
xp(config)# system set syslog server 10.1.43.77 level audit
```

Note: To override the global Syslog message severity defined with the **system set syslog** command, use the **system set syslog-levels** command.

Log Audit Trail to Local Flash

The **local** parameter of the **system set syslog** command logs Syslog messages to a local log file, **int-flash/cfg/syslog**—even if you have not configured a remote Syslog server.

Note: The local Flash is *NOT the PCMCIA card*—it is the X-Pedition router's internal memory.

Each time the router reboots and the Syslog facility initializes, the local Syslog file moves to **int-flash/cfg/syslog.bak** and a new log is created. Local logging is subject to the Syslog filtering mechanism. To display the contents of the local log files use either of the following:

```
xp# file type syslog
```

```
xp# file type syslog.bak
```

Note: You may still use the **system show syslog buffer** command to display the buffered messages.

The X-Pedition router stores the last <*n*> messages in a local circular buffer. By default, this buffer keeps the last 50 Syslog messages; however, you can change the buffer size to hold 10–200 messages. To view the current buffer size, enter the **system show syslog buffer** command.

To configure an Audit Trail for **all facilities** and log to the internal Flash, do the following:

- Set the *global* message level to write to the internal Flash. The following example tracks Audit, Warning, Error, and Fatal messages:

```
xp(config)# system set syslog local level audit
```

Note: When a user configures the router to display messages of a lower severity level, the X-Pedition router also displays all messages for that facility that are of higher severity (see following table). To override the global Syslog message severity defined with the **system set syslog** command, use the **system set syslog-levels** command.

Fatal	Log fatal messages only.
Error	Log fatal and error messages only.
Warning	Log fatal, error, and warning messages only.
Audit	Log fatal, error, audit, and warning messages only.
Info	Log all messages.

To configure an Audit Trail for **a specific facility** only, set the global level to Warning and override the level only for the facility you wish to monitor.

- The following example overrides the VLAN default set by the previous command to display Audit, Warning, Error, and Fatal messages:

```
xp(config)# system set syslog-levels vlan level audit
```

ACL Logging

The **report-denied [all | periodic]** option of the **acl apply interface** and **acl apply port** commands allows enhanced detection of messages denied access by ACL rules. This aids network testing, diagnosis, and security monitoring. By default, the router logs only the first ACL match and records a hardware flow entry—subsequent messages that match the ACL are dropped.

Note: This functionality is currently available for IP ACL's only.

Note: You may not apply ACLs to interface EN0 of the control module.

Select the **all** parameter to log every deny ACL match. Under normal system operation, it is not necessary to configure the system to log each occurrence of an ACL match.

Note: Logging each occurrence will cause an extra load on the CPU. Use this option for testing or network diagnosis only.

Use **periodic** to periodically log the number of packets dropped by the hardware for the current deny ACL. This option will interrogate hardware flow entries installed for deny ACLs as part of the aging process. When the dropped packet count increases from the value saved from the last interrogation, the router will generate a log entry reporting the value of the increased count.

Chapter 6

SmartTRUNK Configuration Guide

This chapter explains how to configure and monitor SmartTRUNKs on the X-Pedition router. A SmartTRUNK is Enterasys Networks' technology for load balancing and load sharing. A SmartTRUNK is a group of two or more ports that are logically combined into a single port. Multiple physical connections between devices are aggregated into a single logical, high-speed path that acts as a single link. Traffic is balanced across all interfaces in the combined link, increasing overall available system bandwidth. SmartTRUNKs allow administrators the ability to increase bandwidth at congestion points in the network, thus eliminating potential traffic bottlenecks. SmartTRUNKs also provide improved data link resiliency. If one port in a SmartTRUNK should fail, its load is distributed evenly among the remaining ports and the entire SmartTRUNK link remains operational.

Note: SmartTRUNKs may benefit from enabling flow bridging on all receiving ports. This will add both *src* and *dst* fields into the flow distribution.

SmartTRUNK is Enterasys standard for building high-performance links between Enterasys switching platforms. SmartTRUNKs can interoperate with switches, routers, and servers from other vendors as well as Enterasys platforms. SmartTRUNKs are compatible with all X-Pedition features, including VLANs, STP, VRRP, etc. SmartTRUNK operation is supported over different media types and a variety of technologies including 10/100/1000 Mbps Ethernet. SmartTRUNK operation also supports 10 Gigabit-Ethernet with no protocol or 802.3ad only.

Note: When you enable Spanning Tree Protocol (STP) or Multiple Spanning Tree Protocol (MSTP) with a SmartTRUNK that consists of ports of the same type, Enterasys recommends that the ports in the SmartTRUNK be neighbors (i.e., in consecutive order) on the same line card.

Note: If you configure a SmartTRUNK to carry traffic for an IP VLAN and traffic for the Layer-2 default VLAN, Enterasys recommends that you configure the SmartTRUNK as an 802.1Q trunk. With the trunk configured, disable and re-enable the SmartTRUNK.

Configuring SmartTRUNKs

To configure a SmartTRUNK, perform the following tasks:

1. Create a SmartTRUNK and specify a control protocol for it.
2. Add physical ports to the SmartTRUNK.
3. Specify the policy for distributing traffic across SmartTRUNK ports (optional). By default, the X-Pedition router distributes traffic to ports in a round-robin (sequential) manner.

Note: The maximum number of SmartTRUNKs you may configure depends on the router you are using:

X-Pedition 2000 router allows up to 40
X-Pedition 8000 router allows up to 64
X-Pedition 8600 router router allows up to 128
ER-16 allows up to 256

Note: When using firmware version E9.1.0.0 or later, the ER-16 supports SmartTRUNKs that use the lower index ranges of 1-240 only (i.e., st.1–st.240).

Creating a SmartTRUNK

When you create a SmartTRUNK, you must specify if you will use the DEC Hunt Group control protocol, LACP, or if you will not use any control protocol:

- If you connect the SmartTRUNK to another X-Pedition router, other Enterasys devices such as the SmartSwitch 6000 or SmartSwitch 9000, or Digital GIGAswitch/Router, specify the DEC Hunt Group control protocol. The Hunt Group protocol is useful in detecting errors like transmit/receive failures and misconfiguration.

  Note: When SmartTRUNKing to an Ethernet GIGASwitch product, set the **port set** command's IFG parameter to 4 or greater to allow the GIGASwitch enough time to properly process incoming frames.

- If you connect the SmartTRUNK to a device that does not support the DEC Hunt Group control protocol, such as those devices that support Cisco's EtherChannel technology, specify no control protocol. This mode detects only link failures.

- If you configure the SmartTRUNK for 802.3ad link aggregation, specify the Link Aggregation Control Protocol (LACP). You will then be able to use the **smarttrunk lacp** command to set the various actor parameters for this protocol.

To create a SmartTRUNK, enter the following command in Configure mode:

Create a SmartTRUNK that will connect to a device that supports the DEC Hunt Group control protocol.	**smarttrunk create** <*smarttrunk*> **protocol huntgroup**
Create a SmartTRUNK that will connect to a device that does not support the DEC Hunt Group control protocol.	**smarttrunk create** <*smarttrunk*> **protocol no-protocol**
Create a SmartTRUNK to use for 802.3ad link aggregation.	**smarttrunk create** <*smarttrunk*> **protocol lacp**

Note: SmartTRUNK links may be affected when the Huntgroup protocol is enabled on SmartTRUNKs while the Control Module is too busy to send or receive Huntgroup PDUs.

Add Physical Ports to the SmartTRUNK

A SmartTRUNK may include any port on any module and may include up to the total number of ports on the X-Pedition router. If one module should go down, the ports on other modules will remain operational.

Note: The Huntgroup protocol supports up to 256 ports. In the X-Pedition 8600 router, Huntgroup is not supported on the 16th port of 16-port cards inserted into slot 15. SmartTRUNK without Huntgroup protocol is supported for all modules. On the ER-16, you may configure SmartTRUNKs that use the Huntgroup control protocol on slots 1-7 only.

Ports added to a SmartTRUNK must:

- Be set to full duplex.
- Be in the same VLAN.
- Share the same properties (L2 aging, STP state, and so on).

Note: Do not use the **add ports** command to add ports to an LACP SmartTRUNK. When you use the **lacp actor-parameters** command to enable LACP, the X-Pedition router adds ports to the SmartTRUNK dynamically.

Note: SmartTRUNK links may be affected when the Huntgroup protocol is enabled on SmartTRUNKs while the Control Module is too busy to send or receive Huntgroup PDUs.

Hot Swapping

When hot swapping any card, the link for any SmartTRUNK configured with Huntgroup protocol will go down for a few seconds. If you hot swap a card that has SmartTRUNK and Layer-2 filters configured, the X-Pedition router will mark SmartTRUNK commands with an “E”.

Specify Traffic Distribution Policy (Optional)

The default policy for distributing traffic across the ports in a SmartTRUNK is “round-robin,” where the X-Pedition router selects the port on a rotating basis. The other policy that can be chosen is “link-utilization,” where packets are sent to the least-used port in a SmartTRUNK. You can choose to specify the link-utilization policy for a particular SmartTRUNK, a list of SmartTRUNKs, or for all SmartTRUNKs on the X-Pedition router.

Specify traffic distribution policy.	**smarttrunk set load-policy on** <***smarttrunk*** *list*> | **all-smarttrunks round-robin| link-utilization**

Specify Actor Parameters for LACP (Link Aggregation Control Protocol)

To enable LACP and set the actor parameters on a port to run Link Aggregation with the 802.3ad protocol, do the following:

Enable LACP.	**smarttrunk lacp actor-parameters port** <*port_list*> **enable**
Disable LACP.	Negate the command you used to enable LACP.
Set the administrative key for the port.	**smarttrunk lacp actor-parameters port** <*port_list*> **enable port-key** <*number*>
Set the administrative priority for the port.	**smarttrunk lacp actor-parameters port** <*port_list*> **enable port-priority** <*number*>
Set the administrative LACP activity for the port.	**smarttrunk lacp actor-parameters enable port** <*port_list*> **activity active| passive**
Set the administrative aggregation for the port.	**smarttrunk lacp actor-parameters enable port** <*port_list*> **aggregation aggregatable| individual**
Set the administrative LACP Timeout for the port.	**smarttrunk lacp actor-parameters enable port** <*port_list*> **timeout long| short**

Monitoring SmartTRUNKs

Statistics are gathered for data flowing through a SmartTRUNK and each port in the SmartTRUNK. To display SmartTRUNK statistics, enter one of the following commands in Enable mode:

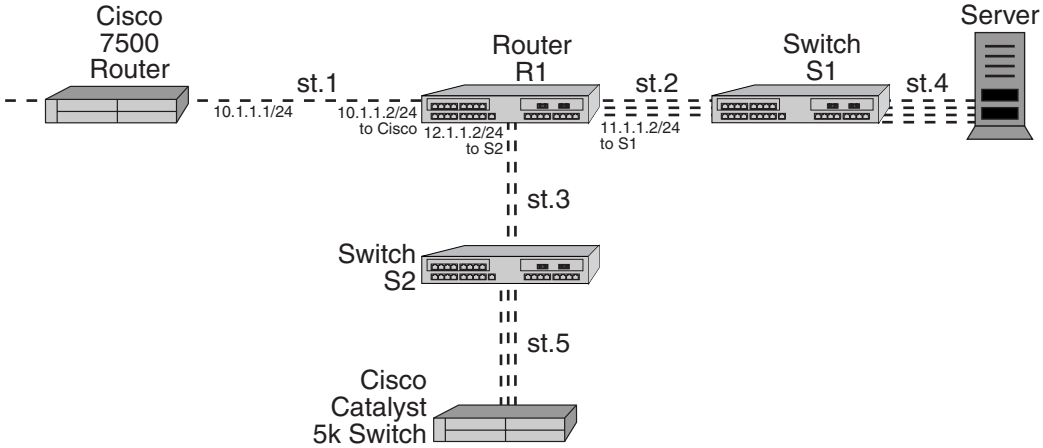
Display information about all SmartTRUNKS and the control protocol used.	**smarttrunk show trunks**
Display statistics on traffic distribution statistics on SmartTRUNK.	**smarttrunk show distribution** <*smarttrunk list*> | **all-smarttrunk**s
Display information about the control protocol on a SmartTRUNK.	**smarttrunk show protocol-state** <*smarttrunk list*>| **all-smarttrunk**s
Display information about the SmartTRUNK connection (DEC Hunt Group control protocol connections only).	**smarttrunk show connections** <*smarttrunk list*>| **all-smarttrunk**s
Display the LACP port parameters for this SmartTRUNK.	**smarttrunk show lacp-control** <*smarttrunk list*>| **all-smarttrunk**s
Display the LACP statistics counters.	**smarttrunk show lacp-stats** <*smarttrunk list*>| **all-smarttrunk**s
Display the link-aggregation groups and their members.	**smarttrunk show lags** <*smarttrunk list*>| **all-smarttrunk**s

To clear statistics for SmartTRUNK ports, enter the following command in Enable mode:

Clear load distribution statistics for SmartTRUNK ports.	**smarttrunk clear load-distribution** <*smarttrunk list*>| **all-smarttrunks**

Example Configurations

The following shows a network design based on SmartTRUNKs. R1 is an X-Pedition platform operating as a router, while S1 and S2 are X-Pedition platforms operating as switches.



The following is the configuration for the Cisco 7500 router:

```
interface port-channel 1
ip address 10.1.1.1 255.255.255.0
ip route-cache distributed
interface fasteth 0/0
no ip address
channel-group 1
```

The following is the configuration for the Cisco Catalyst 5K switch:

```
set port channel 3/1-2 on
```

The following is the SmartTRUNK configuration for the X-Pedition router labeled ‘R1’ in the diagram:

```
smarttrunk create st.1 protocol no-protocol
smarttrunk create st.2 protocol huntgroup
smarttrunk create st.3 protocol huntgroup
smarttrunk add ports et.1(1-2) to st.1
smarttrunk add ports et.2(1-2) to st.2
smarttrunk add ports et.3(1-2) to st.3
interface create ip to-cisco address-netmask 10.1.1.2/24 port st.1
interface create ip to-s1 address-netmask 11.1.1.2/24 port st.2
interface create ip to-s2 address-netmask 12.1.1.2/24 port st.3
```

The following is the SmartTRUNK configuration for the X-Pedition router labeled ‘S1’ in the diagram:

```
smarttrunk create st.2 protocol huntgroup
smarttrunk create st.4 protocol no-protocol
smarttrunk add ports et.1(1-2) to st.2
smarttrunk add ports et.2(1-2) to st.4
```

The following is the SmartTRUNK configuration for the X-Pedition router labeled ‘S2’ in the diagram:

```
smarttrunk create st.3 protocol huntgroup
smarttrunk create st.5 protocol no-protocol
smarttrunk add ports et.1(1-2) to st.3
smarttrunk add ports et.2(1-2) to st.5
```

Configuring the Link Aggregation Control Protocol (LACP)

When you configure Enterasys Networks’ SmartTRUNK to support the 802.3ad Link Aggregation Control Protocol (LACP), the SmartTRUNK is treated as an *aggregator*. The aggregator presents a standard IEEE 802.3 service interface and communicates with the MAC client. The aggregator binds to one or more ports and is responsible for distributing frames from a MAC client to its attached ports, and for collecting received frames from the ports and passing them to the MAC client transparently.

You can enable LACP on all 10/100 and Gigabit Ethernet ports on the X-Pedition router. LACP ports exchange LACP PDUs with their peers and from one or more Link Aggregation Groups (LAGs). After joining a LAG, the port attaches to an appropriate aggregator or SmartTRUNK.

Note: Ports on which you enable LACP are devoted solely to LACP.

Configuring SmartTRUNKs for LACP

To configure a SmartTRUNK and specify the LACP protocol, perform the following tasks:

1. Enter the following command in Configure mode to create a SmartTRUNK and specify the LACP protocol.

Create a SmartTRUNK that will run LACP	**smarttrunk create** <smarttrunk> **protocol lacp**

Note: The maximum number of SmartTRUNKs you may configure depends on the router you are using:

X-Pedition 2000 router allows up to 40
X-Pedition 8000 router allows up to 64
X-Pedition 8600 router allows up to 128
ER-16 allows up to 256

2. Enter the following command in Configure mode to enable the LACP protocol on ports.

Enable LACP on 10/100 and Gigabit Ethernet Ports	**smarttrunk lacp actor-parameters port** <*port_list*> **enable** [**port-key** <*number*>] [**port-priority** <*number*>] [**activity active**	**passive**] [**aggregation aggregatable**	**individual**] [**timeout long**	**short**]

In addition to turning on LACP, you can set the following parameters:

- The **port-key** is used to associate a port with an aggregator. With an associated port and aggregator, the port key is the same as the aggregator actor key.
- The **activity** parameter allows you to specify how to transmit LACP PDUs. **Active** allows the port to transmit LACP PDUs regardless of its partner's control value. **Passive** allows the port to transmit LACP PDUs only if the partner's control value is *active*.
- If a port is able to join a Link Aggregation Group (LAG), specify **aggregatable**. Otherwise, specify **individual**.
- The **timeout** parameter controls the rate at which LACP PDUs transmit.

3. Enter the following command in Configure mode to configure the aggregator's properties.

Configure the aggregator's properties.	**smarttrunk lacp aggregator** <*smarttrunk*> **port-type 10-100-Ethernet**| **Gigabit-Ethernet** **actor-key** <*number*>| **default-10-100**| **default-gig** [**system-priority** <*number*>]

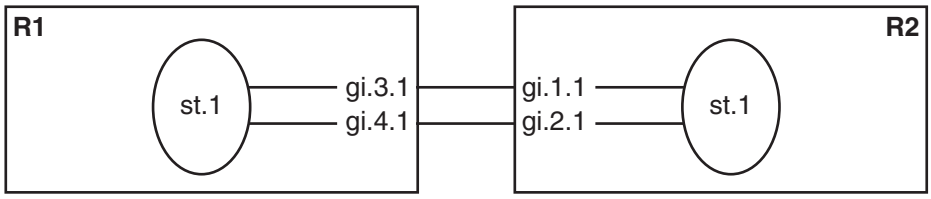
The following parameters are required. For additional information about the LACP Set command, see *Enterasys X-Pedition Command Line Interface Reference Manual*.

– Use the **aggregator** parameter to identify the SmartTRUNK you will configure.

– Defining the **port-type** parameter specifies whether the ports associated with the aggregator are 10/100 Ethernet ports or Gigabit Ethernet ports.

LACP Configuration Example

On R1 and R2 below, SmartTRUNK st.1 was configured with the LACP protocol specified. On R1, st.1 is the aggregator and ports gi.3.1 and gi.4.1 bind to it. On R2, st.1 is the aggregator and ports gi.1.1 and gi.2.1 bind to it.



The configuration for R1 is as follows:

```
xp# smarttrunk create st.1 protocol lacp
smarttrunk lacp actor-parameters port gi.3.1 enable port-key 10
smarttrunk lacp actor-parameters port gi.4.1 enable port-key 10
smarttrunk lacp aggregator st.1 port-type Gigabit-Ethernet actor-key 10
```

The configuration for R2 is as follows:

```
xp# smarttrunk create st.1 protocol lacp
smarttrunk lacp actor-parameters port gi.1.1 enable port-key 20
smarttrunk lacp actor-parameters port gi.2.1 enable port-key 20
smarttrunk lacp aggregator st.1 port-type Gigabit-Ethernet actor-key 20
```

Chapter 7

Bridging Configuration Guide

Bridging Overview

The Enterasys X-Pedition router provides the following bridging functions:

- Compliance with the IGMP multicast bridging standard
- Wire-speed address-based bridging or flow-based bridging
- Ability to logically segment a transparently bridged network into Virtual Local-Area Networks (VLANs), based on physical ports or protocol (IP or IPX or bridged protocols like Appletalk)
- Frame filtering based on MAC address for bridged and multicast traffic
- Integrated routing and bridging, which supports bridging of intra-VLAN traffic and routing of inter-VLAN traffic

Bridging Modes (Flow-Based and Address-Based)

The X-Pedition router provides the following types of wire-speed bridging:

Address-based bridging - The X-Pedition router performs this type of bridging by looking up the destination address in an L2 lookup table on the line card that receives the bridge packet from the network. The L2 lookup table indicates the exit port(s) for the bridged packet. If the packet is addressed to the X-Pedition router's own MAC address, the packet is routed rather than bridged.

Flow-based bridging - The X-Pedition router performs this type of bridging by looking up an entry in the L2 lookup table containing both the source and destination addresses of the received packet in order to determine how the packet is to be handled.

The X-Pedition ports perform address-based bridging by default but can be configured to perform flow-based bridging instead, on a per-port basis. A port cannot be configured to perform both types of bridging at the same time.

The X-Pedition router performance is equivalent when performing flow-based bridging or address-based bridging. However, address-based bridging is more efficient because it requires fewer table entries while flow-based bridging provides tighter management and control over bridged traffic.

VLAN Overview

Virtual LANs (VLANs) are a means of dividing a physical network into several logical (virtual) LANs. The division can be done on the basis of various criteria, giving rise to different types of VLANs. For example, the simplest type of VLAN is the port-based VLAN. Port-based VLANs divide a network into a number of VLANs by assigning a VLAN to each port of a switching device. Then, any traffic received on a given port of a switch *belongs* to the VLAN associated with that port.

VLANs are primarily used for broadcast containment. A layer-2 (L2) broadcast frame is normally transmitted all over a bridged network. By dividing the network into VLANs, the *range* of a broadcast is limited, i.e., the broadcast frame is transmitted only to the VLAN to which it belongs. This reduces the broadcast traffic on a network by an appreciable factor.

Note: If you configure a SmartTRUNK to carry traffic for an IP VLAN and traffic for the Layer-2 default VLAN, you must configure the SmartTRUNK as an 802.1Q trunk. With the trunk configured, disable and re-enable the SmartTRUNK.

Types of VLANs

The type of VLAN depends upon one criterion: how a received frame is classified as belonging to a particular VLAN. VLANs can be categorized into the following types:

- Port based
- MAC address based
- Protocol based
- Subnet based
- Multicast based
- Policy based

Detailed information about these types of VLANs is beyond the scope of this manual. Each type of VLAN is briefly explained in the following subsections.

Port-based VLANs

Ports of L2 devices (switches, bridges) are assigned to VLANs. Any traffic received by a port is classified as belonging to the VLAN to which the port belongs. For example, if ports 1, 2, and 3 belong to the VLAN named “Marketing,” then a broadcast frame received by port 1 is transmitted on ports 2 and 3. It is not transmitted on any other port.

Note: Layer-4 bridging is allowed on IP or IPX VLANs only.

MAC-address-based VLANs

In this type of VLAN, each switch (or a central VLAN information server) keeps track of all MAC addresses in a network and maps them to VLANs based on information configured by the network administrator. When a frame is received at a port, its destination MAC address is looked up in the VLAN database. The VLAN database returns the name of the VLAN to which this frame belongs.

This type of VLAN is powerful in the sense that network devices such as printers and workstations can be moved anywhere in the network without the need for network reconfiguration. However, the administration is intensive because all MAC addresses on the network need to be known and configured.

Protocol-based VLANs

Protocol-based VLANs divide the physical network into logical VLANs based on protocol. When a frame is received at a port, its VLAN is determined by the protocol of the packet. For example, there could be separate VLANs for IP, IPv6, IPX and Appletalk. An IP broadcast frame will only be sent to all ports in the IP VLAN.

Subnet-based VLANs

Subnet-based VLANs are a subset of protocol based VLANs and determine the VLAN of a frame based on the subnet to which the frame belongs. To do this, the switch must look into the network layer header of the incoming frame. This type of VLAN behaves similar to a router by segregating different subnets into different broadcast domains.

Multicast-based VLANs

Multicast-based VLANs are created dynamically for multicast groups. Typically, each multicast group corresponds to a different VLAN. This ensures that multicast frames are received only by those ports that are connected to members of the appropriate multicast group.

Policy-based VLANs

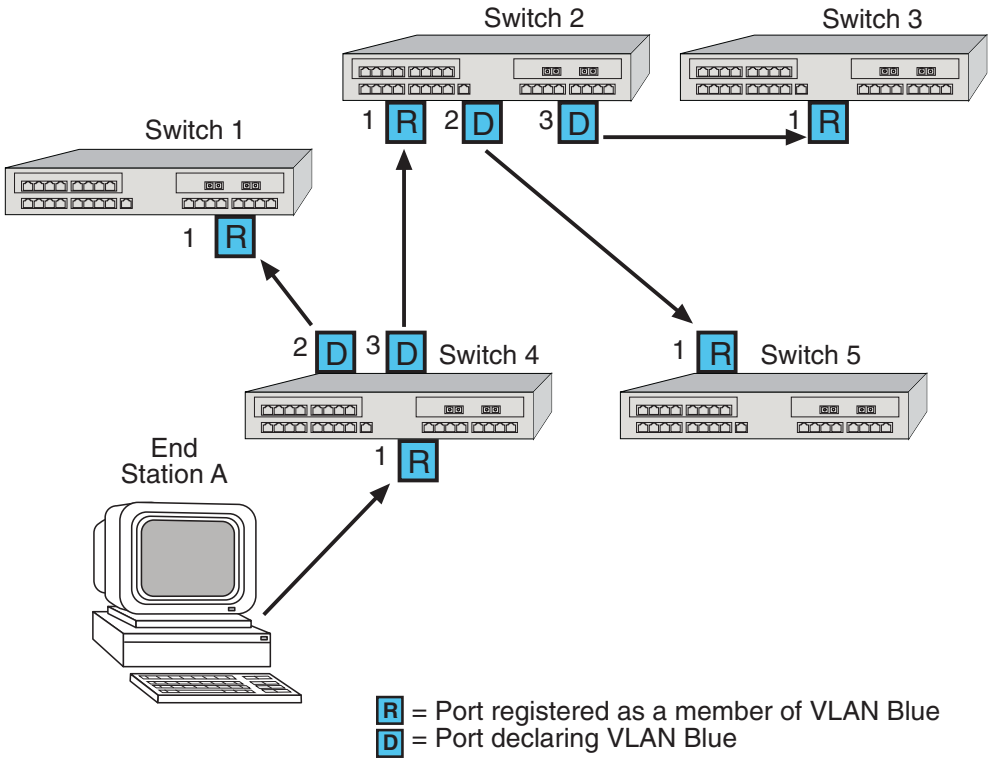
Policy-based VLANs are the most general definition of VLANs. Each incoming (untagged) frame is looked up in a policy database, which determines the VLAN to which the frame belongs. For example, you could set up a policy which creates a special VLAN for all E-mail traffic between the management officers of a company, so that this traffic will not be seen anywhere else.

GVRP

The purpose of the GARP VLAN Registration Protocol (GVRP) is to dynamically create VLANs across a switched network. When a VLAN is declared, the information is transmitted out GVRP configured ports on the device in a GARP formatted frame using the GVRP multicast MAC address. A switch/router that receives this frame, examines the frame, and extracts the VLAN IDs. GVRP then creates the VLANs and adds the receiving port to its tagged member list for the extracted VLAN ID (s). The information is then transmitted out the other GVRP configured ports of the device.

Figure 12 shows an example of how VLAN Blue from end station A would be propagated across a switch/router network. In this figure, port 1 of Device 4 is registered as being a member of VLAN Blue and then declares this fact out all its ports (2 and 3) to Device 1 and Device 2. These two devices register this in the port egress lists of the ports (Device 1, port 1 and Device 2, port 1) that received the frames with the information. Device 2, which is connected to Device 3 and Device 5 declares the same information to those two devices and the port egress list of each port is updated with the new information, accordingly.

Figure 12. Example of VLAN Propagation via GVRP



X-Pedition Router VLAN Support

The X-Pedition router supports:

- Port-based VLANs
- Protocol-based VLANs
- Subnet-based VLANs

When using the X-Pedition router as an L2 bridge/switch, use the port-based and protocol-based VLAN types. When using the X-Pedition router as a combined switch and router, use the subnet-based VLANs in addition to port-based and protocol-based VLANs. It is not necessary to remember the types of VLANs in order to configure the X-Pedition router, as seen in the section on configuring the X-Pedition router.

Note: Layer-4 bridging is allowed on IP or IPX VLANs only.

VLANs and the X-Pedition Router

VLANs are an integral part of the X-Pedition family of switching routers. The X-Pedition switching routers can function as layer-2 (L2) switches as well as fully-functional layer-3 (L3) routers. Hence they can be viewed as a switch and a router in one box. To provide maximum performance and functionality, the L2 and L3 aspects of the X-Pedition switching routers are tightly coupled.

The X-Pedition router can be used purely as an L2 switch. Frames arriving at any port are bridged and not routed. In this case, setting up VLANs and associating ports with VLANs is all that is required. You can set up the X-Pedition switching router to use port-based VLANs, protocol-based VLANs, or a mixture of the two types.

The X-Pedition router can also be used purely as a router, i.e., each physical port of the X-Pedition router is a separate routing interface. Packets received at any interface are routed and not bridged. In this case, no VLAN configuration is required. Note that VLANs are still created implicitly by the X-Pedition router as a result of creating L3 interfaces for IP and/or IPX. However, these implicit VLANs do not need to be created or configured manually. The implicit VLANs created by the X-Pedition router are subnet-based VLANs.

Most commonly, an X-Pedition router is used as a combined switch and router. For example, it may be connected to two subnets S1 and S2. Ports 1-8 belong to S1 and ports 9-16 belong to S2. The required behavior of the X-Pedition router is that intra-subnet frames be bridged and inter-subnet packets be routed. In other words, traffic between two workstations that belong to the same subnet should be bridged, and traffic between two workstations that belong to different subnets should be routed.

The X-Pedition switching routers use VLANs to achieve this behavior. This means that a L3 subnet (i.e., an IP or IPX subnet) is mapped to a VLAN. A given subnet maps to exactly one and only one VLAN. With this definition, the terms *VLAN* and *subnet* are almost interchangeable.

To configure an X-Pedition router as a combined switch and router, the administrator must create VLANs whenever multiple ports of the X-Pedition router are to belong to a particular VLAN/subnet. Then the VLAN must be *bound to* an L3 (IP/IPX) interface so that the X-Pedition router knows which VLAN maps to which IP/IPX subnet.

Ports, VLANs, and L3 Interfaces

The term *port* refers to a physical connector on the X-Pedition router, such as an ethernet port. Each port must belong to at least one VLAN. When the X-Pedition router is unconfigured, each port belongs to a VLAN called the “default VLAN.” By creating VLANs and adding ports to the created VLANs, the ports are moved from the default VLAN to the newly created VLANs.

Unlike traditional routers, the X-Pedition router has the concept of logical interfaces rather than physical interfaces. An L3 interface is a logical entity created by the administrator. It can contain more than one physical port. When an L3 interface contains exactly one physical port, it is equivalent to an interface on a traditional router. When an L3 interface contains several ports, it is equivalent to an interface of a traditional router which is connected to a layer-2 device such as a switch or bridge.

Access Ports and Trunk Ports (802.1P and 802.1Q Support)

The ports of an X-Pedition router can be classified into two types, based on VLAN functionality: **access ports** and **trunk ports**. By default, a port is an access port. An access port can belong to at most one VLAN of the following types: IP, IPX, or bridged protocols.

Note: The 802.1Q bridge MIB can create an IP-based VLAN with Layer-4 bridging enabled.

Trunk ports (802.1Q) carry traffic belonging to several VLANs and are usually used to connect one VLAN-aware switch to another. For example, X-Pedition router A and B are both configured with VLANs V1 and V2. A frame arriving at a port on router A must be sent to router B, if the frame belongs to VLAN V1 or to VLAN V2. Thus the ports on routers A and B which connect the two routers together must belong to both VLAN V1 and VLAN V2. Also, when these ports receive a frame, they must be able to determine whether the frame belongs to V1 or to V2. This is accomplished by "tagging" the frames, i.e., by prepending information to the frame in order to identify the VLAN to which the frame belongs. In the X-Pedition switching routers, trunk ports always transmit and receive tagged frames only. The format of the tag is specified by the IEEE 802.1Q standard. The only exception to this is Spanning Tree Protocol frames, which are transmitted as untagged frames.

The X-Pedition router can automatically determine whether a received frame is an IP frame, an IPX frame or neither. Based on this, it selects a VLAN for the frame. Frames transmitted out of an access port contain no special information about the VLAN to which they belong. These frames are classified as belonging to a particular VLAN based on the protocol of the frame and the VLAN configured on the receiving port for that protocol. For example, if port 1 belongs to VLAN *IPX_VLAN* for IPX, VLAN *IP_VLAN* for IP and VLAN *OTHER_VLAN* for any other protocol, then an IP frame received by port 1 is classified as belonging to VLAN *IP_VLAN*.

You can use the **port enable 8021p** command to tag frames transmitted from access ports with a one-byte, 802.1p class of service (CoS) value. The CoS value indicates the frame's priority. There are 8 CoS values, 0 is the lowest priority and 7 is the highest.

Note: A packet entering a Q-trunk has an 802.1Q header containing a priority field. Typically, users can change the 802.1Q priority using the **qos set l2** commands. However, current hardware restrictions ignore any request to overwrite the packet's priority—the packet simply replicates at the exit port and continues with its original priority.

802.1Q functionality permits the operation of VLAN Bridges on WAN and ATM ports. It is possible to configure ATM, PPP, MLPPP, and Frame-Relay ports as trunk ports, add those ports to multiple VLANs, and generate the appropriate tags for packets crossing created links.

Note: Be aware of the following in association with 802.1Q Over WAN:

- The X-Pedition router does not support Per VLAN Spanning Tree (PVST) or Multiple Spanning Tree Protocol (MSTP) over WAN. Therefore, Q-trunks over redundant WAN connections are not recommended.
- MLP bundles are considered a single logical connection for Spanning Tree Protocol (STP) applications.
- ATM connections do not support STP.

By default, trunk ports forward tagged traffic and access ports forward untagged traffic. The **vlan untagged ports** command for a VLAN replaces the default behavior, and the ports identified by the command forward untagged traffic. The ports that are members of the VLAN but not identified by the **vlan untagged ports** command forward tagged traffic.

The **vlan forbid ports** command configures the port to not be a member of a specific VLAN and, therefore, not forward any packets with the specific VLAN ID.

Default VLANs

When a user adds a trunk port to a VLAN with the **vlan add ports** command, the router sets the default VLAN to VLAN 1 (also called the DEFAULT VLAN). Untagged ingress traffic to this port is assigned to the blackhole VLAN (VLAN 4095), where user traffic is dropped and control traffic, such as BPDU and CDP, directed to the router is processed. The default VLAN can be changed to another VLAN for trunk ports only.

After changing the default VLAN on a trunk port, the port is no longer associated with the previous default VLAN. For example when the default VLAN is changed from VLAN 1 to VLAN 2, all ingress traffic tagged with the VLAN ID 1 will be dropped. Untagged ingress traffic will now go to the VLAN 2 in this example. The default VLAN only affects the handling of untagged traffic on ingress. Note that the **vlan untagged ports** command can configure the trunk port to forward untagged egress traffic.

Explicit and Implicit VLANs

VLANs can either be created explicitly by the administrator (explicit VLANs) or are created implicitly by the X-Pedition router when L3 interfaces are created (implicit VLANs).

IPv6 and VLANs

The X-Pedition router supports bridging of IPv6 traffic within a user-configured protocol-based VLAN. IPv6 frames received on a port belonging to a user-configured IPv6 VLAN will be bridged to other ports that are part of the same VLAN. However, the X-Pedition router does not allow an IPv6 routing interface to be configured over a user-configured IPV6 VLAN. Therefore IPv6 traffic cannot be routed to or from a user-configured IPv6 VLAN.

When creating an IPv6 routing interface over a port, the router automatically creates an implicit IPv6 VLAN with that port as its sole member. The IPv6 routing interface then operates over this VLAN. It is not meaningful to talk about bridging within this VLAN, since it has only one member port. However, IPv6 traffic may be routed to and from this automatically created VLAN, since it has an IPv6 routing interface associated with it.

This automatically created VLAN is assigned a VLAN ID internally. However, you can override the internally assigned VLAN ID by explicitly specifying the ID when creating the IPv6 interface. Normally, you would only need to specify a VLAN ID for IPv6 interfaces created on Q-trunks. IPv6 interfaces (configured on different routers) that are connected via a Q-trunk need to share a common VLAN ID to achieve connectivity.

Configuring X-Pedition Bridging Functions

The following sections describe how to configure the bridge functions.

Configuring Address-Based or Flow-Based Bridging

The X-Pedition ports perform address-based bridging by default but can be configured to perform flow-based bridging instead of address-based bridging, on a per-port basis. A port cannot be configured to perform both types of bridging at the same time.

The X-Pedition router performance is equivalent when performing flow-based bridging or address-based bridging. However, address-based bridging is more efficient because it requires fewer table entries while flow-based bridging provides tighter management and control over bridged traffic.

For example, an X-Pedition router has traffic being sent from port A to port B, port B to port A, port B to port C, and port A to port C. The corresponding bridge tables for address-based and flow-based bridging are shown below. The bridge table contains more information on the traffic patterns when flow-based bridging is enabled compared to address-based bridging.

Address-Based Bridge Table	Flow-Based Bridge Table
A (source)	A → B
B (source)	B → A
C (destination)	B → C
	A → C

With the X-Pedition router configured in flow-based bridging mode, the network manager has “per flow” control of layer-2 traffic. The network manager can then apply Quality of Service (QoS) policies or security filters based on layer-2 traffic flows.

To enable flow-based bridging on a port, enter the following command in Configure mode.

Configure a port for flow-based bridging.	**port flow-bridging** <*port-list*>|**all-ports**

To change a port from flow-based bridging to address-based bridging, enter the following command in Configure mode:

Change a port from flow-based bridging to address-based bridging.	**negate** <*line-number of active config containing command*>**: port flow-bridging** <*port-list*>|**all-ports**

Configuring a Port- or Protocol-Based VLAN

To create a port or protocol based VLAN, perform the following steps.

1. Create a port or protocol based VLAN by entering the following command in Configure mode.

Create a VLAN.	**vlan create** <*vlan-name*> <*type*> **id** <*num*>

2. Add physical ports to a VLAN by entering the following command in Configure mode.

Add ports to a VLAN.	**vlan add ports** <*port-list*> **to** <*vlan-name*>

Note: Use the **port-list** commands as a shorthand mechanism for specifying affected ports. The **port-list compress** command allows multiple commands, each with a separate port list, to be compressed into a single command with an expanded port list.

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Configuring VLAN Trunk Ports

The X-Pedition router supports standards-based VLAN trunking between multiple routers as defined by IEEE 802.1Q. 802.1Q adds a header to a standard Ethernet frame which includes a unique VLAN ID per trunk between two routers. These VLAN IDs extend the VLAN broadcast domain to more than one router. To configure a VLAN trunk, enter the following command in the Configure mode.

Configure 802.1Q VLAN trunks.	**vlan make trunk-port** <*port-list*>

Note: The 802.1q bridge MIB can create an IP-based VLAN with Layer-4 bridging enabled.

By default, the default VLAN (VLAN ID 1) is always assigned to all VLAN trunks, and untagged traffic is assigned to the blackhole VLAN (VLAN 4095), where user traffic is dropped and control traffic, such as BPDU and CDP, directed to the router is processed. To change the default VLAN, enter the following command in the Configure mode:

Change the default VLAN.	**vlan default-vlan ports** <*port-list*> **on** <*vlan-name*>

To configure the trunk port to forward untagged traffic, enter the following command in the Configure mode. This command only applies to egress traffic.

Change the trunk port to forward untagged traffic.	**vlan untagged ports** <*port-list*> **on** <*vlan-name*>

802.1Q functionality also permits the operation of VLAN Bridges on WAN and ATM ports. It is possible to configure ATM, PPP, MLPPP, and Frame-Relay ports as trunk ports, add those ports to multiple VLANs, and generate the appropriate tags for packets crossing created links. Be aware of the following in association with 802.1Q Over WAN:

- The X-Pedition router does not support Per VLAN Spanning Tree (PVST) or Multiple Spanning Tree Protocol (MSTP) over WAN. Therefore, Q-trunks over redundant WAN connections are not recommended.
- MLP bundles are considered a single logical connection for Spanning Tree Protocol (STP) applications.
- ATM connections do not support STP.

Configuring VLANs for Bridging

The X-Pedition router allows you to create VLANs for AppleTalk, DECnet, SNA, and IPv6 traffic as well as for IP and IPX traffic. You can create a VLAN for handling traffic for a single protocol, such as a DECnet VLAN. Or, you can create a VLAN that supports several specific protocols, such as SNA and IP traffic.

Note: Some commands in this facility require updated X-Pedition hardware.

Configuring Layer-2 Filters

Layer-2 security filters on the X-Pedition router allow you to configure ports to filter specific MAC addresses. When defining a Layer-2 security filter, you specify to which ports you want the filter to apply. Refer to the *"Security Configuration Chapter"* for details on configuring Layer-2 filters. You can specify the following security filters:

- Address filters

  These filters block traffic based on the frame's source MAC address, destination MAC address, or both source and destination MAC addresses in flow bridging mode. Address filters are always configured and applied to the input port.

- Port-to-address lock filters

  These filters prohibit a user connected to a locked port or set of ports from using another port.

- Static entry filters

  These filters allow or force traffic to go to a set of destination ports based on a frame's source MAC address, destination MAC address, or both source and destination MAC addresses in flow bridging mode. Static entries are always configured and applied at the input port.

- Secure port filters

  A secure filter shuts down access to the X-Pedition router based on MAC addresses. All packets received by a port are dropped. When combined with static entries, however, these filters can be used to drop all received traffic but allow some frames to go through.

Monitoring Bridging

The X-Pedition router provides display of bridging statistics and configurations contained in the router.

To display bridging information, enter the following commands in Enable mode.

Show IP routing table.	**ip show routes**
Show all MAC addresses currently in the l2 tables.	**l2-tables show all-macs**
Show l2 table information on a specific port.	**l2-tables show port-macs**
Show information the master MAC table.	**l2-tables show mac-table-stats**
Show information on a specific MAC address.	**l2-tables show mac**
Show information on MACs registered.	**l2-table show bridge-management**
Show all VLANs.	**vlan show**
Show STP bridge information	**stp show bridging-info**
Show PVST bridge information	**pvst show bridging-info spanning-tree** *<string>*
Show MSTP bridge information	**mstp show bridging**

Configuration Examples

VLANs are used to associate physical ports on the X-Pedition router with connected hosts that may be physically separated but need to participate in the same broadcast domain. To associate ports to a VLAN, you must first create a VLAN and then assign ports to the VLAN. This section shows examples of creating an IP or IPX VLAN and a DECnet, SNA, and AppleTalk VLAN.

Creating an IP or IPX VLAN

In this example, servers connected to port gi.1.(1-2) on the X-Pedition router need to communicate with clients connected to et.4.(1-8). You can associate all the ports containing the clients and servers to an IP VLAN called ‘BLUE’.

Note: If you configure a SmartTRUNK to carry traffic for an IP VLAN and traffic for the Layer-2 default VLAN, you must configure the SmartTRUNK as an 802.1Q trunk. With the trunk configured, disable and re-enable the SmartTRUNK.

First, create an IP VLAN named ‘BLUE’:

```
xp(config)# vlan create BLUE ip
```

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Next, assign ports to the ‘BLUE’ VLAN:

```
xp(config)# vlan add ports et.4.(1-8),gi.1.(1-2) to BLUE
```

Creating a Non-IP/Non-IPX VLAN

In this example, SNA, DECnet, and AppleTalk hosts are connected to et.1.1 and et.2.(1-4). You can associate all the ports containing these hosts to a VLAN called ‘RED’ with the VLAN ID 5.

First, create a VLAN named ‘RED’:

```
xp(config)# vlan create RED sna dec appletalk id 5
```

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

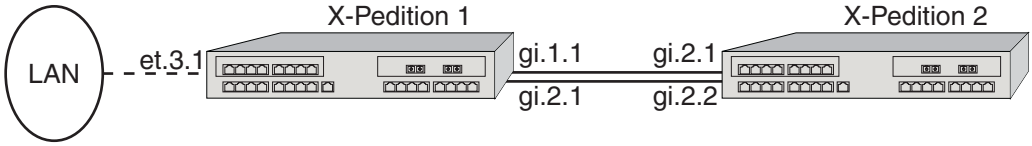
Next, assign ports to the ‘RED’ VLAN:

```
xp(config)# vlan add ports et.1.1, et.2.(1-4) to RED
```

Configuring Per-VLAN Spanning Tree with RSTP Enabled

In this example, two X-Pedition routers are connected as shown in Figure 13. We will create a per VLAN spanning tree instance with RSTP enabled on port gi.1.(1-2), et.3.1 in X-Pedition 1 (XP1) and gi.2.(1-2) in X-Pedition 2 (XP2).

Figure 13. Configuring Per VLAN Spanning Tree with RSTP Enabled



Note: The maximum number of per-VLAN spanning trees you can support relates to the amount of memory installed in your system. Because each spanning tree allocates 1 MB of memory, a system with 256 MB of memory will support approximately 200 per-VLAN spanning trees (depending on the amount of memory needed to configure the hardware).

First, create a VLAN named ‘GREEN’ with VLAN ID 100 on both XP1 and XP2 and add the ports to the VLAN:

```
xp1(config)# vlan create GREEN port-based id 100
xp1(config)# vlan add ports gi.1.(1-2),et.3.1 to GREEN
```

```
xp2(config)# vlan create GREEN port-based id 100
xp2(config)# vlan add ports gi.2.(1-2) to GREEN
```

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Then, create a spanning tree instance for VLAN ‘GREEN’.

```
xp1(config)# pvst create spanningtree vlan_name GREEN
```

```
xp2(config)# pvst create spanningtree vlan_name GREEN
```

Enable PVST on gi.1.1(1-2),et.3.1 in XP1 and gi.2.(1-2) in XP2.

```
xp1(config)# pvst enable port gi.1.(1-2),et.3.1 spanning-tree GREEN
```

```
xp2(config)# pvst enable port gi.2.(1-2) spanning-tree GREEN
```

To enable RSTP for this spanning tree, set the protocol version to ‘rstp’:

```
xp1(config)# pvst set protocol-version rstp spanning-tree GREEN
```

```
xp2(config)# pvst set protocol-version rstp spanning-tree GREEN
```

To allow 1-second link reconfiguration, the links between gi.1.(1-2) in XP1 and gi.2.(1-2) in XP2 must be point-to-point links. You should also define et.3.1 in XP1 as an edge-port if it is not connected to any other bridges that can generate loops in the network.

```
xp1(config)# pvst set port gi.1.(1-2) point-to-point ForceTrue
xp1(config)# pvst set port et.3.1 edge-port True
```

```
xp2(config)# pvst set port gi.2.(1-2) point-to-point ForceTrue
```

Chapter 8

Spanning Tree Configuration Guide

The Enterasys X-Pedition router supports the Spanning Tree Protocol (STP), Rapid Spanning Tree Protocol (RSTP), Per-VLAN Spanning Tree (PVST), and Multiple Spanning Tree Protocol (MSTP). These protocols are defined in the following standards, except for PVST which is a proprietary protocol.

- IEEE 802.1D (Spanning Tree Protocol)
- IEEE 802.1w (Rapid Spanning Tree Protocol)
- IEEE 802.1s (Multiple Spanning Tree Protocol)

MSTP cannot run concurrently with STP, RSTP, or PVST on the same router. These protocols must be disabled on the router, and their commands commented out in the configuration file, before using MSTP. Note that MSTP can be configured to behave like STP or RSTP.

Overview of the Spanning Tree Protocols

The following sections describe the spanning tree protocols supported by the X-Pedition router.

Spanning Tree (IEEE 802.1D)

The Spanning Tree Protocol (STP) defined in IEEE 802.1D allows bridges to dynamically discover a subset of the topology that is loop-free. The loop-free tree that is discovered contains paths to every LAN segment.

The Spanning Tree Protocol is used to eliminate data loops in an Ethernet network by creating a tree where there is only one data route between any two end stations. STP blocks redundant data paths. Should a path become unreachable, STP automatically activates a blocked path. Should a bridge be added, creating a redundant path, STP blocks one of the paths. STP can also change data paths based on a cost change.

All bridges that support the spanning tree exchange information using Bridge Protocol Data Unit (BPDU) messages. Using the information exchanged by the BPDUs, STP designates a bridge for each switched LAN segment, and one root bridge for the spanning tree. The root bridge is the logical center of the spanning tree and is used to determine which paths to block and which to open.

A network administrator can determine the topology of the spanning tree by adjusting the bridge priority, port priority, and path cost. The bridge priority assigns the bridge's relative priority compared to other bridges. The port priority assigns the port's priority in relation to the other ports on the same bridge. By default, the port cost is a value assigned to the port based on the speed of the port. The faster the speed, the lower the cost. This helps to determine the quickest path between the root bridge and a specified destination. The segment attached to the root bridge normally has a path cost of zero.

Note: The terms *path cost* and *port cost* are sometimes used interchangeably. Normally, the port cost is the value assigned to a specific port, while path cost, especially in a BPDU, is the sum of the port costs in a path.

Each bridge has a Bridge identification (BID), which is derived from the bridge's MAC address and bridge priority. The bridge with the lowest BID becomes the root bridge.

Each port on a bridge is in one of the following states:

- Blocking: Port actively prevents traffic from using this path.
- Listening: Port waits for protocol information to determine whether to go back to the Blocking State or continue to the Learning State. The port continues to block traffic.
- Learning: Port learns station location information but continues to block traffic.
- Forwarding: Port forwards traffic and continues to learn station location information.
- Disabled: Port is disabled administratively or by failure.

Rapid Spanning Tree (IEEE 802.1w)

The Rapid Spanning Tree Protocol (RSTP) defined in IEEE 802.1w enhances the STP by allowing the network topology to reconverge in a significantly smaller amount of time. This is partially accomplished by being independent of the protocol timer values when configuring the active topology of a LAN.

Additionally, RSTP recognizes edge ports and point-to-point links as having no redundant paths to their end stations, and subsequently is able to transition faster.

Per-VLAN Spanning Tree

By default, all VLANs belong to the default spanning tree. The Enterasys proprietary Per-VLAN Spanning Tree (PSVT) protocol allows you to create a separate instance of the spanning tree and assign a VLAN to it. With PVST, each spanning tree instance only supports one VLAN. Therefore, each VLAN has its own spanning tree instance. Each instance of the spanning tree is configured and managed the same as the spanning tree created by STP or RSTP.

PVST allows a form of load balancing. With a single spanning tree, a port is either forwarding or blocked to all traffic. By creating an instance of the spanning tree for a VLAN, one port can forward data from one VLAN while blocking data from other VLANs. Another port can forward data from a different VLAN. A new instance of the spanning tree is required for each VLAN.

Other vendors may have their own version of PVST. These proprietary protocols are not necessarily compatible.

Multiple Spanning Trees (IEEE 802.1s)

The Multiple Spanning Tree Protocol (MSTP), defined in IEEE 802.1s, expands upon STP and RSTP, and incorporates many of the major functions of PVST. Some of the MSTP features include:

- Backwards compatible with STP and RSTP.
- Can be configured for a type of load balancing across redundant links.
- Creates a single Common and Internal Spanning Tree (CIST) that represents the connectivity of the entire network.
- Users can group any number of devices into individual regions, with each region behaving and presenting itself as a single device to the rest of the network.
- A region can contain multiple instances of the spanning tree, where each instance can support multiple VLANs.
- In addition to using hello time, forward delay, and max age information, MSTP also utilizes the hop count for improved performance.

MSTP is backwards compatible with the IEEE legacy STP and RSTP spanning tree protocols. MSTP can automatically detect the version of spanning tree being used on a LAN and send out the equivalent type of BPDU. In addition, MSTP incorporates a force version feature where the user may force MSTP to behave as STP or RSTP.

The X-Pedition router does not support MSTP over ATM, Packet over SONET (POS), or WAN.

Common and Internal Spanning Tree (CIST)

MSTP uses all STP, RSTP and Multiple Spanning Tree (MST) Region information to create a single Common and Internal Spanning Tree (CIST) that represents the connectivity of the entire network. This is equivalent to the single spanning tree used for STP and RSTP.

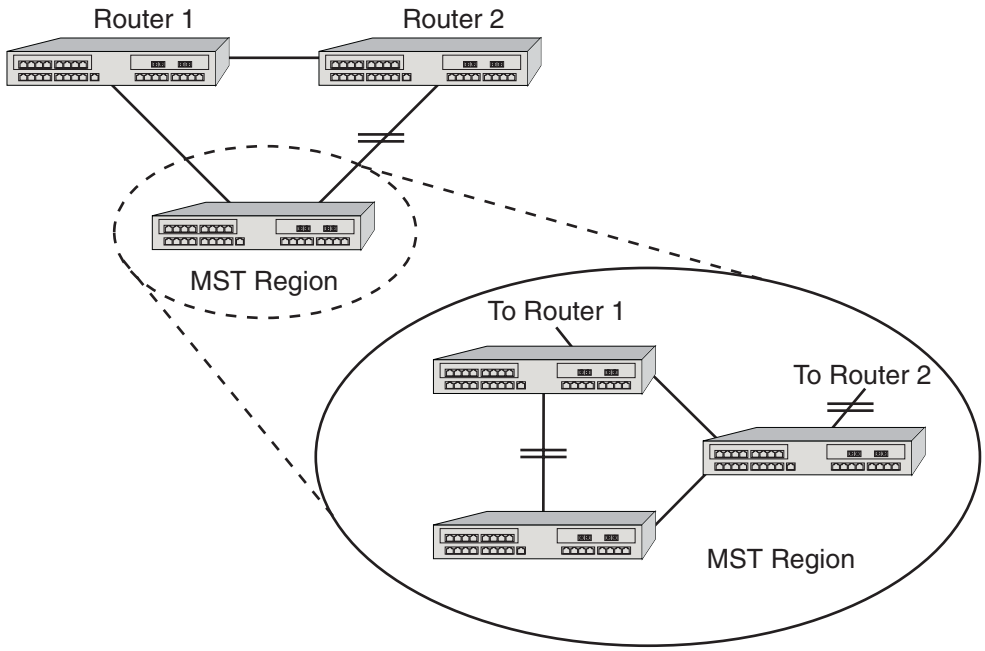
The MSTP enabled network contains one CIST and a minimum of at least one MST Region. A typical network may contain numerous MST Regions as well as separate LAN segments running legacy STP and RSTP spanning tree protocols.

The CIST contains a root bridge, which is the root of the spanning tree for the network. The CIST root is not necessarily located inside an MST region. Each region contains a CIST regional root, unless the CIST root is part of the region.

MST Region

An MST Region is a group of devices that are configured together to form a logical region. The MST Region presents itself to the rest of the network as a single device, which simplifies administration. Path cost is only incremented when entering or leaving region, regardless of the number of devices within the region. Each LAN can only be a member of one region. Figure 14 shows that the MST region appears as a single device to Routers 1 and 2, but really consists of three devices.

Figure 14. MST Region



For a device to be considered as part of an MST region, it must be administratively configured with the same Configuration Identifier information as all other devices in the MST Region. The Configuration Identifier consists of four separate parts:

- Format Selector

  The format selector is one octet in length and is always 0. It cannot be administratively changed.

- Configuration Name

  The Configuration Name is a user assigned case sensitive name given to the region. The maximum length of the name is 32 octets.

- Revision Level

  The Revision Level is two octets in length. The default value of 0 may be administratively changed.

- Configuration Digest

  The Configuration Digest is a 16 octet HMAC-MD5 signature created from the configured VLAN Identification (VID)/Filtering Identification (FID) to Multiple Spanning Tree Instances (MSTI) mappings. All devices must have identical mappings to have identical Configuration Digests.

The MST region designates one CIST regional root bridge for the region, regardless of the number of MSTIs. The regional root provides the connectivity from the region to the CIST root, where the CIST root lies outside the region. For compatibility with STP and RSTP, MSTP increments Message Age at the regional root, unless the regional root is also the CIST root.

Multiple Spanning Tree Instances (MSTI)

Inside the MST Region, a separate topology is maintained from the outside world. Each MST Region may contain up to 64 different MSTIs. The X-Pedition router maps VLAN IDs (VIDs) and Filtering IDs (FIDs) to each other in a one to one correlation; for example, FID 3 = VID 3. VID/FIDs are mapped to different MSTIs to create a type of load balancing. All devices in the region belong to all the MSTIs, which means that an MSTI cannot contain just a subset of devices in the region.

Since an MSTI is a separate spanning tree, each MSTI has its own root inside the MST Region. Figure 15 and Figure 16 show two MSTIs in a single region. Router 3 is the root for MSTI 1, Router 2 is the root for MSTI 2, and Router 5 is the CIST root for the region. Traffic for all the VLANs attached to an MSTI follow the MSTI's spanned topology.

Various options may be configured on a per-MSTI basis to allow for differing topologies between MSTIs. To reduce network complexity and processing power needed to maintain MSTIs, you should only create as many MSTIs as needed.

Figure 15. MSTI 1 in a Region

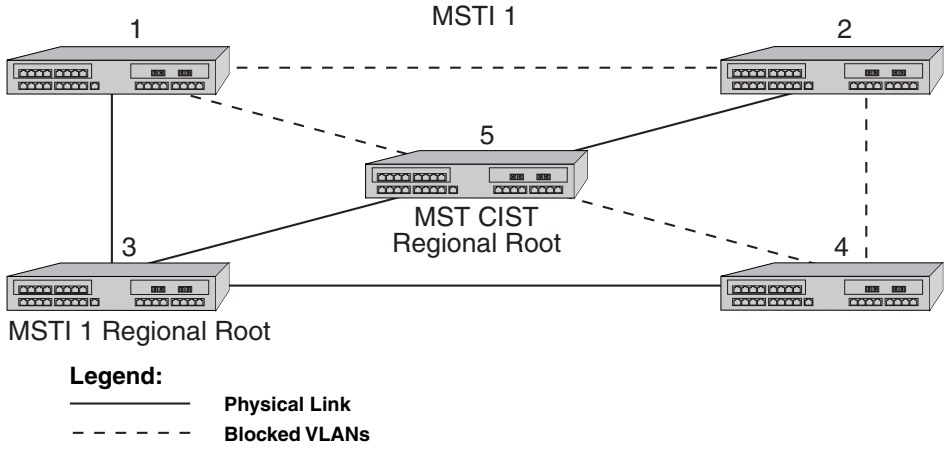
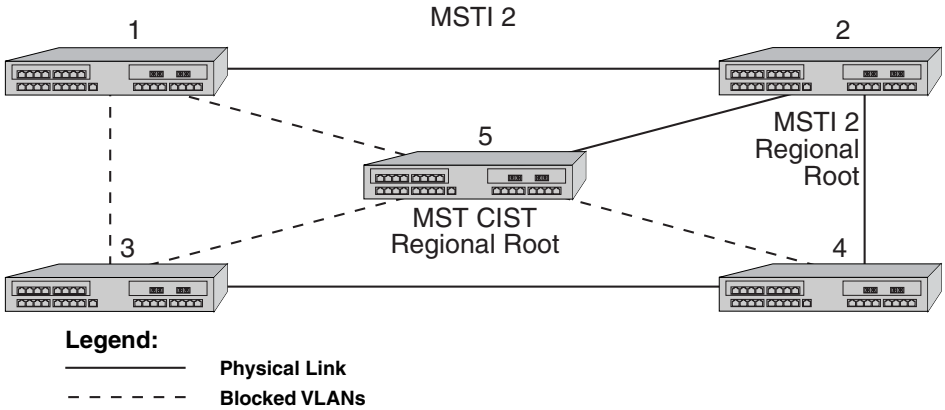


Figure 16. MSTI 2 in the Same Region



MSTIs do not need hello time, forward delay, and max age information. These parameters are used by the CIST when communicating with other regions.

Figure 17 shows 3 regions with five MSTIs. Table 5 defines the characteristics of each MSTI. Ports going to PCs only on routers 1, 3, 9, and 11 can be configured as edge ports. Routers 4 and 10 are the regional roots. Each MSTI can be configured to forward and block various VLANs.

Figure 17. Example of Multiple Regions and MSTIs

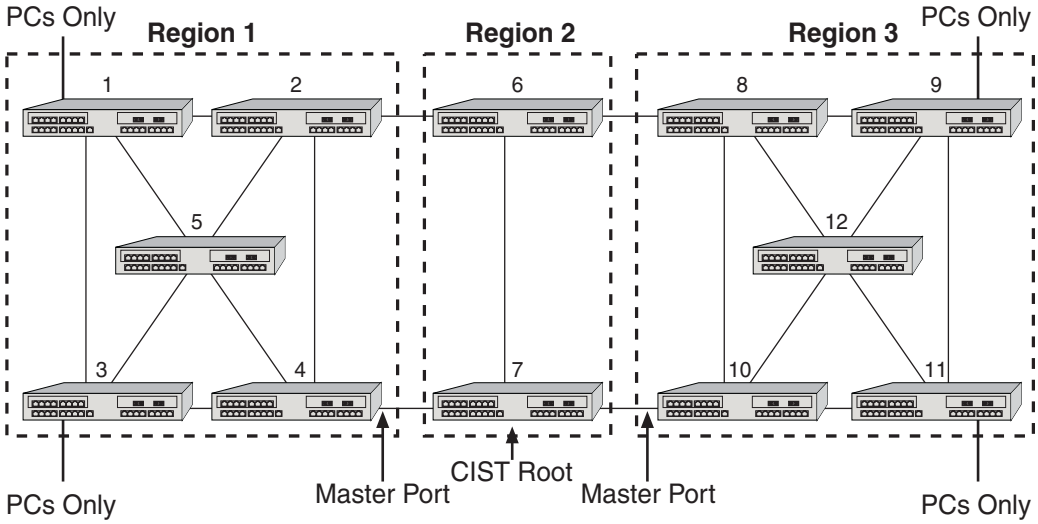


Table 5. MSTI Characteristics

MSTI 1 in Region 1	Root is Router 4, which also the regional root
MSTI 2 in Region 1	Root is Router 5
MSTI 1 in Region 2	Root is Router 7, which is also the CIST root
MSTI 1 in Region 3	Root is Router 11
MSTI 2 in Region 3	Root is Router 12
	Router 10 is the regional root

MSTP Port States and Roles

The MSTP port states are consistent with those defined in RSTP. The three different states an MSTP enabled port can be in are forwarding, learning and discarding.

The six different roles an MSTP enabled port can be in are root, designated, master, alternate, backup and disabled. MSTP port roles are consistent with those defined in RSTP with the addition of the master port role. The master port role only applies to an MST Region. The master port can be thought of as the root port for the entire MST Region. It provides connectivity from the MST Region to a CIST root that lies outside the MST Region. In Figure 17, Routers 4 and 10 are the regional roots and contain the Master Port for their respective regions.

STP, RSTP, and PVST Configuration

The following sections describe how to configure a spanning tree using STP, RSTP, or PVST. These protocols cannot be used on the same router as MSTP.

Configuring Spanning Tree

The X-Pedition router supports per VLAN spanning tree. By default, all the VLANs defined belong to the default spanning tree. You can create a separate instance of spanning tree using the following command:

Create spanning tree for a VLAN.	**pvst create spanningtree vlan-name** <*string*>

Note: The maximum number of per-VLAN spanning trees you can support relates to the amount of memory installed in your system. Because each spanning tree allocates 1 MB of memory, a system with 256 MB of memory will support approximately 200 per-VLAN spanning trees (depending on the amount of memory needed to configure the hardware).

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

By default, spanning tree is disabled on the X-Pedition router. To enable spanning tree, perform the following on the ports where you want spanning tree enabled.

Enable spanning tree on one or more ports for default spanning tree.	**stp enable port** <*port-list*>

To enable a PVST instance, perform the following on the ports where you want spanning tree enabled. The name of the spanning tree instance is the same as the name of the VLAN.

Enable spanning tree on one or more ports for a particular VLAN.	**pvst enable port** <*port-list*> **spanning-tree** <*string*>

Enabling Rapid Spanning Tree Protocol (RSTP)

By default, all spanning trees in the X-Pedition router run in STP (802.1D) compatible mode. The time each port needs to reconfigure its connection after a topology change depends on the forward delay set for the spanning tree, which ranges from 4 to 30 seconds. With RSTP enabled, this converging time could be reduced to as little as 1 second in certain cases. You can enable RSTP using the following:

Enable RSTP on the default spanning tree	**stp set protocol-version rstp**

You can enable RSTP on a spanning tree instance using the following command. The name of the spanning tree instance is the same as the name of the VLAN.

Enable RSTP on a particular spanning tree instance	**pvst set protocol-version rstp spanning-tree** <*string*>

To remain compatible with STP, an RSTP-capable bridge sends out STP-compatible BPDUs to individual ports that are connected to other legacy bridges running STP, and continues to do so should you replace the legacy bridges with RSTP-capable bridges. Additionally, changes in the point-to-point or the edge-port operational status can go undetected by the X-Pedition router. In these cases, the X-Pedition router will not be able to take full advantage of RSTP and will need to be reset. The reset command forces the ports to send RSTP BPDUs until a version 0 STP BPDU is received.

To reset RSTP on these ports and test for changes on spanning trees and spanning tree instances, enter the following commands in Enabled mode:

Reset RSTP on one or more ports for default spanning tree	**stp reset-rstp port** <*port-list*>
Reset RSTP on one or more ports for a particular spanning tree instance	**pvst reset-rstp port** <*port-list*> **spanning-tree** <*string*>

Adjusting Spanning-Tree Parameters

You may need to adjust certain spanning-tree parameters if the default values are not suitable for your bridge configuration. Parameters affecting the entire spanning tree are configured with variations of the bridge global configuration command. Interface-specific parameters are configured with variations of the bridge-group interface configuration command.

You can adjust spanning-tree parameters by performing any of the tasks in the following sections, such as setting the bridge or port priority.

Note: Poorly chosen adjustments to these parameters can have a negative impact on performance. A good source on bridging is the IEEE 802.1D specification.

Setting the Bridge Priority

You can globally configure the priority of an individual bridge when two bridges tie for position as the root bridge, or you can configure the likelihood that a bridge will be selected as the root bridge. The lower the bridge's priority, the more likely the bridge will be selected as the root bridge. This priority is determined by default; however, you can change it.

To set the bridge priority, enter the following command in Configure mode:

Set the bridge priority for default spanning tree.	**stp set bridging priority** <*num*>
Set the bridge priority for a particular instance of spanning tree.	**pvst set bridging spanning-tree** <*string*> **priority** <*num*>

Setting a Port Priority

You can set a priority for an interface. When two bridges tie for position as the root bridge, you configure an interface priority to break the tie. The bridge with the lowest interface value is elected. To set an interface priority, enter the following command in Configure mode:

Establish a priority for a specified interface for default spanning tree.	**stp set port** <*port-list*> **priority** <*num*>
Establish a priority for a specified interface for a particular instance of spanning tree.	**pvst set port** <*port-list*> **spanning-tree** <*string*> **priority** <*num*>

Assigning Port Costs

Each interface has a port cost associated with it. By convention, the higher the port speed, the lower the port cost. By default, the port cost value is based on the port speed using the X-Pedition legacy values, described in the *X-Pedition NATIVE Command Line Interface Reference Manual*. To assign different port costs, enter the following command in Configure mode:

Set a different port cost other than the defaults for default spanning tree.	**stp set port** <*port-list*> **port-cost** <*num*>
Set a different port cost other than the defaults for a particular instance of spanning tree.	**pvst set port** <*port-list*> **spanning-tree** <*string*> **port-cost** <*num*>

Adjusting Bridge Protocol Data Unit (BPDU) Intervals

You can adjust the following BPDU intervals: Interval between Hello BPDUs, Forward Delay Interval, and Maximum Idle Interval.

Adjusting the Interval Between Hello Times

To adjust the interval between hello time, enter the following command in Configure mode:

Specify the interval between hello time for default spanning tree.	**stp set bridging hello-time** <*num*>
Specify the interval between hello time for a particular instance of spanning tree.	**pvst set bridging spanning-tree** <*string*> **hello-time** <*num*>

Defining the Forward Delay Interval

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. To change the default interval setting, enter the following command in Configure mode for the spanning tree or PVST spanning tree instance:

Set the default of the forward delay interval for default spanning tree.	**stp set bridging forward-delay** <*num*>
Set the default of the forward delay interval for a particular instance of spanning tree.	**pvst set bridging spanning-tree** <*string*> **forward-delay** <*num*>

Defining the Maximum Age

If a bridge does not hear BPDUs from the root bridge within a specified interval, it assumes that the network has changed and recomputes the spanning-tree topology. To change the default interval setting, enter the following command in Configure mode for the spanning tree or PVST spanning tree instance:

Change the amount of time a bridge will wait to hear BPDUs from the root bridge for default spanning tree.	**stp set bridging max-age** <*num*>
Change the amount of time a bridge will wait to hear BPDUs from the root bridge for a particular instance of spanning tree.	**pvst set bridging spanning-tree** <*string*> **max-age** <*num*>

Adjusting RSTP Parameters

Since 1-second link reconfiguration can happen only on a point-to-point link or an edge port (a port that is known to be on the edge of a bridged LAN), you may want to define them administratively. You can define the point-to-point link on the RSTP spanning tree or an instance of the PVST spanning tree as shown on the following commands:

Specify a link in the default spanning tree as a point-to-point link [ForceTrue], a non-point-to-point link [ForceFalse], or as determined by the X-Pedition router.	**stp set port** <*port-list*> **point-to-point [ForceTrue|ForceFalse|Auto]**
Specify a link in a particular spanning tree instance as a point-to-point link [ForceTrue], a non-point-to-point link [ForceFalse], or as determined by the X-Pedition router.	**pvst set port** <*port-list*> **point-to-point [ForceTrue|ForceFalse|Auto] spanning-tree** <*string*>

Monitoring Spanning Tree

The X-Pedition router provides display of spanning statistics and configurations contained in the X-Pedition router. To display this information, enter the following commands in Enable mode.

Show STP bridge information	**stp show bridging-info**
Show PVST bridge information	**pvst show bridging-info spanning-tree** <*string*>
Show STP port information	**port show stp-info** <*port_list*>|**all-ports**
Show PVST port information	**port show pvst-info** <*port_list*>|**all-ports spanning-tree** <*string*>

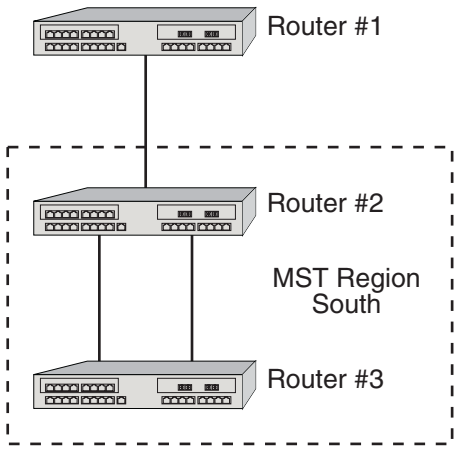
MSTP Configuration

The following sections describe how to configure a spanning tree using MSTP. If the devices were previously configured for STP, RSTP, or PVST, these protocols must be disabled before using MSTP.

Configuring Spanning Tree

The following example describes setting up the simple MSTP network shown in Figure 18. This configuration groups Router #2 and Router #3 into a single MST Region and performs a type of load balancing of traffic across their two connecting ports. By default Router #1 is in its own MST Region. In this example, all the ports are configured as Q-trunks to demonstrate a type of load balancing.

Figure 18. MSTP Sample Network Configuration



The MSTP protocol must be started before configuring any mstp commands.

Start the MSTP protocol	**mstp start**

By default, MSTP in not enabled to run on any port. MSTP must be enabled on all needed or desired ports.

Enable MSTP on one or more ports.	**mstp enable port** *<port-list>*

By default, each router will be in its own MST Region using its own MAC address as the Configuration Name. In this example, Router #2 and Router #3 need to be configured so that they are part of the same MST Region. Set the Configuration Name to “South.”.

Set the Configuration Name portion of the Configuration Identifier.	**mstp set configuration-id name** <*string* | **mstp-null-name**>

We can leave the Revision Level as default 0. The Configuration Digest will automatically be calculated using our mappings of VLANs to MSTIs. Before creating the mappings, we need to create two different MSTIs.

Create a Multiple Spanning Tree Instance (MSTI) for MSTP.	**mstp create msti** <*num*>

By default, all VLAN IDs (VID) are mapped to MSTI 0, which is the CIST. We need to change the mappings for the desired VIDs/FIDs to the appropriate MSTI. Note that when you map a VID to a MSTI, the same FID is also mapped to the MSTI.

Map one or more VLANs to an MSTI for MSTP.	**mstp map** <*num*> **vlan** <*num*>

If the VID to MSTI mappings are identical on Router #2 and Router #3, they will have the same Configuration Digest.

Since we want the traffic between Router #2 and Router #3 to be split across both links, we need to assign different priorities for each of the ports for each MSTI. This will cause the MSTIs to have different topologies.

By default, the port priority for any MSTI is 8 (converted internally to 128). Port priorities can range from 0 to 15 (converted internally 0 to 240 in increments of 16). The lower the priority, the less likely a port will go into a discarding state. For the ports that we want the VLAN traffic to flow over, let’s set the port priority to 0.

Set port related parameters for MSTP.	**mstp set port** <*port-list*> **priority** <*num*> **msti** <*num*>

This completes the setup. Table 6 shows the complete configuration files for the three routers.

Table 6. Example Configuration Files

X-Pedition Router	Configuration File
Router #1	1: vlan make trunk-port et.4.1 2 : vlan create vlan_100 ip id 100 3: vlan create vlan_200 ip id 200 4: vlan add ports et.4.1 to vlan_100 5: vlan add ports et.4.1 to vlan_200 6: mstp start 7: mstp enable port et.4.1 8: system set name Router#1
Router #2	1: vlan make trunk-port et.4.1 2: vlan make trunk-port et.6.1 3: vlan make trunk-port et.7.1 4: vlan create vlan_100 ip id 100 5: vlan create vlan_200 ip id 200 6: vlan add ports et.4.1 to vlan_100 7: vlan add ports et.4.1 to vlan_200 8: vlan add ports et.6.1 to vlan_100 9: vlan add ports et.6.1 to vlan_200 10: vlan add ports et.7.1 to vlan_100 11: vlan add ports et.7.1 to vlan_200 12: mstp start 13: mstp create msti 1 14: mstp create msti 2 15: mstp enable port et.4.1 16: mstp enable port et.6.1 17: mstp enable port et.7.1 18: mstp map 1 vlan 100 19: mstp map 2 vlan 200 20: mstp set configuration-id name South 21: mstp set port et.6.1 priority 0 msti 1 22: mstp set port et.7.1 priority 0 msti 2 23: system set name Router#2
Router #3	1: vlan make trunk-port et.6.1 2: vlan make trunk-port et.7.1 3: vlan create vlan_100 ip id 100 4: vlan create vlan_200 ip id 200 5: vlan add ports et.6.1 to vlan_100 6: vlan add ports et.6.1 to vlan_200 7: vlan add ports et.7.1 to vlan_100 8: vlan add ports et.7.1 to vlan_200 9: mstp start 10: mstp create msti 1 11: mstp create msti 2 12: mstp enable port et.6.1 13: mstp enable port et.7.1 14: mstp map 1 vlan 100 15: mstp map 2 vlan 200 16: mstp set configuration-id name South 17: mstp set port et.6.1 priority 0 msti 1 18: mstp set port et.7.1 priority 0 msti 2 19: system set name Router#3

Adjusting Spanning-Tree Parameters

You may need to adjust certain spanning tree parameters if the default values are not suitable for your bridge configuration, as described in the following sections.

Setting the Bridge Priority

You can globally configure the priority of an individual bridge when two bridges tie for position as the root bridge, or you can configure the likelihood that a bridge will be selected as the root bridge. The lower the bridge's priority, the more likely the bridge will be selected as the root bridge. This priority is determined by default; however, you can change it.

The bridging priority of the device for the specified MSTI can range from 0 to 61440, being specified in increments of 4096. The **priority** option in the CLI command can range from 0 to 15. The value is internally multiplied by 4096. By default, the bridge priority is 32768 (*num* is set to 8).

To set the bridge priority, enter the following command in Configure mode:

Set the bridge priority for the CIST or MSTP instance of the spanning tree.	**mstp set bridging priority** <*num*> [**msti** <*num*>]

Setting a Port Priority

You can set a priority for an interface. When two bridges tie for position as the root bridge, you configure an interface priority to break the tie. The bridge with the lowest interface value is elected. To set an interface priority, enter the following command in Configure mode:

Establish a priority for the CIST or MSTP instance of the spanning tree.	**mstp set port** <*port-list*> **priority** <*num*> **msti** <*num*>

Assigning Port Costs

Each interface has a port cost associated with it. By convention, the higher the port speed, the lower the port cost. By default, the port cost value is based on the port speed using the IEEE 802.1t values. This is further described in the *X-Pedition NATIVE Command Line Interface Reference Manual*. To assign different port costs, enter the following command in Configure mode:

Set a different port cost other than the defaults for the CIST or MSTP instance of the spanning tree.	**mstp set port** <*port-list*> **port-cost** <*num*> **msti** <*num*>

Adjusting Bridge Protocol Data Unit (BPDU) Intervals

You can adjust the following BPDU intervals:

- Interval between Hello BPDUs
- Forward Delay Interval
- Maximum Idle Interval
- Maximum hops

To adjust the interval between hello time, enter the following command in Configure mode:

Specify the interval between hello time for all ports on the device.	**mstp set bridging hello-time** <*num*>
Specify the interval between hello time for a specific ports.	**mstp enable per-port-hello-mode** **mstp set bridging hello-time** <*num*> **port** <*port-list*>

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. To change the default interval setting, enter the following command in Configure mode:

Set the default of the forward delay interval for the device.	**mstp set bridging forward-delay** <*num*>

If a bridge does not hear BPDUs from the root bridge within a specified interval, it assumes that the network has changed and recomputes the spanning-tree topology. To change the default interval setting, enter the following command in Configure mode:

Change the amount of time a bridge will wait to hear BPDUs from the root bridge.	**mstp set bridging max-age** <*num*>

Max hops are used as an aging timer within a region to limit circulating old information through redundant paths. However, the max hop count should be high enough to account for failed links so that information can still reach all systems. To change the max hops for a device, enter the following command in Configure mode:

Change the maximum hop count.	**mstp set bridging max-hops** <*num*>

Monitoring MSTP

The X-Pedition router provides display of MSTP statistics and configurations contained in the X-Pedition router. To display this information, enter the following commands in Enable mode.

Show MSTP bridge information	**mstp show bridging**
Show the configured MSTIs	**mstp show instances**
Show the VLAN/FID to MSTI table mapping	**mstp show map**
Show MSTP statistics	**mstp show stats** <*num*> [**port** <*port-list*>|**all-mstp-ports**]

MSTP Migration Considerations

The X-Pedition system firmware containing MSTP support is backward compatible with existing spanning tree configurations. That is, all valid spanning tree configurations involving 802.1D, 802.1w or PVST still operate without change as long as MSTP is not configured.

MSTP can be used to provide:

- Single instance spanning tree support. MSTP automatically provides 802.1D or 802.1w service as dictated by the capabilities of the other devices in the network.
- Multiple spanning tree support.

A decision to select MSTP involves the following considerations:

1. To use MSTP, all existing spanning tree (802.1D, 802.1w) and PVST configuration commands must first be negated from the configuration file. MSTP cannot operate concurrently with older spanning tree configurations. MSTP CLI command syntax differs from the other spanning tree and PVST commands in the system.
2. MSTP provides no WAN support whereas the older version of spanning tree is compatible with serial, HSSI and POS line cards, and Frame Relay, PPP and MLPPP protocols.
3. PVST requires a separate spanning tree instance for each VLAN. The total number of PVST instances is limited by the amount of available memory. MSTP is more efficient than PVST in that it can support multiple VLANs on a single instance (MSTI). MSTP is limited to 64 instances. To change a PVST network design to use MSTP, map all PVST spanning trees with identical topology to a single MSTI.

Chapter 9

ATM Configuration Guide

This chapter provides an overview of the Asynchronous Transfer Mode (ATM) features available for the Enterasys X-Pedition router. ATM is a cell switching technology used to establish multiple connections over a physical link, and configure each connection with its own traffic parameters. This provides more control over specific connections within a network.

Note: This release supports PVCs only. To learn more about PVCs, see *Permanent Virtual Circuits (PVCs) on page 553*. Interfaces configured with PVCs do not support LSNAT and VRRP.

The ATM line card provides an ATM interface, allowing integration of ATM with Ethernet and other interfaces within a network topology supported by the X-Pedition router. This chapter discusses the following tasks:

- Creating a Virtual Channel (VC)
- Creating a Service Profile
- Applying a Service Profile
- Enabling Cell Scrambling
- Selecting the Cell Mapping Format
- Setting the Bit Allocation for the Virtual Path Identifier (VPI)
- Mapping a Peer Address to a VC
- Displaying ATM Statistics

Virtual Channels

A virtual channel is a point-to-point connection that exists within a physical connection. You can create multiple virtual channels within one physical connection, with each virtual channel having its own traffic profile. The name “virtual” implies that the connection is located in silicon instead of a physical wire. Refer to *Creating a Service Profile Definition on page 120* for information about defining a set of traffic parameters for a virtual channel.

Creating a Virtual Channel

To create a virtual channel, enter the following command in Configure mode:

Creates a virtual channel.	**atm create vcl port** <*port list*> [**vbr**]

The following is a description of the parameters used to create a virtual channel:

port <*port list*> This parameter identifies the ATM port as well as the virtual channel identifier (VCI) and virtual path identifier (VPI). Specify this parameter in the format: **media.slot.port.vpi.vci**

- **media** Specifies the media type. This is **at** for ATM ports**.**
- **slot** Specifies the slot number where the module is installed.
- **port** Specifies the port on where you want to create a virtual channel.
- **vpi** Specifies the Virtual Path Identifier. This number identifies a particular virtual path.
- **vci** Specifies the Virtual Channel Identifier. This number identifies a particular virtual channel. The combination of VPI and VCI is known as the VPI/VCI pair, and identifies the virtual channel.

  Note: Do not specify VCI numbers 0 through 31. Some protocols use these VPI/VCI pairs for signaling purposes.

vbr Opens the VC with a default VBR service. All VCs to which you will apply a VBR service must be created with this option for traffic shaping to behave properly. If you specify the 'vbr' option, you may apply only VBR services to the VC. If you do not specify the 'vbr' option, you may apply only UBR and CBR services to the VC.

Note: Traffic on ATM Virtual Circuits configured with a Variable Bit Rate (nrt-vbr or rt-vbr) traffic descriptor will not obey the configured traffic descriptor's parameters. ATM policing mechanisms will drop nonconforming ATM cells.

Virtual channel and IPX Routing

The following commands create an ATM virtual channel on an ATM port and associate the port with an IPX interface. This allows IPX routing between two IPX interfaces. As with any IPX interface, IPX routing using RIP (the default) will begin when you configure an IPX interface.

```
xp(config)# atm create vcl port at.3.1.1.100
xp(config)# interface create ipx finance address 01234567 peer-address 01234567.00:00:1d:a9:8c:a1
port at.3.1.1.100
xp(config)# interface create ipx marketing address 01234569 port et.1.1
```

Setting the Operation Mode for a Virtual Channel

You can set certain operational modes on a VC, such as enabling forced-bridging. Enabling forced-bridging forces the VC to encapsulate all ingress/egress traffic into a Layer-2 frame. As a result, all traffic passing through the VC group is formatted as bridged traffic. This feature is advantageous when inter-operability between different vendor equipment is a concern.

To enable forced-bridging on a VC, enter the following command in Configure mode:

Enables forced-bridging on a virtual channel.	**atm set vcl port** <*port*> **forced-bridged**

The following is a description of the parameters used with this command:

port <port> This parameter identifies the ATM port as well as the virtual channel identifier (VCI) and virtual path identifier (VPI). Specify this parameter in the format: **media.slot.port.vpi.vci**

- **media** Specifies the media type. This is at for ATM ports.
- **slot** Specifies the slot number where the module is installed.
- **port** Specifies the port on where you want to create a virtual channel.
- **vpi** Specifies the Virtual Path Identifier. This number identifies a particular virtual path.
- **vci** Specifies the Virtual Channel Identifier. This number identifies a particular virtual channel.

forced-bridged Enables all traffic to be encapsulated as Layer-2 bridged traffic. This parameter can be used for inter-operability between the X-Pedition router and other vendor products.

Service Profile Definition

ATM provides the ability to specify various parameters for each virtual channel. These parameters include traffic parameters which define the bandwidth characteristics and delay guarantees. You can apply a different set of traffic parameters for each virtual channel. This provides network administrators more control of their network resources and more options in connections to accommodate different user needs.

If the bandwidth of a service category is exceeded, packets may be dropped—including routing protocol packets. This may also cause routes to drop.

Creating a Service Profile Definition

To create a service profile definition, enter the following command in Configure mode:

Creates a service profile definition.	**atm define service** <*string*> **[srv-cat cbr| ubr| rt-vbr| nrt-vbr] [pcr| pcr-kbits] [scr| scr-kbits] [mbs] [encaps llc-mux| vc-mux] [oam on| off] [oam-f5-type current-segment| end-to-end]**

The following is a description of the parameters used to create a service profile definition:

service <*string*> Specifies a name for the service profile definition. The maximum length is 32 characters.

srv-cat Defines the category for the service profile definition (UBR is the default):

- **ubr** Unspecified Bit Rate. This service category is strictly best effort and runs at the available bandwidth. Users may limit the bandwidth by specifying a PCR value. The SCR and MBS are ignored. This service class is intended for applications that do not require specific traffic guarantees. UBR is the **default**.
- **cbr** Constant Bit Rate. This service category provides a guaranteed constant bandwidth specified by the Peak Cell Rate (PCR). This service category requires only the PCR value. The Sustainable Cell Rate (SCR) and Maximum Burst Size (MBS) values are ignored. This service category is intended for applications that require constant cell rate guarantees such as uncompressed voice or video transmission.
- **nrt-vbr** Non Real-Time Variable Bit Rate. This service category provides a guaranteed constant bandwidth (specified by the SCR), but also provides for peak bandwidth requirements (specified by the PCR). This service category requires the PCR, SCR, and MBS options and is intended for applications that can accommodate bursty traffic with no need for real-time guarantees.

rt-vbr Real-Time Variable Bit Rate. This service category provides a guaranteed constant bandwidth (specified by the SCR), but also provides for peak bandwidth requirements (specified by the PCR). This service category requires the PCR, SCR, and MBS options and is intended for applications that can accommodate bursty real-time traffic such as compressed voice or video.

pcr Specifies the Peak Cell Rate, which defines the maximum cell transmission rate. The **default** is 353207 cells/sec. This parameter is valid for CBR, rtVBR, nrtVBR, and UBR service categories. This parameter is optional for UBR.

pcr-kbits Specifies the Peak Cell Rate, which defines the maximum cell transmission rate, expressed in kbits/sec. The **default** is 149759 kbits/sec (353207 cells/sec). This is the same as PCR, but is expressed in kbits/sec, and therefore may be a more convenient form. However, since the natural unit for ATM is cells/sec, there may be a difference in the actual rate because the kbit/sec value may not be an integral number of cells. This parameter is valid for CBR, rtVBR, nrtVBR, and UBR service categories.

scr The Sustainable Cell Rate. This rate specifies the average cell rate, expressed in cells/sec. The **default** is 0 cells/sec. This parameter is valid only for rtVBR and nrtVBR service categories.

scr-kbits The Sustainable Cell Rate in kbts/sec. This rate specifies the average cell rate, expressed in kbits/sec. The **default** is 0 kbits/sec. This is the same as SCR, but is expressed in kbits/sec, and therefore may be a more convenient form. However, since the natural unit for ATM is cells/sec, there may be a difference in the actual rate because the kbit/sec value may not be an integral number of cells. This parameter is valid only for rtVBR and nrtVBR service categories.

mbs Specifies the Maximum Burst Size in cells. **MBS** specifies how many cells can be transmitted at the Peak Cell Rate You may specify between 2 and 255 cells—the **default** is 0 cells. This parameter is valid only for rtVBR and nrtVBR service categories.

encaps The encapsulation scheme used to transport multi-protocol data over the AAL5 layer. This scheme will be LLC-MUX (logical link control based multiplexing) or vc-mux (virtual channel based multiplexing). The default scheme is LLC-MUX.

oam OAM (Operation, Administration, and Management) loopback cells are used to provide loopback capabilities and confirm whether a VC connection is up. F5 OAM and end-to-end segments are supported, which provides loopback capabilities on a VC connection level. This parameter turns OAM **ON** or **OFF** on the PVC. The default is **OFF**. OAM OFF means that the X-Pedition router responds to F5 OAM requests, but will not generate F5 OAM responses.

oam-f5-type

Designates whether the OAM loopback cells have meaning in the current segment (the default) or end-to-end.

Applying a Service Profile Definition

To apply a service profile definition to a virtual channel, virtual path, or an ATM port, enter the following command in Configure mode:

Applies a service profile definition.	**atm apply service** <*string*> **port** <*port list*>

The following is a description of the parameters used to apply a service profile definition:

service <*string*> Specifies the name of the service profile definition which you want to apply. The maximum length is 32 characters.

port <*port list*> Specifies the port, in the format: **`media.slot.port.vpi.vci`**

- **media** Specifies the media type. This is **at** for ATM ports**.**
- **slot** Specifies the slot number where the module is installed.
- **port** Specifies the port number.
- **vpi** Specifies the Virtual Path Identifier. This parameter identifies the virtual path. This parameter is optional.
- **vci** Specifies the Virtual Channel Identifier. This parameter identifies the virtual channel. This parameter is optional.

An important concept when applying service profile definitions is the concept of *inheritance*. Since a service profile definition can be applied to a virtual channel, virtual path, or an ATM port, the actual connection can inherit the service profile definition from any one of the three. The virtual channel will inherit the service profile definition that is directly applied on it. If no service profile was applied to the virtual channel, the connection will inherit the service profile applied to the virtual path. If no service profile definition is applied to the virtual path, the connection will inherit the service profile applied to the ATM port. If no service profile is applied to the port, the **default** service class UBR is applied.

Cell Scrambling

Cell scrambling is useful for optimizing the transmission density of the data stream. Since all transmissions use the same source clock for timing, scrambling the cell using a random number generator converts the data stream to a more random sequence. This ensures optimal transmission density of the data stream.

Enabling Cell Scrambling

This command allows you to enable cell scrambling for the PDH (plesiochronous digital hierarchy) physical (PHY) interfaces available on the ATM line card, such as T1, T3, E1, and E3 PHYs.

Note: Refer to the SONET chapter in the Command Line Interface Manual for information about cell scrambling on SONET PHY interfaces.

To enable cell scrambling on an ATM port, enter the following command in Configure mode:

Enables cell scrambling on an ATM port.	**atm set port** <*port list*> **pdh-cell-scramble on| off**

The following is a description of the parameters used to enable cell scrambling:

port <*port list*> Specifies the port, in the format: **media.slot.port.** Specify **all-ports** to enable cell scrambling on all ports.

media Specifies the media type. This is **at** for ATM ports**.**

slot Specifies the slot number where the module is installed.

port Specifies the port number.

pdh-cell-scramble on|off
Specify **on** to enable cell scrambling. Specify **off** to disable cell scrambling.

Cell Mapping

The ATM cells are mapped into a PDH (E3, T3, E1) frame using two different mapping formats. The two mapping formats available are **direct** ATM cell mapping and physical layer convergence protocol (**PLCP**) mapping.

Selecting the Cell Mapping Format

To select a cell mapping format on an ATM port, enter the following command in Configure mode:

Selects a cell mapping format on an ATM port.	**atm set port** <*port list*> **cell-mapping direct| plcp**

The following is a description of the parameters used to select the cell mapping format:

port <*port list*> Specifies the port, in the format: **`media.slot.port`**. Specify **all-ports** to select the cell mapping format for all ports.

media Specifies the media type. This is **at** for ATM ports**.**

slot Specifies the slot number where the module is installed.

port Specifies the port number.

cell-mapping direct|plcp
Specify **direct** to select direct ATM cell mapping. Specify **plcp** to select PLCP mapping.

VPI Bit Allocation

The Virtual Path Identifier defines a virtual path, a grouping of virtual channels transmitting across the same physical connection. The actual number of virtual paths and virtual channels available on an ATM port depends upon how many bits are allocated for the VPI and VCI, respectively. By default, there is 1 bit allocated for VPI and 11 bits allocated for VCI. You can specify a different allocation of bits for VPI and VCI for a port.

There are 12 bits available for each VPI/VCI pair per port. The number of bits allocated define the amount of VPI and VCI values available. The following equations define the number of virtual paths and virtual channels:

# of virtual paths = 2^n; where *n* is the number of bits allocated for VPI and *n* is a value from 1 to 4

# of virtual channels = $2^{(12-n)}$; where (12-*n*) is the number of bits allocated for VCI

The bit allocation command allows you to set the number of bits allocated for VPI; the remaining number of bits are allocated for VCI. Since there are only 12 bits available for each VPI/VCI pair on an ATM port, the more bits you allocate for VPI, the fewer bits remain for VCI.

Setting the Bit Allocation for VPI

To set the bit allocation for VPI on an ATM port, enter the following command in Configure mode:

Sets the number of bits allocated for VPI on a port.	**atm set port** <*port list*> **vpi-bits** <*num*>

The following is a description of the parameter used to set the number of bits allocated for VPI on an ATM port:

port <*port list*> This parameter identifies the ATM port. Specify this parameter in the format: **media.slot.port**. Specify **all-ports** to set bit allocation on all ports.

- **media** Specifies the media type. This is **at** for ATM ports**.**
- **slot** Specifies the slot number where the module is installed.
- **port** Specifies the port number.

vpi-bits <*num*> This parameter sets the number of bits for VPI. Specify any number between 1 and 4. The **default** is 1.

Peer Address Mapping

Peer addresses allow you to specify a certain destination address for a specific virtual channel. This allows you to set the destination address for a virtual channel using the **atm set peer-addr** command. This way, a virtual channel can be dedicated to handle traffic between two specific devices.

Note: The default interface type is "broadcast." If you connect an interface to a router capable of "point-to-point" only, you must specify the interface type as point-to-point within the configuration. In this case, you must specify a peer address.

Mapped addresses are useful when you do not want to specify the peer address for the ATM port using the **interface create** command. This would be the case if the interface is created for a VLAN and there are many peer addresses on the VLAN. If any of the peers on the VLAN do not support InArp or IPCP/IPXCP, then a mapped address must be configured to determine the destination address.

Note: If a virtual channel is configured in the "VC-Muxing" mode or if the interface type is set to "point-to-point," you must specify a peer address as part of the **interface create** command.

Mapping a Peer Address to a Virtual Channel

To map a peer address on an ATM port to a specific virtual channel, enter the following command in Configure mode:

Maps a peer address to a specific virtual channel.	**atm set peer-addr port** <*port*> **ip-address** <*ipaddr*> | **ipx-address** <*netaddr*>.<*macaddr*>

The following is a description of the parameters used to map a peer address to a virtual channel:

port <*port*> Specifies a single port, including virtual channel, in the format: **media.slot.port.vpi.vci**.

- **media** Is the media type. This is always **at** for an ATM port.
- **slot** The slot number where the module is installed.
- **port** The number of the port through which data is passing.
- **vpi** Virtual Path Identifier.
- **vci** Virtual Channel Identifier. The virtual channel to which you will map a peer address. This allows you to define an interface (assigned to a VLAN) with multiple peer membership. In addition, it allows you to assign intercommunication between specific peers on that VLAN. This feature is helpful when a peer does not support inverse ARP or IPCP/IPXCP; in this case, you can configure any peer mapping to that peer to define the peer destination address.

Note: If the associated virtual channel is in the VC-muxing mode, use virtual peer mapping.

ip-address <*ipaddr*>
Specifies an IP address for the peer. Specify a unicast IP address and netmask value in the following format: **a.b.c.d/e.** This IP address will be mapped to the VC.

ipx-address <*netaddr*>.<*macaddr*>
Specifies an IPX address for the peer. Specify an IPX network and node address in the following format: **a1b2c3d4.aa:bb:cc:dd:ee:ff**. If a <*macaddr*> is not specified, a wildcard address is used. This IPX address will be mapped to the VC.

Displaying ATM Port Information

There are a variety of ATM statistics that can be accessed through the command line interface. The **atm show** commands can only be used in Enable mode.

To display information about the VPL configurations on an ATM port:

Displays the VPL configurations on an ATM port.	**atm show vpl port** <*port list*> | **all-ports**

The following is an example of the information that is displayed with the command listed above:

```
xp(atm-show)# vpl port at.9.1

VPL Table Contents for Slot 9, Port 1:
   Virtual Path Identifier:     1
   Administrative Status:       Up
   Operational Status:          Up
   Last State Change:           1581
   Service Definition:          default-OC3
      Service Class:            UBR
      Peak Bit Rate:            Best Effort
      Encapsulation Type:       LLC Multiplexing
      F5-OAM:                   Responses Only
      F5-OAM-Type:              Current Segment
```

The following is a description of the display fields:

- Virtual Path Identifier — Identifies a particular VP.
- Administrative Status — Shows whether the VP is a viable network element.
  Up indicates a viable network element.
  Down indicates a non-viable network element.

- Operational Status — Shows whether the VP is passing traffic.
  Up indicates traffic.
  Down indicates no traffic.
- Last State Change — Shows the last time the VP went up or down. Time is in seconds relative to system bootup.
- Service Definition — Shows the name of the defined service and its traffic parameters

To display information about the service definition on an ATM port:

Displays the service definition on an ATM port.	**atm show service| all**

The following is an example of the information that is displayed with the command listed above:

```
xp# atm show service all

default-OC3
   Service Class:          UBR
   Peak Bit Rate:          Best Effort
   Sustained Bit Rate:     0 Kbits/sec (0 cps)
   Maximum Burst Size:     0 cells
   Encapsulation Type:     LLC Multiplexing
   F5-OAM:                 Responses Only
   F5-OAM-Type:            Current Segment
```

The following is a description of the display fields:

- Service Class — Shows the type of service class.
  UBR indicates Unspecified Bit Rate
  CBR indicates Constant Bit Rate
  RT-VBR indicates Real-time Variable Bit Rate
  NRT-VBR indicates Non Real-time Variable Bit Rate
- Peak Bit Rate — Shows the maximum bit transmission rate.
- Sustained Bit Rate — Shows the average bit transmission rate (in Kilobits per second).
- Maximum Burst Size — Shows how many cells can be transmitted at the Peak Bit Rate.
- Encapsulation Type — Shows the encapsulation scheme to transport multi protocol data over the AAL5 layer.

  LLC-MUX indicates logical link control encapsulation (**default**).
  VC-MUX indicates VC-based multiplexing encapsulation.

- F5-OAM — Shows how OAM (Operation, Administration, and Management) loopback cells provide loopback capabilities and confirm whether a VC connection is up.

  Responses Only indicates that the port will respond but doesn't generate OAM cells.
  Requests & Responses indicates that the port will respond and generate OAM cells.

To display information about the port settings on an ATM port:

Displays the port setting configurations on an ATM port.	**atm show port-settings** <*port list*>| **all-ports**

The following is an example of the information that is displayed with the command listed above (for a PDH PHY interface):

```
xp(atm-show)# port-settings at.9.1
Port information for Slot 9, Port 1:
   Port Type:              T3 ATM coaxial cable
   Xmt Clock Source:        Local
   Scramble Mode:           Payload
   Line Coding:             B3ZS
   Cell Mapping:            Direct
   Framing                  Cbit-Parity
   VC Mode:                 1 bit of VPI, 11 bits of VCI
   Service Definition:      default-OC3
      Service Class:        UBR
      Peak Bit Rate:        Best Effort
      Sustained Bit Rate:    0 Kbits/sec (0 cps)
      Maximum Burst Size:   0 cells
      Encapsulation Type:   LLC Multiplexing
      F5-OAM:               Requests & Responses
      F5-OAM-Type:          Current Segment
```

- Port Type — Shows the type of PHY interface for the port.

- Xmt Clock Source — Shows the timing source for the port.
  Local indicates the onboard clock oscillator as the timing source.
  Loop indicates the receiver input as the timing source.

- Scramble Mode — Shows the scramble/descramble mode for the port.
  None indicates no scrambling.
  Payload indicates scrambling of the payload only.
  Frame indicates scrambling of the stream only.
  Both indicates scrambling of payload and stream.

- Line Coding — Shows the particular DS1/T1 and DS3/T3 coding convention.

- Cell Mapping — Shows the format used to map ATM cells.
  Direct indicates direct cell mapping.
  Plcp indicates physical layer convergence protocol mapping.

- Framing — Shows the type of framing scheme.
  cbit-parity is used for T3 framing.
  m23 is used for T3 framing.
  esf indicates extended super frame and is used for T1 framing.
  g832 is used for E3 framing.
  g751 is used for E3 framing.
- VC Mode — Shows the bit allocation for vpi and vci.
- Service Definition — Shows the name of the defined service on the port and its traffic parameters.

The following is an example of the information that is displayed with the command listed above (for a SONET PHY interface):

```
xp (atm-show)# port-settings at.8.1
Port information for Slot 8, Port 1:
   Port Type:              SONET STS-3c MMF
   Xmt Clock Source:        Local
   VC Mode:                 1 bit of VPI, 11 bits of VCI
   Service Definition:      default-OC3
      Service Class:        UBR
      Peak Bit Rate:        Best Effort
      Encapsulation Type:   LLC Multiplexing
      F5-OAM:               Requests & Responses
      F5-OAM-Type:          Current Segment
```

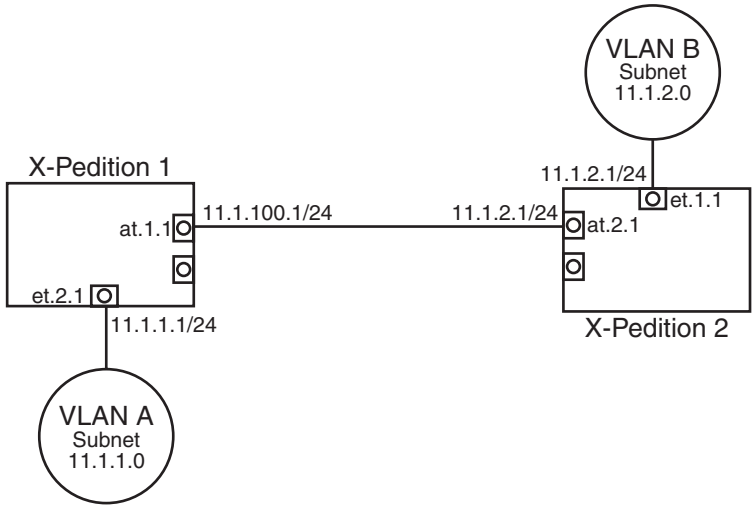
- Port Type — Shows the type of PHY interface for the port.
- Xmt Clock Source — Shows the timing source for the port.
  Local indicates the onboard clock oscillator as the timing source.
  Loop indicates the receiver input as the timing source.
- VC Mode — Shows the bit allocation for vpi and vci.
- Service Definition — Shows the name of the defined service on the port and its traffic parameters.

To display traffic statistics for a virtual channel:

Displays the traffic statistics for a virtual channel.	**atm show stats port** <*port list*>

ATM Sample Configuration 1

Consider the following network configuration:



The network shown consists of two X-Pedition routers, VLAN A, and VLAN B. Both routers have an ATM module with two ATM ports. Also both routers contain a 10/100 TX Ethernet module. X-Pedition 1 (XP1) is connected to VLAN A through Ethernet port et.2.1, while X-Pedition 2 (XP2) is connected to VLAN B through Ethernet port et.1.1.

This example shows how to configure this network so that we are able to pass traffic from VLAN B to VLAN A. The following steps will lead you through the configuration process.

Configuring an Interface on each Ethernet Port

There are two separate VLANs in this network, VLAN A and VLAN B. VLAN A is connected to Ethernet port et.2.1 on X-Pedition 1, and VLAN B is connected to Ethernet port et.1.1 on X-Pedition 2.

Apply an interface on both Ethernet ports. Creating an interface on an Ethernet port assigns a network IP address and submask on that port.

The following command assigns an IP address of 11.1.1.1/24 on port et.2.1 on XP1:

```
xp1(config)# interface create ip subnetA address-netmask 11.1.1.1/24 port et.2.1
```

The following command assigns an IP address of 11.1.2.1/24 on port et.1.1 on XP2:

```
xp2(config)# interface create ip subnetB address-netmask 11.1.2.1/24 port et.1.1
```

Creating a Virtual Channel

Create a VC to connect ATM port at.1.1 on X-Pedition 1 to ATM port at.2.1 on X-Pedition 2. Use the following command to create a virtual channel on XP1 with vpi=0 and vci=100:

```
xp1(config)# atm create vcl port at.1.1.0.100
```

You must now configure a corresponding vpi/vci pair on ATM port at.2.1. Use the following command to create a virtual channel on XP2 with vpi=0 and vci=100:

```
xp2(config)# atm create vcl port at.2.1.0.100
```

Note that you are using the same vpi and vci on both routers. This establishes a common VC from one ATM port to another ATM port.

Defining an ATM Service Profile

After creating a VC connection from ATM port at.1.1 to at.2.1, the next step is to define an ATM service profile for this connection.

The following command lines defines a service profile named ‘cbr1m’ on both XP1 and XP2 where CBR is the service category and peak cell rate is set to 10000 kcells/second:

```
xp1(config)# atm define service cbr1m srv-cat cbr pcr-kbits 10000
```

```
xp2(config)# atm define service cbr1m srv-cat cbr pcr-kbits 10000
```

Applying an ATM Service Profile

After defining a service profile on X-Pedition 1 and X-Pedition 2, apply them to the VC connection we created earlier.

The following command line applies the service profile ‘cbr1m’ to the VC (vpi=0, vci=100) on ATM port at.1.1 of XP1:

```
xp1(config)# atm apply service cbr1m port at.1.1.0.100
```

The following command line applies the service profile ‘cbr1m’ to the VC (vpi=0, vci=100) on ATM port at.2.1 of XP2:

```
xp2(config)# atm apply service cbr1m port at.2.1.0.100
```

Configuring an Interface on an ATM Port

The next step is to configure an interface for each ATM port. Creating an interface on an ATM port assigns a network IP address and submask on that port, and assigns it to a specified VC (VPI/VCI pair). Since a VC is a connection in the ATM Layer only, creating an interface for an ATM port is necessary to establish a connection in the IP network layer. You can assign a peer-address to an ATM port interface. This peer-address specifies the IP address for the other end of the VC connection.

Note: Enterasys recommends that you use alphabetic characters when defining interface names—purely numeric interfaces will be interpreted as IP addresses. The X-pedition router displays interface names up to 32 characters in length.

Set the IP interface name as ‘atm1’ and IP address as 11.1.100.1/24 on ATM port at.1.1.0.100. The following command line configures the interface on XP1:

```
xp1(config)# interface create ip atm1 address-netmask 11.1.100.1/24 peer-address 11.1.100.2 port
at.1.1.0.100
```

Set the IP interface name as ‘atm2’ and IP address as 11.1.100.2/24 on ATM port at.2.1.0.100. The following command line configures the interface on XP2:

```
xp2(config)# interface create ip atm2 address-netmask 11.1.100.2/24 peer-address 11.1.100.1 port
at.2.1.0.100
```

Configuring an IP Route

The next step is to add an IP route which will specify a gateway address to reach a certain subnet. You already configured IP interfaces for both Ethernet ports. VLAN B (connected to IP interface 11.1.2.1/24) belongs to the subnet 11.1.2.0. Similarly, VLAN A (connected to IP interface 11.1.1.1/24) belongs to the subnet 11.1.1.0.

Creating an IP route allows the interfaces on the ATM ports to act as gateways to any subnet. Traffic from VLAN A reaches the Ethernet port on X-Pedition 1 and is automatically directed to the gateway address (interface on the ATM port for X-Pedition 2). Then the traffic travels through the VC and arrives at the Ethernet port connected to VLAN B. Add the IP route for the subnet 11.1.2.0. The following command line configures the route on XP1:

```
xp1(config)# ip add route 11.1.2.0/24 gateway 11.1.100.2
```

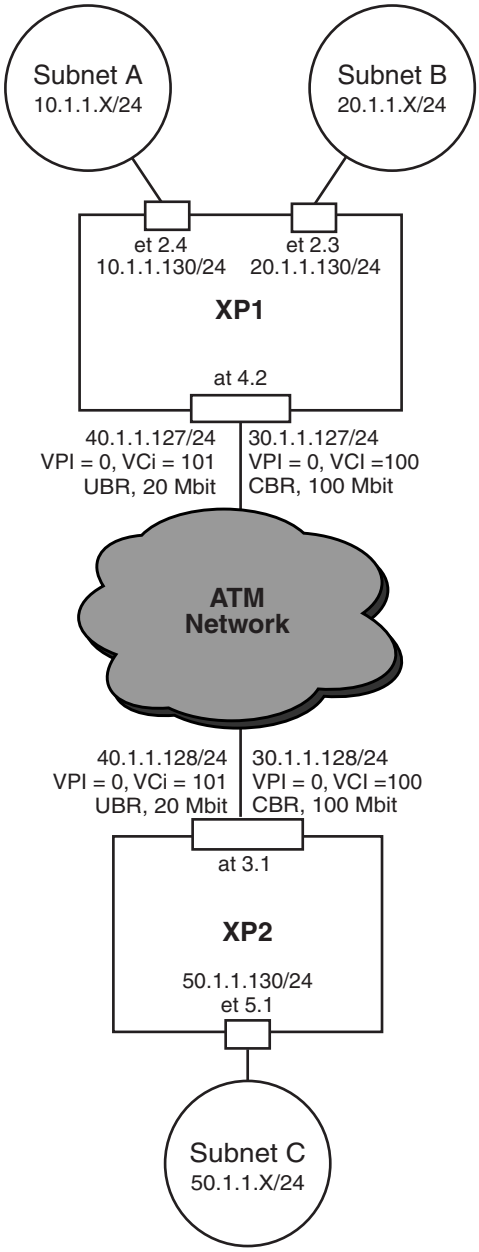
Add the IP route for the subnet 11.1.1.0. The following command line configures the route on XP2:

```
xp2(config)# ip add route 11.1.1.0/24 gateway 11.1.100.1
```

The gateways specified are the interface for the ATM port on the other end of the VC connection.

ATM Sample Configuration 2

Consider the following network configuration:



Suppose you are a network administrator in charge of managing a network with two client groups: Subnet A and Subnet B. These two client groups have very different bandwidth needs and requirements for their respective users. Subnet A consists of users who need access to a high bandwidth connection, able to support video conferencing. Subnet B consists of users who require less stringent requirements and are mainly concerned with email and server backup type of traffic.

As the network administrator, you can accommodate both client groups using only one ATM physical connection. This is accomplished by setting up two VCs on the ATM port, each with its own service profile definitions. The example shows how to configure this network. The following sections will lead you through the configuration process.

This will be done in two steps. The first step is to configure the network for traffic from Subnet A and Subnet B to Subnet C. The second step is to configure the network for traffic from Subnet C to Subnet A and Subnet B.

Note: There are two routers in this example. The command line prompt indicates whether the command is being inputted in X-Pedition router1 (xp1) or X-Pedition router2 (xp2).

Traffic from Subnet A and Subnet B to Subnet C

Step 1: Configuring an Interface on Each Ethernet Port

The following command assigns an IP address of 10.1.1.130/24 on port et.2.4:

```
xp1(config)# interface create ip subnetA address-netmask 10.1.1.130/24 port et.2.4
```

The following command assigns an IP address of 20.1.1.130/24 on port et.2.3:

```
xp1(config)# interface create ip subnetB address-netmask 20.1.1.130/24 port et.2.3
```

The following command assigns an IP address of 50.1.1.130/24 on port et.5.1:

```
xp2(config)# interface create ip subnetC address-netmask 50.1.1.130/24 port et.5.1
```

Step 2: Creating a Virtual Channel

The following command creates a virtual channel on port at.4.2 with VPI=0 and VCI=100:

```
xp1(config)# atm create vcl port at.4.2.0.100
```

The following command creates a virtual channel on port at.4.2 with VPI=0 and VCI=101:

```
xp1(config)# atm create vcl port at.4.2.0.101
```

The following command creates a virtual channel on port at.3.1 with VPI=0 and VCI=100:

```
xp2(config)# atm create vcl port at.3.1.0.100
```

The following command creates a virtual channel on port at.3.1 with VPI=0 and VCI=101:

```
xp2(config)# atm create vcl port at.3.1.0.101
```

Step 3: Configuring an Interface on Each ATM Port

The following command assigns an IP address of 40.1.1.127/24 on port at.4.2.0.101:

```
xp1(config)# interface create ip ubrservice address-netmask 40.1.1.127/24 peer-address 40.1.1.128
port at.4.2.0.101
```

The following command assigns an IP address of 30.1.1.127/24 on port at.4.2.0.100:

```
xp1(config)# interface create ip cbrservice address-netmask 30.1.1.127/24 peer-address 30.1.1.128
port at.4.2.0.100
```

The following command assigns an IP address of 40.1.1.128/24 on port at.3.1.0.101:

```
xp2(config)# interface create ip ubrservice address-netmask 40.1.1.128/24 peer-address 40.1.1.127
port at.3.1.0.101
```

The following command assigns an IP address of 30.1.1.128/24 on port at.3.1.0.100:

```
xp2(config)# interface create ip cbrservice address-netmask 30.1.1.128/24 peer-address 30.1.1.127
port at.3.1.0.100
```

Step 4: Defining an ATM Service Profile

The following commands define a service profile called ‘ubrservice’ with PCR of 20 Mbits:

```
xp1(config)# atm define service ubrservice srv-cat ubr pcr-kbits 20000
```

```
xp2(config)# atm define service ubrservice srv-cat ubr pcr-kbits 20000
```

The following commands define a service profile called ‘cbrservice’ with PCR of 100 Mbits:

```
xp1(config)# atm define service cbrservice srv-cat cbr pcr-kbits 100000
```

```
xp2(config)# atm define service cbrservice srv-cat cbr pcr-kbits 100000
```

Step 5: Applying an ATM Service Profile

The following command applies the ‘ubrservice’ service profile on at.4.2.0.101:

```
xp1(config)# atm apply service ubrservice port at.4.2.0.101
```

The following command applies the ‘cbrservice’ service profile on at.4.2.0.100:

```
xp1(config)# atm apply service ubrservice port at.4.2.0.100
```

The following command applies the ‘ubrservice’ service profile on at.3.1.0.101:

```
xp2(config)# atm apply service cbrservice port at.3.1.0.101
```

The following command applies the ‘cbrservice’ service profile on at.3.1.0.100:

```
xp2(config)# atm apply service cbrservice port at.3.1.0.100
```

Step 6: Create an IP ACL

The following command creates an IP ACL policy for port et.2.4:

```
xp1(config)# acl subnetAtoCacl permit 10.1.1.0/24 any any any
```

The following command creates an IP ACL policy for port et.2.3:

```
xp1(config)# acl subnetBtoCacl permit 20.1.1.0/24 any any any
```

Step 7: Specify a Gateway for an IP Policy

The following command specifies 40.1.1.128/24 as the gateway for the IP ACL ‘subnetAacl’:

```
xp1(config)# ip-policy subnetAtoCpolicy permit acl subnetAtoCacl next-hop-list 40.1.1.128/24 action policy-first
```

The following command specifies 30.1.1.128/24 as the gateway for the IP ACL ‘subnetBacl’:

```
xp1(config)# ip-policy subnetBtoCpolicy permit acl subnetBtoCacl next-hop-list 30.1.1.128/24 action policy-first
```

Step 8: Apply the IP Policy to the Ethernet Ports

The following command applies the IP policy ‘subnetApolicy’ to port et.2.4:

```
xp1(config)# ip-policy subnetAtoCpolicy apply interface subnetA
```

The following command applies the IP policy ‘subnetBpolicy’ to port et.2.3:

```
xp1(config)# ip-policy subnetBtoCpolicy apply interface subnetB
```

Traffic from Subnet C to Subnet A and Subnet B

Step 9: Create an IP ACL

The following command creates an IP ACL policy for port et.5.1 to subnet A:

```
xp2(config)# acl subnetCtoAacl permit 50.1.1.0/24 10.1.1.0/24 any any
```

The following command creates an IP ACL policy for port et.5.1 to subnet B:

```
xp2(config)# acl subnetCtoBacl permit 50.1.1.0/24 20.1.1.0/24 any any
```

Step 10: Specify a Gateway for an IP Policy

The following command specifies 40.1.1.127/24 as the gateway for the IP ACL ‘subnetCtoAacl’:

```
xp2(config)# ip-policy subnetCtoApolicy permit acl subnetCtoAacl next-hop-list 40.1.1.127/24 action
policy-first
```

The following command specifies 30.1.1.127/24 as the gateway for the IP ACL ‘subnetCtoBacl’:

```
xp2(config)# ip-policy subnetCtoBpolicy permit acl subnetCtoBacl next-hop-list 30.1.1.127/24 action
policy-first
```

Step 11: Apply the IP Policy to the Ethernet Port

The following command applies the IP policy ‘subnetCtoApolicy’ to port et.5.1:

```
xp2(config)# ip-policy subnetCtoApolicy apply interface subnetC
```

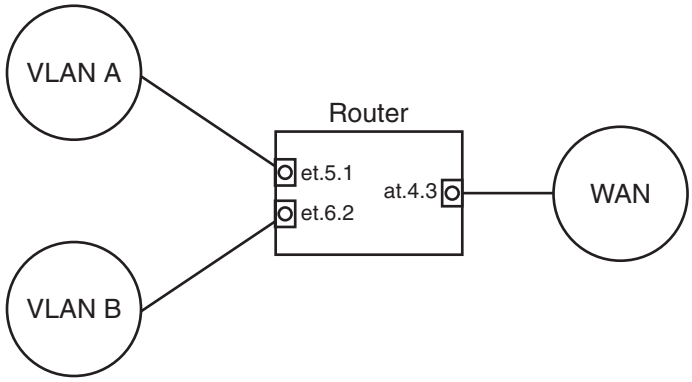
The following command applies the IP policy ‘subnetCtoBpolicy’ to port et.5.1:

```
xp2(config)# ip-policy subnetCtoBpolicy apply interface subnetC
```

All traffic between SubnetA and Subnet C will now travel over the 20 mbit VCL. Traffic between Subnet B and Subnet C will travel over the 100 mbit VCL.

ATM Sample Configuration 3

Consider the following network configuration:



Suppose you are a network administrator in charge of managing a network with two client groups, VLAN A and VLAN B. These two client groups have very different needs and requirements for their users. VLAN A consists of users who need access to a high bandwidth connection able to support video conferencing. VLAN B consists of users who require less bandwidth and are mainly concerned with email and server backup traffic.

This network configuration is similar to the *ATM Sample Configuration 2 on page 134*. However, the use of bridging and VLANs will provide the solution to your networking requirements.

You as the network administrator can accommodate their different requirements using only one ATM port. The following steps will lead you through the configuration process.

Configure an Interface on an Ethernet Port

There are two separate VLANs in this network, VLAN A and VLAN B. VLAN A is connected to ethernet port et.5.1, and VLAN B is connected to ethernet port et.6.2.

Apply an interface on both ethernet ports. Creating an interface on an ethernet port assigns a network IP address and submask on that port.

The following command assigns an IP address of 11.1.1.1/24 on port et.5.1:

```
xp(config)# interface create ip subnetA address-netmask 11.1.1.1/24 port et.5.1
```

The following command assigns an IP address of 11.1.2.1/24 on port et.6.2:

```
xp(config)# interface create ip subnetB address-netmask 11.1.2.1/24 port et.6.2
```

Create a Virtual Channel

The following command line creates a virtual channel with VPI=0 and VCI=100:

```
xp(config)# atm create vcl port at.4.3.0.100
```

The following command line creates another virtual channel with VPI=0 and VCI=101:

```
xp(config)# atm create vcl port at.4.3.0.101
```

Create a VLAN

The ATM OC-3c line card supports only IP-based VLANs.

The following command line creates an IP-based VLAN A with ID number 2:

```
xp(config)# vlan create VLAN_A ip id 2
```

The following command line creates an IP-based VLAN B with ID number 3:

```
xp(config)# vlan create VLAN_B ip id 3
```

Associate a Virtual Channel to a VLAN

The following command line associates VLAN A (et.5.1) to the virtual channel with VPI=0 and VCI=100 (at.4.3.0.100):

```
xp(config)# vlan add ports et.5.1,at.4.3.0.100 to VLAN_A
```

The following command line associates VLAN B (et.6.2) to the virtual channel with VPI=0 and VCI=101 (at.4.3.0.101):

```
xp(config)# vlan add ports et.6.2,at.4.3.0.101 to VLAN_B
```

Define an ATM Service Class

The following command lines define a service class named ‘vlanA’ where CBR is the service category and peak cell rate is set to 100000 kcells/second to ensure proper support for video conferencing:

```
xp(config)# atm define service vlanA srv-cat cbr pcr-kbits 100000
```

The following command line defines a service class named ‘vlanB’ where UBR is the service category:

```
xp(config)# atm define service vlanB srv-cat ubr
```

Apply an ATM Service Class

The following command line applies service class ‘vlanA’ on ATM port at.4.3.0.100:

```
xp(config)# atm apply service vlanA port at.4.3.0.100
```

The following command line applies service class ‘vlanB’ on ATM port at.4.3.0.101:

```
xp(config)# atm apply service vlanB port at.4.3.0.101
```

Configure an Interface on an ATM Port

For both command line examples, Enterasys recommends that you use alphabetic characters when defining interface names—purely numeric interfaces will be interpreted as IP addresses. The X-Pedition router displays interface names up to 32 characters in length.

The following command line sets an interface name ‘vlanA’ and IP address 11.1.100.1/24 on ATM port at.4.3.0.100:

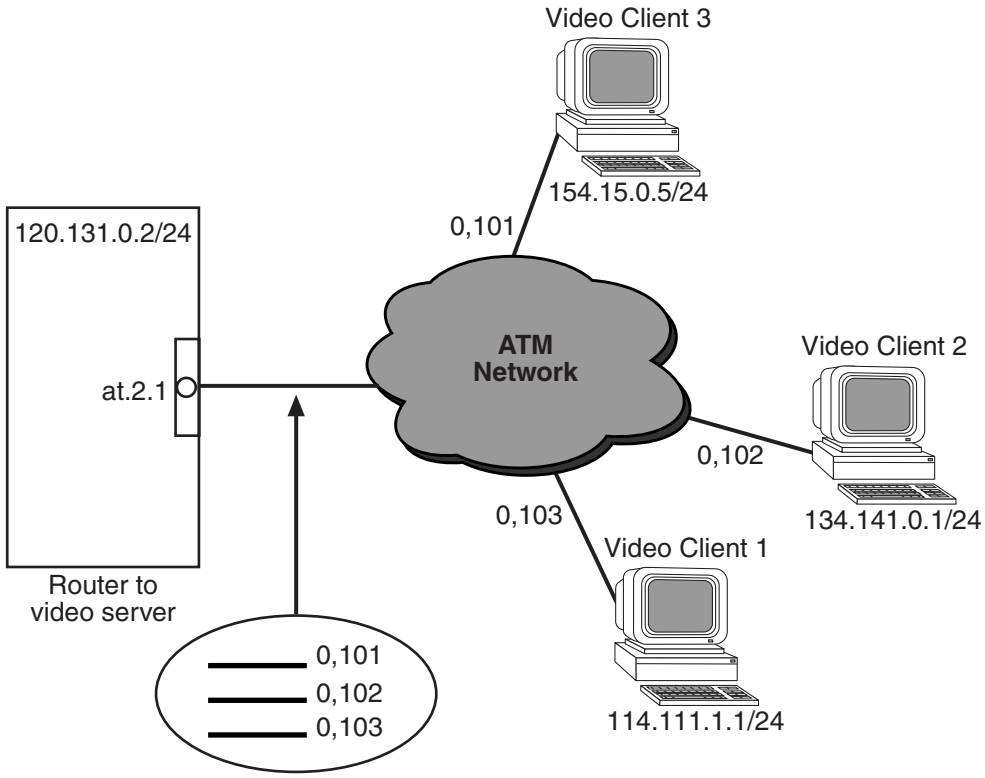
```
xp(config)# interface create ip vlanA address-netmask 11.1.100.1/24 port at.4.3.0.100
```

The following command line sets an interface name ‘vlanB’ and IP address 11.1.100.2/24 on ATM port at.4.3.0.101:

```
xp(config)# interface create ip vlanB address-netmask 11.1.100.2/24 port at.4.3.0.101
```

ATM Sample Configuration 4

Consider the following scenario:



Suppose you wish to set up a connection between a video server and 3 video clients. The video server will route data through the router to the 3 separate video clients (Video Client 1, Video Client 2, and Video Client 3). The requirement is to pass traffic to the video clients through 3 separate virtual channels. This will allow you to apply a unique service profile to each Client's connection.

To keep things simple, you prefer to connect all 3 virtual channels to a single interface on the router, instead of creating a separate interfaces for each VC.

You as the network administrator can accommodate these requirements by using the router's ability to configure multiple VCs on a single interface. The following steps lead you through the configuration process on the router side of the ATM cloud.

Create the Virtual Channels

The following command line creates a virtual channel with VPI=0 and VCI=101, connected to Video Client 1:

```
xp(config)# atm create vcl port at.2.1.0.101
```

The following command line creates a virtual channel with VPI=0 and VCI=102, connected to Video Client 2:

```
xp(config)# atm create vcl port at.2.1.0.102
```

The following command line creates a virtual channel with VPI=0 and VCI=103, connected to Video Client 3:

```
xp(config)# atm create vcl port at.2.1.0.103
```

Create a VLAN

The following command line creates a VLAN called ‘video’ with ID number 20 and which supports all protocols:

```
xp(config)# vlan create video ip id 20
```

Associate a Virtual Channel to a VLAN

The following command line associates VLAN ‘video’ to the virtual channel with VPI=0 and VCI=101:

```
xp(config)# vlan add ports at.2.1.0.101 to video
```

The following command line associates VLAN ‘video’ to the virtual channel with VPI=0 and VCI=102:

```
xp(config)# vlan add ports at.2.1.0.102 to video
```

The following command line associates VLAN ‘video’ to the virtual channel with VPI=0 and VCI=103:

```
xp(config)# vlan add ports at.2.1.0.103 to video
```

Configure an Interface on an ATM Port

The following command line sets the interface name for the VLAN as ‘atm-video’ and the IP address as 120.131.0.2/24:

```
xp(config)# interface create ip atm-video address-netmask 120.131.0.2/24 vlan video
```

Note: Enterasys recommends that you use alphabetic characters when defining interface names—purely numeric interfaces will be interpreted as IP addresses. The X-Pedition router displays interface names up to 32 characters in length.

Assign a Peer Address to Each Virtual Channel (for VC-mux Encapsulation Traffic)

This step applies only when you are transmitting VC-mux encapsulation traffic. In this specific case, you must configure a peer address to each virtual channel if you wish to associate a particular destination address to the VC. This will allow the router to route traffic to a specific client without multicasting to every virtual channel. You can dedicate a VC to handle traffic to a specific client.

The following command line associates destination address 114.111.1.1/24 (Video Client 1) to the virtual channel with VPI=0 and VCI=101:

```
xp(config)# atm set peer-addr port at.2.1.0.101 ip-address 114.111.1.1/24
```

The following command line associates destination address 134.141.0.1/24 (Video Client 2) to the virtual channel with VPI=0 and VCI=102:

```
xp(config)# atm set peer-addr port at.2.1.0.102 ip-address 134.141.0.1/24
```

The following command line associates destination address 154.15.0.5/24 (Video Client 3) to the virtual channel with VPI=0 and VCI=101:

```
xp(config)# atm set peer-addr port at.2.1.0.103 ip-address 154.15.0.5/24
```

Chapter 10

Packet-over-SONET Configuration Guide

This chapter explains how to configure and monitor packet-over-SONET (PoS) on the X-Pedition router. See the **sonet** commands section of the *Enterasys X-Pedition Command Line Interface Reference Manual* for a description of each command.

Overview

PoS requires installation of the OC-3c or OC-12c PoS line cards in an X-Pedition 8000 or 8600. The OC-3c line card has four PoS ports, while the OC-12c line card has two PoS ports. You must use the "so." prefix for PoS interface ports. For example, you would specify a PoS port located at router slot 13, port 1 as "so.13.1." By default, PoS ports are set for point-to-point protocol (PPP) encapsulation. You cannot change this encapsulation type for PoS ports.

Note: While PoS ports use PPP encapsulation, other PPP characteristics such as service profiles, encryption, compression, and MLP bundles are not supported for PoS ports.

Note: The X-Pedition router does *not* support PVST over POS. However, the router *will* support STP over POS.

By default, PoS ports are configured to receive a maximum transmission unit (MTU) size of 1500 octets. The actual MTU size used for transmissions over a PoS link is a result of PPP negotiation. For transmission of "jumbo frames" (MTUs up to 65442 octets), you can increase the MTU size of the PoS port. The MTU must be set at the port level.

Configuring IP Interfaces for PoS Links

Configuring IP interfaces for PoS links is generally the same as for WANs and for LANs. You assign an IP address to each interface and define routing mechanisms such as OSPF or RIP as with any IP network. You can configure the IP interface on the physical port or you can configure the interface as part of a VLAN for PoS links. You can also configure multiple IP addresses for each interface, as described in *Configuring IP Interfaces and Parameters* on page 166.

When creating the IP interface for a PoS link, you can either specify the peer address if it is known (*static* address), or allow the peer address to be automatically discovered via IPCP negotiation (*dynamic* address). If the peer address is specified, any address supplied by the peer during IPCP negotiation is ignored.

IP interfaces for PoS links can have primary and secondary IP addresses. The primary addresses may be either dynamic or static, but the secondary address must be static. This is because only the primary addresses of both the local and peer devices are exchanged during IP Control Protocol (IPCP) negotiation.

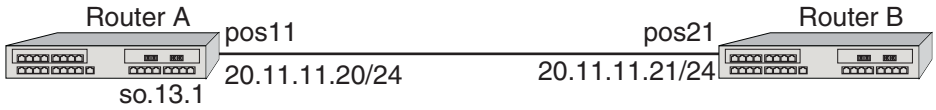
Source filtering and ACLs can be applied to an IP interface for a PoS link. Unlike WAN ports, the applied filter or ACL presents no limitation. Different filters can be configured on different PoS ports.

Note: You may not apply ACLs to interface EN0 of the control module.

Configuring Packet-over-SONET Links

To configure a packet-over-SONET link:

1. On the X-Pedition router, assign an interface to the PoS port to which you will connect via fiber cable in a point-to-point link. Assign an IP address and netmask to the interface. If possible, determine the peer address of the interface at the other end of the point-to-point link. In the following example, the port so.13.1 on the X-Pedition router will be associated with the interface pos11:



2. Create a point-to-point interface with the **interface create** command, specifying the IP address and netmask for the interface on the X-Pedition router and the peer address of the other end of the connection:

```
interface create ip pos11 address-netmask 20.11.11.20/24 peer-address 20.11.11.21 port so.13.1
```

   When you create the point-to-point interface as shown above, the X-Pedition router creates an implicit VLAN called "SYS_L3_<*interface-name*>." In the above example, the X-Pedition router creates the VLAN 'SYS_L3_pos11.'

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

3. If you want to increase the MTU size on a port, specify the parameter **mtu** with the '**port set**' command and define a value up to 65442 (octets). See *Configuring Jumbo Frames* on page 168 for more information.

4. Specify the bit error rate thresholds, if necessary. See *Specifying Bit Error Rate Thresholds* on page 150 for more information.

5. Modify any other PoS operating parameters, as needed. Table 7 lists the operating parameters that you can modify and the configuration commands that you use.

Table 7. PoS Optional Operating Parameters

Parameter	Default Value	Configuration Command
Framing	SONET	**sonet set** <*port*> **framing sdh|sonet**
Loopback	Disabled	**sonet set** <*port*> **loopback**
Path tracing	(none)	**sonet set** <*port*> **pathtrace**
Circuit identifier	(none)	**sonet set** <*port*> **circuit-id**
Frame Check Sequence	32-bit	**sonet set** <*port*> **fcs-16-bit**
Scrambling	Enabled	**sonet set** <*port*> **no-scramble**

Configuring Automatic Protection Switching

Automatic protection switching (APS) provides a mechanism to support redundant transmission circuits between SONET devices. The X-Pedition router supports the following APS features:

- Linear network topology. Ring topologies are not supported.
- 1+1 switching. Each working line is protected by one protecting line and the same signal is transmitted on both the working and protecting lines. The two transmission copies that arrive at the receiving end are compared, and the best copy is used. If there is a line failure or line degradation, the end node switches the connection over to the protecting line.

Note: In APS terminology, *bridge* means to transmit identical traffic on both the working and protecting lines, while *switch* means to select traffic from either the protecting line or the working line.

- Unidirectional switching, where one set of line terminating equipment (LTE) can switch the line independent of the other LTE. Bidirectional switching (where both sets of LTEs perform a coordinated switch) is not supported.
- Revertive switching. You can enable automatic switchover from the protecting line to the working line after the working line becomes available.

If the working circuit is disrupted or the bit error rates on the working circuit exceed the configured thresholds, traffic is automatically switched over to the protecting circuit. Any physical or logical characteristics configured for the working port are applied to the protecting port. This includes the IP address and netmask configured for the interface, spanning tree protocol (STP), per-VLAN spanning tree (PVST), etc.

Configuring Working and Protecting Ports

APS on the X-Pedition router requires configuration of a working port and a corresponding protecting port. You can configure any number of PoS ports. The limit is the number of PoS ports on the X-Pedition router. Any port on any module can be configured for APS. If one module should go down, the remaining ports on other modules will remain operational.

Note: The working and protecting ports must reside on the *same* X-Pedition router. You *cannot* configure APS operation for working and protecting ports on two *different* X-Pedition routers.

Note: PPP does not renegotiate when you hot swap a POS card (with APS enabled) or if either end of a connection goes down. To renegotiate PPP, the system must reboot at both ends.

The working port must:

- Be associated with a point-to-point IP interface that is configured with an IP address and netmask. See *Configuring Packet-over-SONET Links* on page 147 for the details on configuring the interface.

The protecting port must:

- Be in the default VLAN, where the default VLAN ID must be 1. The port may be either access or trunk but no other VLAN except VID 1 may be configured. The protecting port must *not* be configured for an interface.
- Not have any explicitly configured parameters. The protecting port inherits the configuration of the working port.

To configure a working and a protecting PoS port, enter the following command in Configure mode:

Configure working and protecting PoS ports.	**sonet set** <*working-port*> **protection 1+1 protected-by** <*protecting-port*>

To manage the working and protecting PoS interfaces, enter the following commands in Configure mode:

Prevent a working interface from switching to a protecting port. This command can only be applied to a port configured as a protecting port.	**sonet set** <*port*> **protection-switch lockoutprot**
Force a switch to the specified port. This command can be applied to either the working or protecting port.	**sonet set** <*port*> **protection-switch forced**
Manually switch the line to the specified port. This command can be applied to either the working or protecting port.	**sonet set** <*port*> **protection-switch manual**

Note: You can only specify one option, **lockoutprot**, **forced** or **manual**, for a port. Also, an option can be applied to *either* the working port or the protecting port, but not *both* working and protecting ports at the same time.

To return the circuit to the working interface after the working interface becomes available, enter the following commands in Configure mode:

Enable automatic switchover from the protecting interface to the working interface after the working interface becomes available. This command can only be applied to a protecting port.	**sonet set** <*port*> **revertive off| on**
Sets the number of minutes after the working interface becomes available that automatic switchover from the protecting interface to the working interface takes place. The default value is 5 minutes.	**sonet set** <*port*> **WTR-timer** <*minutes*>

Specifying Bit Error Rate Thresholds

If the bit error rate (BER) on the working line exceeds one of the configured thresholds, the receiver automatically switches over to the protecting line.

BER is calculated with the following:

BER = errored bits received/total bits received

The default BER thresholds are:

- Signal degrade BER threshold of 10^{-6} (1 out of 1,000,000 bits transmitted is in error). Signal degrade is associated with a “soft” failure. Signal degrade is determined when the BER exceeds the configured rate.
- Signal failure BER threshold of 10^{-3} (1 out of 1,000 bits transmitted is in error). Signal failure is associated with a “hard” failure. Signal fail is determined when any of the following conditions are detected: loss of signal (LOS), loss of frame (LOF), line alarm indication bridge and selector signal (AIS-L), or the BER threshold exceeds the configured rate.

To specify different BER thresholds, enter the following commands in Enable mode:

Specify signal degrade BER threshold.	**sonet set** <*port*> **sd-ber** <*number*>
Specify signal failure BER threshold.	**sonet set** <*port*> **sf-ber** <*number*>

Monitoring PoS Ports

To display PoS port configuration information, enter one of the following commands in Enable mode:

Show framing status, line type, and circuit ID of the optical link.	**sonet show medium** <*port list*>
Show working or protecting line, direction, and switch status.	**sonet show aps** <*port list*>
Show received path trace.	**sonet show pathtrace** <*port list*>
Show loopback status.	**sonet show loopback** <*port list*>

The following table describes additional monitoring commands for IP interfaces for PoS links, designed to be used in Enable mode:

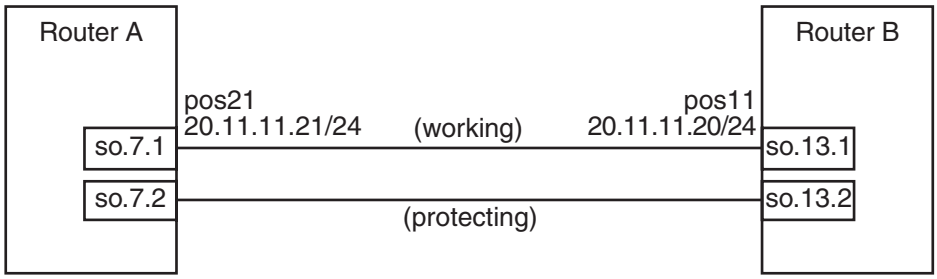
Display bridge NCP statistics for specified PoS port.	**ppp show stats port** <*port name*> **bridge-ncp**
Display IP NCP statistics for specified PoS port.	**ppp show stats port** <*port name*> **ip-ncp**
Display link-status statistics for specified PoS port.	**ppp show stats port** <*port name*> b
Display summary information for specified PoS port.	**ppp show stats port** <*port name*> **summary**

Example Configurations

This section shows example configurations for PoS links.

APS PoS Links Between X-Pedition Routers

The following example shows APS PoS links between two X-Pedition routers, A and B.



Note: PPP does not renegotiate when you hot swap a POS card (with APS enabled) or if either end of a connection goes down. To renegotiate PPP, the system must reboot at both ends.

The following is the configuration for router A:

```
interface create ip pos21 address-netmask 20.11.11.21/24 peer-address 20.11.11.20 type point-to-point port
so.7.1 sonet set so.7.1 protection 1+1 protected-by so.7.2
```

The following is the configuration for router B:

```
interface create ip pos11 address-netmask 20.11.11.20/24 peer-address 20.11.11.21 type point-to-point port
so.13.1 sonet set so.13.1 protection 1+1 protected-by so.13.2
```

PoS Link Between the X-Pedition Router and a Cisco Router

The following example shows a PoS link between an X-Pedition router, router A, and a Cisco 12000 series Gigabit Switch Router, router B. The MTU on both routers is configured for same size of 9216 octets.

The following is the configuration for router A:

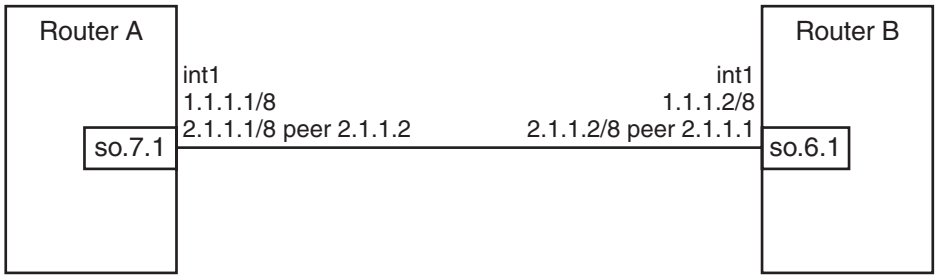
```
port set so.6.1 mtu 9216
interface create ip so-1 address-netmask 40.1.1.1/16 port so.6.1
```

The following is the configuration for router B:

```
interface POS1/0
 mtu 9216
 ip address 40.1.1.2 255.255.0.0
 no ip directed-broadcast
 encapsulation ppp
 crc 32
 pos scramble-atm
 pos flag c2 22
```

Bridging and Routing Traffic Over a PoS Link

The following example shows how to configure a VLAN ‘v1’ that includes the PoS ports on two connected X-Pedition routers, router A and router B. Bridged or routed traffic is transmitted over the PoS link.



The following is the configuration for router A:

```
port set so.7.1 mtu 65442
stp enable port so.7.1
vlan create v1 port-based id 10
vlan add ports so.7.1 to v1
interface create ip int1 address-netmask 1.1.1.1/8 vlan v1
interface add ip int1 address-netmask 2.1.1.1/8 peer-address 2.1.1.2
```

The following is the configuration for router B:

```
port set so.6.1 mtu 65442
stp enable port so.6.1
vlan create v1 port-based id 10
vlan add ports so.6.1 to v1
interface create ip int1 address-netmask 1.1.1.2/8 vlan v1
interface add ip int1 address-netmask 2.1.1.2/8 peer-address 2.1.1.1
```

Chapter 11

DHCP Configuration Guide

DHCP Overview

The Dynamic Host Configuration Protocol (DHCP) server on the X-Pedition router provides dynamic address assignment and configuration to DHCP capable end-user systems, such as Windows 95/98/NT and Apple Macintosh systems. You can configure the server to provide a dynamic IP address from a pre-allocated pool of IP addresses or a static IP address. You can also configure parameters for use by the clients, such as default gateway and network masks, and system-specific parameters, such as NetBIOS Name Server and NetBIOS node type of the client.

The amount of time that a particular IP address is valid for a system is called a *lease*. The X-Pedition router maintains a *lease database* which contains information about each assigned IP address, the MAC address to which it is assigned, the lease expiration, and whether the address assignment is dynamic or static. The DHCP lease database is stored in flash memory and can be backed up on a remote TFTP or RCP server. You can configure the intervals at which updates to the lease database (and backup) are done. Upon system reboot, the lease database will be loaded either from flash memory or from the TFTP or RCP server.

Note: Because the Command Line Interface does not validate the RCP and TFTP remote host addresses and passwords during DHCP configuration, you must ensure that you specify them correctly.

Note: The X-Pedition DHCP server is not designed to work as the primary DHCP server in an enterprise environment with hundreds or thousands of clients that are constantly seeking IP address assignment or reassignment. A standalone DHCP server with a redundant backup server may be more suitable for this enterprise environment.

Configuring DHCP

By default, the DHCP server is not enabled on the X-Pedition router. You can selectively enable DHCP service on particular interfaces and not others. To enable DHCP service on an interface, you must first define a DHCP *scope*. A scope consists of a pool of IP addresses and a set of parameters for a DHCP client. The parameters are used by the client to configure its network environment, for example, the default gateway and DNS domain name.

To configure DHCP on the X-Pedition router, you must configure an IP address pool, client parameters, and optional static IP address for a specified scope. Where several subnets are accessed through a single port, you can also define multiple scopes on the same interface and group the scopes together into a *superscope*.

Note: Although there is no imposed limit to the number of scopes you may define, the number of addresses allowed per scope depends on the amount of memory available. For example, systems with 128 MB of memory installed may include up to 253 addresses per scope—systems with more than 128 MB of memory may include up to 65,536 addresses per scope. However, systems with 64 MB of memory allow a maximum of 256 addresses—regardless of the number of scopes defined.

Configuring an IP Address Pool

To define a pool of IP addresses that the DHCP server can assign to a client, enter the following command in Configure mode:

Define pool of IP addresses to be used by clients.	**dhcp** <*scope*> **define pool** <*ip-range*>

Configuring Client Parameters

You can configure the client parameters shown in the table below.

Table 8. Client Parameters

Parameter	Value
address-mask	Address/netmask of the scope's subnet (This parameter is *required* and must be defined *before* any other client parameters are specified.)
broadcast	Broadcast address
bootfile	Client boot file name
dns-domain	DNS domain name
dns-server	IP address of DNS server
gateway	IP address of default gateway
lease-time	Amount of time the assigned IP address is valid for the system
netbios-name-server	IP address of NetBIOS Name Server (WINS server)
netbios-node-type	NetBIOS node type of the client
netbios-scope	NetBIOS scope of the client

Note: The DHCP server does not currently specify BootP server addresses to requesting clients.

To define the parameters that the DHCP server gives the clients, enter the following command in Configure mode:

Define client parameters.	**dhcp** <*scope*> **define parameters** <*parameter*> <*value*>...

Configuring a Static IP Address

To define a static IP address that the DHCP server can assign to a client with a specific MAC address, enter the following command in Configure mode:

Define static IP address for a particular MAC address.	**dhcp** <*scope*> **define static-ip** <*ipaddr*> **mac-address** <*macaddr*> [<*parameter*> <*value*>...]

Grouping Scopes with a Common Interface

You can apply several scopes to the same physical interface. For example, scopes can define address pools on different subnets that all are accessed through the same X-Pedition port. In this case, scopes that use the same interface must be grouped together into a "superscope." To attach a scope to a superscope, enter the following command in Configure mode:

Attach a scope to a superscope.	**dhcp** <*scope*> **attach superscope** <*name*>

Note: Although there is no imposed limit to the number of scopes you may define, the number of addresses allowed per scope depends on the amount of memory available. For example, systems with 128 MB of memory installed may include up to 253 addresses per scope—systems with more than 128 MB of memory may include up to 65,536 addresses per scope. However, systems with 64 MB of memory allow a maximum of 256 addresses—regardless of the number of scopes defined.

Note: The DHCP server does not process packets that arrive on PPP MLP interfaces.

Configuring DHCP Server Parameters

You can configure several "global" parameters that affect the behavior of the DHCP server itself. To configure global DHCP server parameters, enter the following commands in Configure mode:

Specify a remote location to back up the lease database.	**dhcp global set lease-database** <*url*>
Specify the intervals at which the lease database is updated.	**dhcp global set commit-interval** <*hours*>

Updating the Lease Database

After each client transaction, the DHCP server does not immediately update the information in the lease database. Lease update information is stored in flash memory and flushed to the database at certain intervals. You can use the **dhcp global set commit-interval** command to specify this interval; the default is one hour.

To force the DHCP server to immediately update its lease database, enter the following command in Enable mode:

Force the server to update its lease database.	**dhcp flush**

Monitoring the DHCP Server

To display information from the lease database:

Show lease database information.	**dhcp show binding [active|expired|static]**

To display the number of allocated bindings for the DHCP server and the maximum number allowed:

Show the number of allocated bindings for the DHCP server.	**dhcp show num-clients**

DHCP Configuration Examples

The following configuration describes DHCP configuration for a simple network with just one interface on which DHCP service is enabled to provide both dynamic and static IP addresses.

1. Create an IP VLAN called ‘client_vlan’.

```
vlan create client_vlan ip
```

2. Add all Fast Ethernet ports in the X-Pedition router to the VLAN ‘client_vlan’.

```
vlan add port et.*.* to client_vlan
```

3. Create an IP interface called ‘clients’ with the address 10.1.1.1 for the VLAN ‘client_vlan’.

```
interface create ip clients address-netmask 10.1.1.1./16 vlan client_vlan
```

4. Define DHCP network parameters for the scope ‘scope1’.

```
dhcp scope1 define parameters address-netmask 10.1.0.0/16 gateway 10.1.1.1 lease-time 24 dns-
    domain acme.com dns-server 10.2.45.67 netbios-name-server 10.1.55.60
```

5. Define an IP address pool for addresses 10.1.1.10 through 10.1.1.20.

```
dhcp scope1 define pool 10.1.1.10-10.1.1.20
```

6. Define another IP address pool for addresses 10.1.1.40 through 10.1.1.50.

```
dhcp scope1 define pool 10.1.1.40-10.1.1.50
```

7. Define a static IP address for 10.1.7.5.

```
dhcp scope1 define static-ip 10.1.7.5 mac-address 08:00:20:11:22:33
```

8. Define another static IP address for 10.1.7.7. and give it a specific gateway address of 10.1.1.2.

```
dhcp scope1 define static-ip 10.1.7.7 mac-address 08:00:20:aa:bb:cc:dd gateway 10.1.1.2
```

9. Specify a remote lease database on the TFTP server 10.1.89.88.

```
dhcp global set lease-database tftp://10.1.89.88/lease.db
```

10. Specify a database update interval of every 15 minutes.

```
dhcp global set commit-interval 15
```

Configuring Secondary Subnets

In some network environments, multiple logical subnets can be imposed on a single physical segment. These logical subnets are sometimes referred to as “secondary subnets” or “secondary networks.” For these environments, the DHCP server may need to give out addresses on different subnets. The DNS server, DNS domain, and WINS server may be the same for clients on different secondary subnets, however, the default gateway will most likely be different since it must be a router on the client’s local subnet.

The following example shows a simple configuration to support secondary subnets 10.1.x.x and 10.2.x.x.

1. Define the network parameters for ‘scope1’ with the default gateway 10.1.1.1.

```
dhcp scope1 define parameters address-netmask 10.1.0.0/16 gateway 10.1.1.1 dns-domain
    acme.com dns-server 10.1.44.55
```

2. Define the address pool for ‘scope1’.

```
dhcp scope1 define pool 10.1.1.10-10.1.1.20
```

3. Define the network parameters for ‘scope2’ with the default gateway 10.2.1.1.

```
dhcp scope2 define parameters address-netmask 10.2.0.0/16 gateway 10.2.1.1 dns-domain
    acme.com dns-server 10.1.77.88
```

4. Define the address pool for ‘scope2’.

```
dhcp scope2 define pool 10.2.1.40-10.2.1.50
```

5. Create a superscope ‘super1’ that includes ‘scope1’.

```
dhcp scope1 attach superscope super1
```

6. Include ‘scope2’ in the superscope ‘super1’.

```
dhcp scope2 attach superscope super1
```

Since there are multiple pools of IP addresses, the pool associated with ‘scope1’ is used first since ‘scope1’ is applied to the interface before ‘scope2’. Clients that are given an address from ‘scope1’ will also be given parameters from ‘scope1,’ which includes the default gateway 10.1.1.1 that resides on the 10.1.x.x subnet. When all the addresses for ‘scope1’ are assigned, the server will start giving out addresses from ‘scope2’ which will include the default gateway parameter 10.2.1.1 on subnet 10.2.x.x.

Secondary Subnets and Directly-Connected Clients

A directly-connected client is a system that resides on the same physical network as the DHCP server and does not have to go through a router or relay agent to communicate with the server. If you configure the DHCP server on the X-Pedition router to service directly-connected clients on a secondary subnet, you must configure the secondary subnet using the **interface add ip** command. The **interface add ip** command configures a secondary address for an interface that was previously created with the **interface create ip** command.

The following example shows a simple configuration to support directly-connected clients on a secondary subnet.

1. Create an interface ‘clients’ with the primary address 10.1.1.1.

```
interface create ip clients address-mask 10.1.1.1/16 port et.1.1
```

2. Assign a secondary address 10.2.1.1 to the interface ‘clients’.

```
interface add ip clients address-mask 10.2.1.1/16
```

3. Define the network parameters for ‘scope1’ with the default gateway 10.1.1.1.

```
dhcp scope1 define parameters address-netmask 10.1.0.0/16 gateway 10.1.1.1 dns-domain
      acme.com dns-server 10.1.44.55
```

4. Define the address pool for ‘scope1’.

```
dhcp scope1 define pool 10.1.1.10-10.1.1.20
```

5. Define the network parameters for ‘scope2’ with the default gateway 10.2.1.1.

```
dhcp scope2 define parameters address-netmask 10.2.0.0/16 gateway 10.2.1.1 dns-domain
    acme.com dns-server 10.1.77.88
```

6. Define the address pool for ‘scope2’.

```
dhcp scope2 define pool 10.2.1.40-10.2.1.50
```

7. Create a superscope ‘super1’ that includes ‘scope1’.

```
dhcp scope1 attach superscope super1
```

8. Include ‘scope2’ in the superscope ‘super1’.

```
dhcp scope2 attach superscope super1
```

For clients on the secondary subnet, the default gateway is 10.2.1.1, which is also the secondary address for the interface ‘clients’.

Interacting with Relay Agents

For clients that are not directly connected to the DHCP server, a relay agent (typically a router) is needed to communicate between the client and the server. The relay agent is usually only needed during the initial leasing of an IP address. Once the client obtains an IP address and can connect to the network, the renewal of the lease is performed between the client and server without the help of the relay agent.

The default gateway for the client must be capable of reaching the X-Pedition router’s DHCP server. The X-Pedition router must also be capable of reaching the client’s network. The route must be configured (with static routes, for example) or learned (with RIP or OSPF, for example) so that the DHCP server can reach the client.

The following example shows a simple configuration to support clients across a relay agent.

1. Create an interface ‘clients’ with the primary address 10.1.1.1.

```
interface create ip clients address-mask 10.1.1.1/16 port et.3.3
```

2. Define a static route to the 10.5.x.x. subnet using the gateway 10.1.7.10 which tells the DHCP server how to send packets to the client on the 10.5.x.x subnet.

```
ip add route 10.5.0.0/16 gateway 10.1.7.10
```

3. Define the network parameters for ‘scope1’ with the default gateway 10.5.1.1 (the relay agent for the client).

```
dhcp scope1 define parameters address-netmask 10.5.0.0/16 gateway 10.5.1.1 dns-domain
    acme.com
```

4. Define the address pool for ‘scope1’.

```
dhcp scope1 define pool 10.5.1.10-10.5.1.20
```

Chapter 12

IP Routing Configuration Guide

The X-Pedition router supports standards-based TCP, UDP, and IP. This chapter describes how to configure IPv4 interfaces and general non-protocol-specific routing parameters.

Note: Refer to *IPv6 Configuration Guide* on page 179 for information on configuring IPv6 interfaces.

IP Routing Protocols

The X-Pedition router supports standards-based unicast and multicast routing. Unicast routing protocol support includes Interior Gateway Protocols and Exterior Gateway Protocols. Multicast routing protocols are used to determine how multicast data is transferred in a routed environment.

Unicast Routing Protocols

Interior Gateway Protocols are used for routing networks that are within an "autonomous system," a network of relatively limited size. All IP interior gateway protocols must be specified with a list of associated networks before routing activities can begin. A routing process listens to updates from other routers on these networks and broadcasts its own routing information on those same networks. The X-Pedition router supports the following Interior Gateway Protocols:

- Routing Information Protocol (RIP) Version 1, 2 (RFC 1058, 1723). Configuring RIP for the X-Pedition router is described in Chapter 15.
- Routing Information Protocol next generation (RIPng) (RFP 2080). Configuring RIPng for the X-Pedition router is described in Chapter 16.
- Open Shortest Path First (OSPF) Version 2 (RFC 2328). Configuring OSPF for the X-Pedition router is described in Chapter 17.

Exterior Gateway Protocols are used to transfer information between different “autonomous systems.” The X-Pedition router supports the following Exterior Gateway Protocol:

- Border Gateway Protocol (BGP) Version 4 (RFC 1771). Configuring BGP for the X-Pedition router is described in Chapter 18.

  Note: The X-Pedition router does not currently follow "Breaking Ties (Phase2)," Section 9.1.2.1 (p. 37-38) of RFC 1771. Instead, the router follows "Breaking Ties (Phase2)," Section 9.1.2.2 (p. 49-50) of Draft-ietf-ier-bgp-4-17.

Multicast Routing Protocols

IP multicasting allows a host to send traffic to a subset of all hosts. These hosts subscribe to group membership, thus notifying the X-Pedition router of participation in a multicast transmission.

Multicast routing protocols are used to determine which routers have directly attached hosts, as specified by IGMP, that have membership to a multicast session. Once host memberships are determined, routers use multicast routing protocols, such as DVMRP, to forward multicast traffic between routers.

The X-Pedition router supports the following multicast routing protocols:

- Distance Vector Multicast Routing Protocol (DVMRP) RFC 1075
- Internet Group Management Protocol (IGMP) as described in RFC 2236

The X-Pedition router supports the latest DVMRP Version 3.0 draft specification, which includes Generation ID and Pruning/Grafting. Configuring multicast routing for the X-Pedition router is described in Chapter 20.

Configuring IP Interfaces and Parameters

You can configure an IP interface to a single port or to a VLAN. This section provides an overview of configuring IP interfaces.

Interfaces on the X-Pedition router are logical interfaces. Therefore, you can associate an interface with a single port or with multiple ports:

- To associate an interface with a single port, use the **port** option with the **interface create** command.
- To associate an interface with multiple ports, first create an IP VLAN and add ports to it, then use the vlan option with the **interface create** command.

Note: Users cannot assign IPv6 interfaces to a VLAN or multiple ports. Refer to Chapter 13, *IPv6 Configuration Guide* for information on configuring IPv6 interfaces.

The **interface create ip** command creates and configures an IP interface. Configuration of an IP interface can include information such as the interface's name, IP address, netmask, broadcast address, and so on. You can also create an interface in a disabled (**down**) state instead of the default enabled (**up**) state.

Note: You must use either the **port** option or the **vlan** option with the **interface create** command. The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Configuring IP Interfaces to Ports

You can configure an IP interface directly to a physical port. Each port can be assigned multiple IP addresses representing multiple subnets connected to the physical port. For example, to assign an IP interface 'RED' to physical port et.3.4, enter the following:

```
xp(config)# interface create ip RED address-netmask 10.50.0.0/255.255.0.0 port et.3.4
```

To configure a secondary address of 10.23.4.36 with a 24-bit netmask (255.255.255.0) on the IP interface int4:

```
xp(config)# interface add ip int4 address-mask 10.23.4.36/24
```

Configuring IP Interfaces for a VLAN

You can configure one IP interface per VLAN. Once an IP interface has been assigned to a VLAN, you can add a secondary IP address to the VLAN. To create a VLAN called IP3, add ports et.3.1 through et.3.4 to the VLAN, then create an IP interface on the VLAN:

```
xp(config)# vlan create IP3 ip

xp(config)# vlan add ports et.3.1-4 to IP3

xp(config)# interface create ip int3 address-mask 10.20.3.42/24 vlan IP3
```

To configure a secondary address of 10.23.4.36 with a 24-bit netmask (255.255.255.0) on the IP interface int4:

```
xp(config)# interface add ip int3 address-mask 10.23.4.36/24 vlan IP3
```

Specifying Ethernet Encapsulation Method

The Enterasys X-Pedition router supports two encapsulation types for IP. Use the **interface create ip** command to configure one of the following encapsulation types on a per-interface basis:

- Ethernet II: The standard ARPA Ethernet Version 2.0 encapsulation, which uses a 16-bit protocol type code (the default encapsulation method).
- 802.3 SNAP: SNAP IEEE 802.3 encapsulation, in which the type code becomes the frame length for the IEEE 802.2 LLC encapsulation (destination and source Service Access Points, and a control byte).

Note: When using line cards introduced prior to the "AA" series, SNA/DLC/NetBIOS traffic may not bridge properly. The issue in bridging DLC packets occurs where the length field within an IEEE 802.3 frame indicates less than 46 bytes of data.

The X-Pedition router removes the length field information of incoming IEEE 802.3, 802.2, and Ethernet SNAP packets, then recalculates the field prior to re-transmission. Consequently, the calculation is based on the length of the entire data field. A packet entering the X-Pedition router whose length field indicates a data field of less than 46 bytes will exit with the length field recalculated incorrectly. This can be a problem with LLC2 and legacy IPX applications. Typically, such packets exist only in SNA and NetBIOS/NetBEUI environments.

Configuring Jumbo Frames

Certain X-Pedition router line cards support jumbo frames (frames larger than the standard Ethernet frame size of 1518 bytes).

To transmit frames of up to 65442 octets, you increase the maximum transmission unit (MTU) size from the default of 1500. You must set the MTU at the port level with the **port set mtu** command. You can also set the MTU at the IP interface level; if you set the MTU at the IP interface level, the MTU size must be less than the size configured for each port in the interface. Note that the interface MTU only determines the size of the packets that are forwarded in software.

In the following example, the ports gi.3.1 through gi.3.8 are configured with an MTU size of 65442 octets. Ports gi.3.1 through gi.3.4 are configured to be part of the interface 'int3,' with an MTU size of 65000 octets.

```
xp(config)# port set gi.3.1-8 mtu 65442

xp(config)# vlan create JUMBO1 ip

xp(config)# vlan add ports gi.3.1-4 to JUMBO1

xp(config)# interface create ip int3 mtu 50000 address-mask 10.20.3.42/24 vlan JUMBO1
```

If you do *not* set the MTU at the interface level, the actual MTU of the interface is the lowest MTU configured for a port in the interface. In the following example, port gi.3.1 is configured with an MTU size of 50022 octets while ports gi.3.2-8 are configured with an MTU size of 65442 octets. The interface MTU will be 50000 octets (50022 octets minus 22 octets of link layer overhead).

```
xp(config)# port set gi.3.1 mtu 50022

xp(config)# port set gi.3.2-8 mtu 65442

xp(config)# vlan create JUMBO1 ip

xp(config)# vlan add ports gi.3.1-4 to JUMBO1

xp(config)# interface create ip int3 address-mask 10.20.3.42/24 vlan JUMBO1
```

Configuring Address Resolution Protocol (ARP)

The X-Pedition router allows you to configure Address Resolution Protocol (ARP) table entries and parameters. ARP is used to associate IP addresses with media or MAC addresses. Taking an IP address as input, ARP determines the associated MAC address. Once a media or MAC address is determined, the IP address/media address association is stored in an ARP cache for rapid retrieval. Then the IP datagram is encapsulated in a link-layer frame and sent over the network.

Configuring ARP Cache Entries

To create an ARP entry for the IP address 10.8.1.2 at port et.4.7 for 15 seconds:

```
xp# arp add 10.8.1.2 mac-addr 08:00:20:a2:f3:49 exit-port et.4.7 keep-time 15
```

To create a permanent ARP entry for the host *nfs2* at port et.3.1:

```
xp(config)# arp add nfs2 mac-addr 080020:13a09f exit-port et.3.1
```

To create a permanent ARP entry for IP address 10.8.1.25 that will always flood the traffic out the subnet:

```
xp(config)# arp add 10.8.1.25 mac-addr 080020:a2f360 vlan
```

To remove the ARP entry for the host 10.8.1.2 from the ARP table:.

```
xp# arp clear 10.8.1.2
```

To clear the entire ARP table.

```
xp# arp clear all
```

If the Startup configuration file contains **arp add** commands, the Control Module re-adds the ARP entries even if you have cleared them using the **arp clear** command. To permanently remove an ARP entry, use the **negate** command or **no** command to remove the entry.

Unresolved MAC Addresses for ARP Entries

When the X-Pedition router receives a packet for a host whose MAC address it has not resolved, the X-Pedition router tries to resolve the next-hop MAC address by sending ARP requests. Five requests are sent initially for each host, one every second.

You can configure the router to drop packets for hosts whose MAC addresses the router has been unable to resolve. To enable dropping of packets for hosts with unresolved MAC addresses:

```
xp# arp set drop-unresolved enabled
```

When you enable packets to be dropped for hosts with unresolved MAC addresses, the X-Pedition router will still attempt to periodically resolve these MAC addresses. By default, the X-Pedition router sends ARP requests at 30-second intervals to try to resolve up to 50 dropped entries.

To change the interval for sending ARP requests for unresolved entries to 45 seconds:

```
xp# arp set unresolve-timer 45
```

To change the number of unresolved entries that the X-Pedition router attempts to resolve to 75:

```
xp# arp set unresolve-threshold 75
```

Local Proxy ARP

Local Proxy ARP is used to help hosts with no knowledge of routing to reach other subnets. When you enable proxy arp, the X-Pedition router will process each ARP request it receives and check to ensure that the requested address is within the subnet making the request. If the address request is not targeted to the subnet of the interface that received the request, the X-Pedition router will respond to the ARP request with its own MAC address, regardless of whether or not it is the owner of the IP address being requested—otherwise, normal forwarding will occur. Proxy ARP will not respond to an ARP request that originates from an IP address not contained in the subnet of the interface receiving the request. By default, the X-Pedition router acts as a proxy for ARP requests with destination addresses of hosts to which the X-Pedition router can route traffic.

Note: Enable this feature only after carefully considering its implications on network design. For more information on the implications of this feature, please see ip enable local-proxy-arp interface in the *Enterasys X-Pedition Command Line Interface Reference Manual*.

The following command enables local proxy ARP on a specific interface:

```
xp(config)# ip enable local-proxy-arp interface 10.136.64.5
```

Configuring Reverse Address Resolution Protocol (RARP)

Reverse Address Resolution Protocol (RARP) works exactly the opposite of ARP. Taking a MAC address as input, RARP determines the associated IP address. RARP is useful for X-terminals and diskless workstations that may not have an IP address when they boot. They can submit their MAC address to a RARP server on the X-Pedition router, which returns an IP address.

Configuring RARP on the X-Pedition router consists of two steps:

1. Letting the X-Pedition router know which IP interfaces to respond to
2. Defining the mappings of MAC addresses to IP addresses

Specifying IP Interfaces for RARP

The **rarpd set interface** command allows you to specify which interfaces the X-Pedition router's RARP server responds to when sent RARP requests. You can specify individual interfaces or all interfaces. To cause the X-Pedition router's RARP server to respond to RARP requests from interface int1:

```
xp(config)# rarpd set interface int1
```

Defining MAC-to-IP Address Mappings

The **rarpd add** command allows you to map a MAC address to an IP address for use with RARP. When a host makes a RARP request on the X-Pedition router, and its MAC address has been mapped to an IP address with the **rarp add** command, the RARP server on the X-Pedition router responds with the IP address that corresponds to the host's MAC address. To map MAC address 00:C0:4F:65:18:E0 to IP address 10.10.10.10:

```
xp(config)# rarpd add hardware-address 00:C0:4F:65:18:E0 ip-address 10.10.10.10
```

There is no limit to the number of address mappings you can configure.

Optionally, you can create a list of mappings with a text editor and then use TFTP to upload the text file to the X-Pedition router. The format of the text file must be as follows:

```
MAC-address1 IP-address1
MAC-address2 IP-address2
...
MAC-addressn IP-addressn
```

Then place the text file on a TFTP server that the X-Pedition router can access and enter the following command in Enable mode:

```
xp# copy tftp-server to ethers
TFTP server? <IPaddr-of-TFTP-server>
Source filename? <filename>
```

Monitoring RARP

You can use the following commands to obtain information about the X-Pedition router's RARP configuration:

Display the interfaces to which the RARP server responds.	**rarpd show interface**
Display the existing MAC-to-IP address mappings	**rarpd show mappings**
Display RARP statistics.	**statistics show rarp** <*InterfaceName*>|**all**

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Configuring DNS Parameters

The X-Pedition router can be configured to specify DNS servers, which supply name services for DNS requests. You can specify up to three DNS servers.

To configure three DNS servers and configure the X-Pedition router's DNS domain name to "mrb.com":

```
xp(config)# system set dns server "10.1.2.3 10.2.10.12 10.3.4.5" domain mrb.com
```

Configuring IP Services (ICMP)

The X-Pedition router provides ICMP message capabilities including ping and traceroute. The **ping** command allows you to determine the reachability of a certain IP host, while the **traceroute** command allows you to trace the IP gateways to an IP host.

Configuring IP Helper

The **ip helper-address** command allows the user to forward specific UDP broadcast from one interface to another. Typically, broadcast packets from one interface are not forwarded (routed) to another interface. However, some applications use UDP broadcast to detect the availability of a service. Other services, for example BOOTP/DHCP require broadcast packets to be routed so that they can provide services to clients on another subnet. An IP helper can be configured on each interface to have UDP broadcast packets forwarded to a specific host for a specific service or forwarded to all other interfaces.

You can configure the X-Pedition router to forward UDP broadcast packets received on a given interface to all other interfaces or to a specified IP address. You can specify a UDP port number for which UDP broadcast packets with that destination port number will be forwarded. By default, if no UDP port number is specified, the X-Pedition router will forward UDP broadcast packets for the following services:

- BOOTP/DHCP (port 67 and 68)
- DNS (port 53)
- NetBIOS Name Server (port 137)
- NetBIOS Datagram Server (port 138)
- Time Service (port 37)

To forward UDP broadcast packets received on interface int1 to the host 10.1.4.5 for the six default UDP services:

```
xp(config)# ip helper-address interface int1 10.1.4.5
```

To forward UDP broadcast packets received on interface int2 to the host 10.2.48.8 for packets with the destination port 111 (port mapper):

```
xp(config)# ip helper-address interface int2 10.2.48.8 111
```

To forward UDP broadcast packets received on interface int3 to all other interfaces:

```
xp(config)# ip helper-address interface int3 all-interfaces
```

Configuring Direct Broadcast

Directed broadcast packets are network or subnet broadcast packets which are sent to a router to be forwarded as broadcast packets. They can be misused to create Denial Of Service attacks. The X-Pedition router protects against this possibility by ***not*** forwarding directed broadcasts, by default. To enable the forwarding of directed broadcasts, use the **ip enable directed-broadcast** command.

You can configure the X-Pedition router to forward all directed broadcast traffic from the local subnet to a specified IP address or all associated IP addresses. This is a more efficient method than defining only one local interface and remote IP address destination at a time with the **ip-helper** command when you are forwarding traffic from more than one interface in the local subnet to a remote destination IP address.

To enable directed broadcast forwarding on the “int4” network interface:

```
xp(config)# ip enable directed-broadcast interface int4
```

Configuring Denial of Service (DoS)

By default, the X-Pedition router installs flows in the hardware so that packets sent as directed broadcasts are dropped in hardware, if directed broadcast is not enabled on the interface where the packet is received. You can disable this feature, causing directed broadcast packets to be processed on the X-Pedition router even if directed broadcast is not enabled on the interface receiving the packet.

Similarly, the X-Pedition router installs flows to drop packets destined for the X-Pedition router for which service is not provided by the X-Pedition router. This prevents packets for unknown services from slowing the CPU. You can disable this behavior, causing these packets to be processed by the CPU.

To cause directed broadcast packets to be processed on the X-Pedition router, even if directed broadcast is not enabled on the interface receiving the packet:

```
xp(config)# ip dos disable directed-broadcast-protection
```

To allow packets destined for the X-Pedition router, but do not have a service defined for them on the X-Pedition router, to be processed by the X-Pedition router’s CPU:

```
xp(config)# ip dos disable port-attack-protection
```

Monitoring IP Parameters

The X-Pedition router provides display of IP statistics and configurations contained in the routing table. Information displayed provides routing and performance information. The **ip show** commands display IP information, such as routing tables, TCP/UDP connections, and IP interface configuration, on the router. The following example displays all established connections and services of the X-Pedition router.

```
xp# ip show connections
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        (state)
tcp        0      0  *:gated-gii           *:*                    LISTEN
tcp        0      0  *:http                *:*                    LISTEN
tcp        0      0  *:telnet              *:*                    LISTEN
udp        0      0  127.0.0.1:1025        127.0.0.1:162
udp        0      0  *:snmp                *:*
udp        0      0  *:snmp-trap           *:*
udp        0      0  *:bootp-relay         *:*
udp        0      0  *:route               *:*
udp        0      0  *:*                   *:*
```

The following example displays the contents of the routing table. It shows that some of the route entries are for locally connected interfaces (“directly connected”), while some of the other routes are learned from RIP.

```
xp# ip show routes
Destination          Gateway              Owner       Netif
-----------          -------              -----       -----
10.1.0.0/16          50.1.1.2             RIP         to-linux2
10.2.0.0/16          50.1.1.2             RIP         to-linux2
10.3.0.0/16          50.1.1.2             RIP         to-linux2
10.4.0.0/16          50.1.1.2             RIP         to-linux2
14.3.2.1             61.1.4.32            Static       int61
21.0.0.0/8           50.1.1.2             RIP         to-linux2
30.1.0.0/16          directly connected   -           to-goya
50.1.0.0/16          directly connected   -           to-linux2
61.1.0.0/16          directly connected   -           int61
62.1.0.0/16          50.1.1.2             RIP         to-linux2
68.1.0.0/16          directly connected   -           int68
69.1.0.0/16          50.1.1.2             RIP         to-linux2
127.0.0.0/8          127.0.0.1            Static      lo
127.0.0.1            127.0.0.1            -           lo
210.11.99.0/24       directly connected   -           int41
```

To display additional IP information, enter the following command in Enable mode:

Show ARP table entries.	**arp show all**
Show IP interface configuration.	**interface show ip**
Show DNS parameters.	**system show dns**

Configuring Router Discovery

The router discovery server on the X-Pedition router periodically sends out router advertisements to announce the existence of the X-Pedition router to other hosts. The router advertisements are multicast or broadcast to each interface on the X-Pedition router on which it is enabled and contain a list of the addresses on the interface and the preference of each address for use as a default route for the interface. A host can also send a router solicitation, to which the router discovery server on the X-Pedition router will respond with a unicast router advertisement.

On systems that support IP multicasting, router advertisements are sent to the ‘all-hosts’ multicast address 224.0.0.1 by default. You can specify that broadcast be used, even if IP multicasting is available. When router advertisements are sent to the all-hosts multicast address or an interface is configured for the limited broadcast address 255.255.255.255, the router advertisement includes all IP addresses configured on the physical interface. When router advertisements are sent to a net or subnet broadcast, then only the address associated with the net or subnet is included.

To start router discovery on the router, enter the following command in Configure mode:

```
xp(config)# rdisc start
```

The **rdisc start** command lets you start router discovery on the X-Pedition router. When router discovery is started, the X-Pedition router multicasts or broadcasts periodic router advertisements on each configured interface. The router advertisements contain a list of addresses on a given interface and the preference of each address for use as the default route on the interface. By default, router discovery is disabled.

The **rdisc add address** command lets you define addresses to be included in router advertisements. If you configure this command, only the specified hostname(s) or IP address(es) are included in the router advertisements. For example:

```
xp(config)# rdisc add address 10.10.5.254
```

By default, all addresses on the interface are included in router advertisements sent by the X-Pedition router.

If you want to have only specific addresses included in router advertisements, use the **rdisc add address** command to specify those addresses.

The **rdisc set address** command lets you specify the type of router advertisement in which the address is included and the preference of the address for use as a default route. For example, to specify that an address be included only in broadcast router advertisements and that the address is ineligible to be a default route:

```
xp#(config) rdisc set address 10.20.36.0 type broadcast preference ineligible
```

The **rdisc set interface** command lets you specify the intervals between the sending of router advertisements and the lifetime of addresses sent in a router advertisement. To specify the maximum time between the sending of router advertisements on an interface:

```
xp#(config) rdisc set interface xp4 max-adv-interval 1200
```

To display router discovery information:

```
xp# rdisc show all


Task State: <Foreground NoResolv NoDetach> (1)

   Send buffer size 2048 at 812C68F8
   Recv buffer size 2048 at 812C60D0

Timers:

   RouterDiscoveryServer Priority 30

      RouterDiscoveryServer_XP2_XP3_IP <OneShot>
        last: 10:17:21 next: 10:25:05 (2)

Task RouterDiscoveryServer:
   Interfaces:
      Interface XP2_XP3_IP: (3)
         Group 224.0.0.1: (4)
            minadvint 7:30 maxadvint 10:00 lifetime 30:00 (5)

            Address 10.10.5.254: Preference: 0 (6)

   Interface policy:
      Interface XP2_XP3_IP* MaxAdvInt 10:00 (7)
```

Legend:

1. Information about the RDISC task.
2. Shows when the last router advertisement was sent and when the next advertisement will be sent.
3. The interface on which router advertisement is enabled.
4. Multicast address.
5. Current values for the intervals between the sending of router advertisements and the lifetime of addresses sent in a router advertisement.
6. IP address that is included in router advertisement. The preference of this address as a default route is 0, the default value.
7. Shows configured values for the specified interface.

Configuration Examples

Assigning IP/IPX Interfaces

To enable routing on the X-Pedition router, you must assign an IP or IPX interface to a VLAN. To assign an IP or IPX interface named ‘RED’ to the ‘BLUE’ VLAN, enter the following command:

```
xp(config)# interface create ip RED address-netmask 10.50.0.1/255.255.0.0 vlan BLUE
```

You can also assign an IP or IPX interface directly to a physical port.

Note: The X-Pedition router supports a maximum of 64 IPX interfaces. Enterasys recommends that you use alphabetic characters when defining interface names—purely numeric interfaces will be interpreted as IP addresses. The X-Pedition router displays interface names up to 32 characters in length.

Chapter 13

IPv6 Configuration Guide

This chapter provides a table of the RFCs relevant to the implementation of IPv6 on the X-Pedition router, followed by a discussion of several implementation-specific issues. The remainder of the chapter provides IPv6 configuration procedures. This discussion assumes familiarity with basic IPv6 concepts, as described in the RFC documents.

For information about configuring RIPng, the IPv6 version of RIP, refer to *RIPng Configuration Guide* on page 227.

IPv6 RFC Overview

IPv6 is a major upgrade to IPv4, the version of the Internet Protocol that is currently used in the Internet and in corporate and private intranets. The IPv6 protocol suite offers a number of new features and functionality in addition to solving the IPv4 problem of address space limitations.

The following table lists the RFCs relevant to the X-Pedition IPv6 implementation, with brief notations about how the X-Pedition router conforms to the RFC requirements. See *IPv6 Implementation Notes* on page 182 for more specific information about this implementation.

You can access complete text of these RFCs at the following URL:

http://www/ietf.org/rfc/rfc<*number*>.txt

where <*number*> is the number of the RFC.

RFC 2375	*IPv6 Multicast Address Assignments*	Defines the initial assignment of IPv6 multicast addresses. All supported multicast addresses are conformant with this RFC.

RFC 2460	*Internet Protocol, Version 6 (IPv6) Specification* Specifies version 6 of the Internet Protocol. This RFC specifies the basic IPv6 header and the initially defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols. The X-Pedition router will forward an IPv6 packet that includes an IPv6 Extension header without processing the Extension header itself, unless the packet destination is the router itself.
RFC 2461	*Neighbor Discovery for IP Version 6 (IPv6)* Specifies the Neighbor Discovery protocol for IP Version 6, including Router and Prefix discovery, Address Resolution and Neighbor Unreachability Detection, Redirects, and the ICMP packets used by the protocol. Fully conformant with requirements identified in the RFC for routers.
RFC 2462	*IPv6 Stateless Address Autoconfiguration* Specifies the steps a host takes in deciding how to autoconfigure its interfaces in IP version 6. Defines the process for generating a link-local address, the process for generating site-local and global addresses via stateless address autoconfiguration, and the Duplicate Address Detection procedure. Fully conformant with requirements identified in the RFC for routers.
RFC 2463	*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. Specifies the Internet Control Message Protocol (ICMP) Error and Informational Messages for use with version 6 of the Internet Protocol (IPv6). Fully conformant with requirements identified in the RFC for routers.
RFC 2464	*Transmission of IPv6 Packets over Ethernet Networks.* Specifies the frame format for transmission of IPv6 packets and the method of forming IPv6 link-local addresses and statelessly autoconfigured addresses on Ethernet networks. Also specifies the content of the Source/Target Link-layer Address option used in Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement, and Redirect messages when those messages are transmitted on an Ethernet. Fully conformant.

RFC 2465	*Management Information Base for IP Version 6: Textual Conventions and General Group.* Defines a portion of the Management Information Base (MIB) for use with network management protocols in the IPv6-based internets. Specifically, provides the IPv6 MIB textual conventions as well as the IPv6 MIB General group. Provides a basis for the IPv6 statistics reported by the X-Pedition router.
RFC 2466	*Management Information Base for IP Version 6: ICMPv6 Group.* Defines a portion of the Management Information Base (MIB) for use with network management protocols in the IPv6-based internets. Specifically, defines the ICMPv6 group. Provides a basis for the IPv6 statistics reported by the X-Pedition router.
RFC 2893	*Transition Mechanisms for IPv6 Hosts and Routers.* Specifies IPv4 compatibility mechanisms that can be implemented by IPv6 hosts and routers. These mechanisms include providing complete implementations of both versions of the Internet Protocol (IPv4 and IPv6), and tunneling IPv6 packets over IPv4 routing infrastructures. Conformant with requirements identified in the RFC for routers, with the following limitations: • Only bidirectional configured tunneling is supported by this release. Unidirectional configured tunneling and automatic tunneling are not supported. • No support for fragmentation of the IPv4 packet (after encapsulation). All IPv6-in-IPv4 packets sent by the X-Pedition router will have the IPv4 Don’t Fragment bit set.
RFC 3513 (obsoletes RFC 2373)	*Internet Protocol Version 6 (IPv6) Addressing Architecture.* Defines the addressing architecture of the IP Version 6 (IPv6) protocol. The document includes the IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and an IPv6 node's required addresses. Appendix A describes how to create modified EUI-64 format interface IDs. Fully conformant.
RFC 3587 (obsoletes RFC 2374)	*IPv6 Global Unicast Address Format* Redefines the IPv6 address allocation structure to use a coordinated allocation policy defined by the Regional Internet Registries (RIRs).

RFC 1981	*Path MTU Discovery for IP version 6.* Describes Path MTU Discovery for IP version 6. Support for the initiation of Path MTU Discovery is not provided, since none of the supported applications require this functionality. However, the Path MTU Discovery process is itself supported, for example, ICMPv6 Packet Too Big messages are generated as specified in RFC 2463.

IPv6 Implementation Notes

This section discusses information specific to the implementation of IPv6 on X-Pedition routers. Information describing how to configure IPv6 on an X-Pedition router using Native CLI IPv6-specific commands follows this section.

IPv6 support is introduced in Version 10 of the X-Pedition System Firmware. Both IPv4 and IPv6 can be supported simultaneously on the router because dual IPv4 and IPv6 stacks run in parallel on the Control Module (CM).

Hardware Requirements

IPv6 operation requires installation of an IPv6 NIC in an X-Pedition router chassis.

- X-Pedition 8000/8600 routers require the SSR-IPv6-00 module.

  Note: The SSR-IPv6-00 modules cannot be used with 8000 chassis with Rev 0 backplanes.

- The ER16 router requires the ER16-IPv6-00 module.

The SSR-IPv6 and ER16-IPv6 modules are strictly forwarding engines, used to forward IPv6 frames through an X-Pedition router. Port-based IPv6 interfaces can only be configured on Ethernet line cards.

Note: Line cards that use AA-series or T-series ASIC versions are required. Pre-AA-series line cards are not supported.

There are no restrictions related to the type of ports on which an IPv6-in-IPv4 tunnel's IPv4 interfaces can be configured. IPv6 tunnel traffic can egress an X-Pedition router via an IPv4 interface configured over a WAN port.

IPv6 Module Redundancy

Two Operational SSR-IPv6 Modules in an X-Pedition 8000/8600 Router

Each SSR IO card in an X-Pedition 8000/8600 router will automatically be mapped to an SSR-IPv6 module. The mapping algorithm will balance the number of SSR IO cards installed, across the multiple IPv6 modules. As each IPv6 interface is created, it will get mapped to the IPv6 module that its IO card has been associated with. Whenever an IPv6 interface is deleted, the entire set of IO cards and IPv6 interfaces are remapped and rebalanced across the installed SSR-IPv6 modules.

Two Operational ER16-IPv6 Modules in an ER16 Router

The IO cards in an ER16 router are mapped in a similar manner to the 8000/8600 router, with the exception that an ER16 contains two SSR sets of ports and therefore, two Crossbar Channels per IO card. Mapping and balancing in the software occurs at the Channel level so one ER16 IO card may actually utilize (that is, be mapped and balanced across) the two ER-IPv6 modules.

Hot IPv6 Module Insertion

Hot insertion will automatically cause a remapping and rebalancing of the IPv6 interfaces, including IPv6-in-IPv4 tunnel flows.

Hot IPv6 Module Removal

When the Hot Removal button is pressed, it will trigger the removal of the association of IPv6 interfaces and IO Cards to the to-be-removed IPv6 module. That association removal will result in the remapping of all IPv6 interfaces and IO cards to the remaining IPv6 module.

Interface Configuration

IPv6 interfaces can be configured using the IPv6-specific interface CLI commands **interface create ipv6** and **interface add ipv6**.

- Port-based IPv6 interfaces can be configured only on Ethernet physical ports. IPv6-in-IPv4 tunnel interfaces are virtual interfaces, and there are no restrictions on the type of port a tunnel's IPv4 interfaces are configured on.
- IPv6 interfaces will support multiple IPv6 addresses on a single physical Ethernet port.
- An Ethernet port may be configured as both an IPv6 and an IPv4 interface.

Multicast forwarding is not supported. However, interfaces are capable of joining any multicast groups required for proper protocol operation. Refer to *Configuring IPv6 Port-Based Interfaces* on page 186 for more information.

VLAN Support

The X-Pedition router supports bridging of IPv6 traffic within a user-configured protocol-based VLAN. IPv6 frames received on a port belonging to a user-configured IPv6 VLAN will be bridged to other ports that are part of the same VLAN. However, the X-Pedition router does not allow an IPv6 routing interface to be configured over a user-configured IPV6 VLAN. Therefore IPv6 traffic cannot be routed to or from a user-configured IPv6 VLAN. For additional information, refer to *IPv6 and VLANs* on page 90.

IPv6 Management

The current IPv6 implementation supports the ping and traceroute applications for determining network connectivity and topology. The **ping** and **traceroute** CLI commands have been updated to support IPv6.

IPv6-in-IPv4 Tunneling

This implementation of IPv6 supports bidirectional configured tunneling only, as defined in RFC 2893. Unidirectional configured tunneling and automatic tunneling are not supported. Refer to *Configuring IPv6-in-IPv4 Tunnels* on page 190 for more information about how an X-Pedition router handles this tunneling.

IPv6 Path MTU Discovery Support

Since this implementation of IPv6 supports only the ping and traceroute applications which do not require knowledge of Path MTU, Path MTU discovery for IPv6 as described in RFC 1981 is not supported on an X-Pedition router as a sending node.

However, ICMPv6 is supported as specified in RFC 2463, and therefore an X-Pedition router will send a Packet Too Big message to a source IPv6 node if the X-Pedition router receives a packet that cannot be forwarded because its size exceeds the MTU of the outgoing interface.

Path MTU Discovery Support for IPv6-in-IPv4 Tunnels

The X-Pedition router does not support fragmentation of tunneled packets within the IPv4 network. Each packet sent over a tunnel will have the Do Not Fragment bit set in its IPv4 header.

The MTU of the tunnel's IPv4 path must be at least 1300 bytes (which is equal to the minimum IPv6 link MTU of 1280 bytes plus the 20 byte IPv4 header length).

The tunnel's IPv6 MTU can be set from 1280 (default) to 1480 bytes, at the time of the tunnel (interface) creation. This tunnel MTU will be used by the encapsulating router at the edge of the tunnel to send an ICMPv6 Packet Too Big message when it receives an IPv6 packet which would be routed over the tunnel, but whose length is greater than this configured value. Note that the router will not translate and forward ICMPv4 error messages returned over the tunnel back to an originating IPv6 node.

For the tunnel to operate properly, the minimum MTU along the tunnel's IPv4 path must be at least 20 bytes greater than the tunnel's configured IPv6 MTU. Therefore, it is important to determine the actual minimum MTU along the tunnel's IPv4 path. This will allow IPv6 packets, which after encapsulation would require IPv4 fragmentation somewhere within the tunnel, to be rejected at the encapsulating node with an ICMPv6 message.

For the purpose of determining tunnel operating status, the router will assume the tunnel path's minimum IPv4 MTU is the outgoing link's IPv4 MTU and will mark the tunnel as down if it is not at least 20 bytes greater than the tunnel's configured IPv6 MTU.

Multicast Support

This implementation does not support IPv6 multicast forwarding. However, IPv6 interfaces are capable of joining any multicast groups required for proper protocol operation.

Multicast Listener Discovery (RFC 2710) is not supported in this implementation.

Extension Header Support

IPv6 introduced the concept of Extension headers to deal with options in a more efficient way than in IPv4. Extension headers, if they exist, are located between the IPv6 header and the upper-layer protocol header.

The X-Pedition router will forward an IPv6 packet that includes an IPv6 Extension header without processing the Extension header itself, unless the packet destination is the router itself.

IPv6 Configuration

IPv6 configuration on an X-Pedition router involves the following steps:

1. Configure IPv6 interfaces. An IPv6 interface can be a port-based interface (see *Configuring IPv6 Port-Based Interfaces* on page 186) or an IPv6-in-IPv4 tunnel (see *Configuring IPv6-in-IPv4 Tunnels* on page 190).
2. Configure static IPv6 routes, if necessary. (See *Configuring IPv6 Static Routes* on page 195.)
3. Configure Neighbor Discovery parameters. (See *Configuring Neighbor Discovery Parameters* on page 196.)
4. Configure ICMPv6 parameters. (See *Configuring ICMPv6 Parameters* on page 199.)
5. Determine network connectivity. (See *Determining Network Connectivity* on page 200)
6. Enable RIPng on the configured IPv6 interfaces and enable RIPng routing. RIPng configuration is described in Chapter 16, *RIPng Configuration Guide*.

Configuring IPv6 Port-Based Interfaces

IPv6 interfaces can be configured for Ethernet ports and for IPv6-in-IPv4 tunnels. This section describes configuration of port-based IPv6 interfaces. First, this section discusses how link-local addresses may be configured for an interface (either automatically or manually), and then describes how to create IPv6 port-based interfaces using the **interface create ipv6** command. Configuring IPv6-in-IPv4 tunnels is described on page 190.

Link-Local Addresses

When you create an IPv6 port-based interface (as described below), the router will automatically construct a link-local address for that interface, using the following procedure, in accordance with RFCs 3513 and 2464.

1. An EUI-64 identifier is constructed by inserting 0xFFFE in the middle of the X-Pedition router's IPv6 interface MAC address (router's base MAC address + 3).

   For example, if the IPv6 interface MAC address is 00-01-F4-34-56-78, then the EUI-64 identifier would be 00-01-F4-FF-FE-34-56-78.

2. An interface identifier is constructed by complementing the u/l bit (the next-to-lowest bit of the first octet) of the EUI-64 identifier.

   Continuing the example above, the interface identifier would be 02-01-F4-FF-FE-34-56-78.

3. The interface's link-local address is constructed by appending the interface identifier to the link-local prefix FE80::/64.

   Using the example above, the link-local address would be FE80:0000:0000:0000:0201:F4FF:FE34:5678.

An autoconfigured link-local address will be checked using the Duplicate Address Detection (DAD) facility of IPv6. If it passes, the interface is enabled. If it does not pass (that is, the address is not unique on the link), an error message will be displayed. You need to reenter the **interface create ipv6** command with a unique link-local address before the interface is enabled.

You can also create an IPv6 interface and manually specify only a link-local address (with the FE80::/64 prefix using the **address-prefix** option), not a global address. In this case, the router will not generate an autoconfigured link-local address. You can add global addresses to the interface with the **interface add ipv6** command.

Multicast Group Membership

Each interface enabled for IPv6 automatically is assigned the following anycast and multicast group addresses:

- Subnet-router anycast address for interfaces configured to act as routers on each link
- All-nodes multicast address
- Solicited-node multicast address for each assigned unicast and anycast address
- All-routers multicast address

Interface Configuration

You can configure the following types of IPv6 addresses using CLI commands:

- Multiple aggregatable global unicast addresses
- A site-local address
- Anycast addresses

The following syntax description of the **interface create ipv6** command illustrates the parameters used to create a *port-based* interface:

interface create ipv6 <*InterfaceName*> [**address-prefix** <*address*[/*prefix-len*]> [**eui64**] [**anycast**]] [**mtu** <*num*>] {**port** <*port*> [**mac-addr** <*MACaddr-spec*>] [**vlan-id** <*num*>]} [**up** | **down**]

The following syntax description shows the **interface add ipv6** command parameters to add secondary IPv6 addresses to a port-based interface:

interface add ipv6 <*InterfaceName*> **address-prefix** <*address*[/*prefix-len*]> [**eui64**] [**anycast**]

To create an IPv6 interface named “IPv6_1” on Ethernet port et.3.4 with a global unicast address constructed from the subnet prefix 2000::/64 and the EUI64 interface identifier for this interface, use the **interface create ipv6** command.

xp(config)# **interface create ipv6** IPv6_1 **address-prefix** 2000::/64 **eui64 port** et.3.4

To create an IPv6 interface named “IPv6_2” on Ethernet port et.4.1 and give it a manually specified link-local address, use the **interface create ipv6** command and enter the link-local prefix of FE08::64 with the **address-prefix** parameter.

```
xp(config)# interface create ipv6 IPv6_2 address-prefix FE80::2000/64 port et.4.1
```

To assign a *secondary* global unicast address to the previously created IPv6 interface, use the **interface add ipv6** command. In this example, the entire address and prefix are entered with the **address-prefix** parameter.

```
xp(config)# interface add ipv6 IPv6_2 address-prefix 3456::CD30:1234:4567:8911:ABCD/64
```

For complete syntax information for the **interface create ipv6** and **interface add ipv6** commands, see the “interface Commands” chapter in the *Enterasys X-Pedition NATIVE Command Line Interface Reference Manual*.

Displaying IPv6 Interface Information

Use the **interface show ipv6** or **ipv6 show interfaces** commands to display information about configured IPv6 interfaces.

The following examples display information about the IPv6 interface named “intf1”:

```
xp# interface show ipv6 intf1

Interface intf1:
      Admin State:         up
      Operational State:   up
      Capabilities:        <BROADCAST,SIMPLEX,MULTICAST>
      Configuration:
        VLAN:              SYS_L3_intf1
        Ports:             et.5.1
        MTU:               1500
        MAC Encapsulation:ETHERNET_II
        MAC Address:       00:E0:63:13:20:43
        IPv6 Address:      fe80::2e0:63ff:fe13:2043%intf1  (prefixlen 64 scopeid 0x4)
        IPv6 Address:      20::2  (prefixlen 64)
        IPv6 Address:      20:: (prefixlen 64 anycast)

xp# interface show ipv6 intf1 brief

Interface       Status    Oper.Status      Address                          Vlan/Port
---------       ------    ---------------  -------                          ---------
intf1           up        up               fe80::2e0:63ff:fe13:2043%intf1/64  SYS_L3_intf1/et.5.1
                                           20::2/64
                                           20::/64
```

This example displays information about all IPv6 interfaces configured on the X-Pedition router, including IPv6-in-IPv4 tunnels, if there are any configured:

```
xp# interface show ipv6 all

Interface lo1:
        Admin State:          up
        Operational State:    up
        Capabilities:         <LOOPBACK,MULTICAST>
        Configuration:
          MTU:                16384
          MAC Encapsulation:Unknown
          MAC Address:        None/Unknown
          IPv6 Address:       ::1  (prefixlen 128)

Interface intf1:
        Admin State:          up
        Operational State:    up
        Capabilities:         <BROADCAST,SIMPLEX,MULTICAST>
        Configuration:
          VLAN:               SYS_L3_intf1
          Ports:              et.5.1
          MTU:                1500
          MAC Encapsulation:ETHERNET_II
          MAC Address:        00:E0:63:13:20:43
          IPv6 Address:       fe80::2e0:63ff:fe13:2043%intf1  (prefixlen 64 scopeid 0x4)
          IPv6 Address:       20::2  (prefixlen 64)
          IPv6 Address:       20:: (prefixlen 64 anycast)

Interface intf3:
        Admin State:          up
        Operational State:    up
        Capabilities:         <BROADCAST,SIMPLEX,MULTICAST>
        Configuration:
          VLAN:               SYS_L3_intf3
          Ports:              et.5.7
          MTU:                1500
          MAC Encapsulation:ETHERNET_II
          MAC Address:        00:E0:63:13:20:43
          IPv6 Address:       fe80::2e0:63ff:fe13:2043%intf3  (prefixlen 64 scopeid 0x6)
          IPv6 Address:       23::1  (prefixlen 64)
          IPv6 Address:       23:: (prefixlen 64 anycast)
```

Use the **statistics show ipv6-interface** and **statistics show ipv6** commands to display IPv6 statistics.

The following example displays the RFC 2465 statistics for the IPv6 interface named “intf1”. This output shows the contents of the hardware counters, showing statistics for the IPv6 traffic forwarded directly by the hardware. To see statistics for the IPv6 traffic forwarded by the router software, use the **statistics show ipv6** command.

Refer to the chapter “statistics Commands” in the *Enterasys X-Pedition NATIVE Command Line Interface Reference Manual* for descriptions of the fields displayed.

```
xp# statistics show ipv6-interface intf1 detailed

 interface name: intf1                 serviced by FEA in slot-6
  RFC-2465 Counter                     Packets        Bytes
  InReceives                           35             4470
  InHdrErrors                          0              0
  InTooBigErrors                       0              0
  InNoRoutes                           0              0
  InAddrErrors                         0              0
  InUnknownProtos                      0              0
  InTruncatedPkts                      0              0
  InDiscards                           0              0
  InDelivers                           43             N/A
  OutRequests                          50             N/A
  OutFragFails                         0              0
  InMcastPkts                          45             N/A
  OutMcastPkts                         48             N/A
 ipv6 interface stats cleared:  * Never Cleared *
```

Configuring IPv6-in-IPv4 Tunnels

Router Tunneling Roles

There are three possible roles for an X-Pedition router within an IPv6-in-IPv4 tunnel:

- Encapsulating node
- Transit node
- Decapsulating node

When an X-Pedition router acts as a *transit node*, the IPv6 protocol stack and IPv6 NICs are not involved in the transmission. The router forwards the IPv4 encapsulated IPv6 traffic as it would any other IPv4 traffic.

An IPv4 / IPv6-enabled X-Pedition router acts as an *encapsulating node* when it receives an IPv6 packet with an IPv6 destination address that should be routed over a configured IPv6-in-IPv4 tunnel. The receiving router encapsulates the IPv6 packet, unaltered, in an IPv4 packet. The source address of the IPv4 packet is that of the tunnel's starting point on the encapsulating router, and the destination address is that of the tunnel's endpoint. A protocol type of 41 in the IPv4 header identifies the packet as IPv6-in-IPv4. The IPv4 packet is then routed through the IPv4 network based on the IPv4 Forwarding Information Base (FIB), just like any other IPv4 packet.

An IPv4 / IPv6-enabled X-Pedition router acts as a *decapsulating node* when it receives an IPv4 packet with an IPv4 destination address belonging to the local end of a tunnel, an IPv4 source address belonging to the remote end of a tunnel, and with a protocol type of 41 in the IPv4 header. Any existing IPv4 Access Control Lists (ACLs) will be applied to the packet, and if the packet is allowed, it will be decapsulated and the IPv4 header discarded. Any existing IPv6 ACLs will be applied to the packet, and if the packet is allowed, it will be forwarded using the IPv6 Forwarding Information Base.

As with any other IP interface, routing information (FIB entries) will determine which IPv6 traffic is routed over any configured IPv6-in-IPv4 tunnel.

Tunnel IPv6 Addresses

When creating an IPv6-in-IPv4 tunnel interface, you can choose whether or not to specify an IPv6 address (global or link-local) for the local end of the tunnel with the **address-prefix** option. If you do specify a local address, you must also specify the corresponding IPv6 address (global or link-local) for the remote end of the tunnel with the **peer-address** option.

If you specify *global* IPv6 addresses for both ends of the tunnel, the X-Pedition router automatically creates a link-local IPv6 address for the local end of the tunnel interface based on the tunnel's local IPv4 address, as specified in RFC 2893. That is, the router creates a 64-bit interface identifier from the local IPv4 interface address (entered with the **localv4** parameter), padded at the left with zeros to 64 bits, and then appends the interface id to the link-local prefix, FE80::/64. In addition, the router assumes that the link-local IPv6 address at the remote end is also formed from the remote IPv4 interface address (entered with the **remotev4** parameter).

If you do not specify *any* IPv6 addresses for the tunnel interface, the autogenerated link-local address is used for the IPv6 address of the local end of the tunnel and the router assumes that the IPv6 address of the remote end of the tunnel is also a link-local address formed from the remote IPv4 interface address.

You cannot use the **peer-address** parameter to specify a remote IPv6 address if you have not used the **address-prefix** option to specify the local IPv6 address. That is, you must use both the **address-prefix** and **peer-address** parameters, or neither parameter, when configuring tunnel interfaces.

Since the X-Pedition router can automatically create a link-local address for the local tunnel interface, and make an assumption about the IPv6 address at the remote end, you do not have to manually specify any IPv6 addresses for the tunnel if the other end of the tunnel is another X-Pedition router or some other router implementation that adheres to RFC 2893. However, if the router implementation at the remote end of the tunnel requires user-specified IPv6 addresses, then you will need to configure IPv6 addresses for both ends of the tunnel, local and remote. The manually configured IPv6 addresses can be either global or link-local addresses.

Configuring Tunnels

You create IPv6-in-IPv4 tunnel interfaces with the **interface create ipv6** command. Before an IPv6-in-IPv4 tunnel interface can become operational, a valid route to the remote IPv4 end of the tunnel must exist on the local router.

You can add secondary IPv6 addresses to an existing tunnel interface with the **interface add ipv6** command.

The following syntax description of the **interface create ipv6** command illustrates the parameters used to create a *tunnel* interface:

interface create ipv6 <*Interface Name*> [**address-prefix** <*address*[/*prefix-len*]> [**eui64**] [**anycast**]] [**mtu** <*num*>] {**tunnel** [**peer-address** <*address*>] **localv4** *local_v4_interface* **remotev4** *remote_v4_address*} [**up** | **down**]

The following syntax description of the **interface add ipv6** command illustrates the parameters used to add an IPv6 address to a tunnel interface:

interface add ipv6 <*InterfaceName*> **address-prefix** <*address*[/*prefix-len*]> [**eui64**] [**anycast**] **peer-address** <*address*>

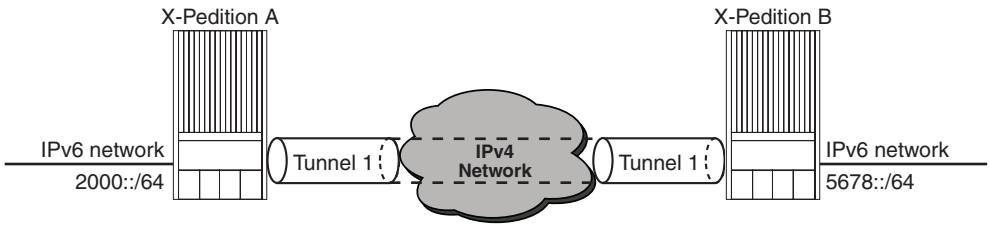
The following example, illustrated in Figure 19 on page 193, creates an IPv6-in-IPv4 tunnel between two X-Pedition routers, A and B. On X-Pedition A, the local IPv4 interface is named “IPv4_1” and has the address 10.10.10.10/24. The tunnel interface is given the name “tunnel_1” and is given the IPv6 address of ABCD::1. On X-Pedition B, the local IPv4 interface is named “IPv4_7” and has the address 10.10.10.20/24. The tunnel interface is given the name “tunnel_1” and is given the IPv6 address of ABCD::2.

Note: Since the tunnel is between two X-Pedition routers, we did not have to manually assign IPv6 addresses to the tunnel endpoints because the autogenerated link-local addresses for the two ends would have been sufficient.

Note that if the two IPv4 interfaces of the tunnel endpoints were not on the same subnet, an IPv4 route would be required to establish IPv4 connectivity. The route can be statically configured or established via an IPv4 routing protocol such as RIP or OSPF. The same is true for the IPv6 endpoints. In the following example, static routes are configured to map additional IPv6 networks to the tunnel.

```
xpa(config)# interface create ip IPv4_1 address-netmask 10.10.10.10/255.255.255.0 port et.1.1
xpa(config)# interface create ipv6 tunnel_1 address-prefix ABCD::1 tunnel peer-address ABCD::2
localv4 IPv4_1 remotev4 10.10.10.20
xpa(config)# ipv6 add route 5678::/64 interface tunnel_1

xpb(config)# interface create ip IPv4_7 address-netmask 10.10.10.20/255.255.255.0 port et.1.1
xpb(config)# interface create ipv6 tunnel_1 address-prefix ABCD::2 tunnel peer-address ABCD::1
localv4 IPv4_7 remotev4 10.10.10.10
xpb(config)# ipv6 add route 2000::/64 interface tunnel_1
```

Figure 19. Example of IPv6-in-IPv4 Tunnel Configuration



X-Pedition A Tunnel Configuration

Local IPv4 interface name: IPv4_1
IPv4_1 IP Address: 10.10.10.10
Remote IPv4 address: 10.10.10.20
Tunnel interface name: Tunnel 1
Tunnel IPV6 address: ABCD::1/128

Static route to 5678::/64

X-Pedition B Tunnel Configuration

Local IPv4 interface name: IPv4_7
IPv4_7 IP Address: 10.10.10.20
Remote IPv4 address: 10.10.10.10
Tunnel interface name: Tunnel 1
Tunnel IPV6 address: ABCD::2/128

Static route to 2000::/64

Displaying IPv6-in-IPv4 Tunnel Information

Use the **interface show ipv6** or **ipv6 show interfaces** commands to display information about configured tunnels.

The following example displays complete information about all tunnel interfaces. You can also use the **brief** option to display summary information.

```
xp# interface show ipv6 tunnels

Interface tunn_1:
        Admin State:          up
        Operational State:    up
        Capabilities:         <POINTTOPOINT,MULTICAST>
        Configuration:
          MTU:                1280
          Encapsulation:      IPv4
          Tunnel Addresses: 20.1.1.2 --> 20.1.1.1
          IPv6 Address:       fe80::1401:102%tunn_1 -->fe80::1401:101%tunn_1 (prefixlen 128 scopeid 0x7)
```

Configuring IPv6 Static Routes

Use the **ipv6 add route** command to configure static routes in the IPv6 Unicast Route Table. The static route can be a default route, a route to a network, or a route to a specific host. Static routes over IPv6-in-IPv4 tunnels are configured in the same way as static routes over Ethernet interfaces, although you can specify an interface without specifying a gateway when configuring a static route over a tunnel.

The following example configures the router 4fde::2:22:f376:ff3b:ab3f as the default gateway for this X-Pedition router:

```
xp(config)# ipv6 add route default gateway 4fde::2:22:f376:ff3b:ab3f
```

The following example configures a static route for the subnet 4000::/64 over the IPv6-in-IPv4 tunnel interface named “tunnel_1”:

```
xp(config)# ipv6 add route 4000::/64 interface tunnel_1
```

For complete syntax information for the **ipv6 add route** command, see the “ipv6 Commands” chapter in the *Enterasys X-Pedition NATIVE Command Line Interface Reference Manual*.

Use the **ipv6 show routes** command to display the IPv6 routing table. The following example shows that some of the route entries are for locally connected interfaces ("directly connected"), while some of the other routes are learned from RIPng. The “=>” after the route on the second line indicates that it shares a duplicate key in the routing table with the route below it, and is the preferred route because of higher preference for static routes.

```
xp# ipv6 show routes

Destination                     Gateway                    Owner        Netif
-----------                     -------                    -----        -----
::1                              directly connected          -            lo1
2700::                          2130::2                    Static      intf3 =>
2700::/64                        fe80::2%intf3               RIPNG        intf3
fe80::2e0:63ff:fe11:6043%intf2    directly connected          -            lo1
fe80::%intf3/64                  directly connected          -           intf3
ff01::%lo1/32                    ::1                        -            lo1
ff01::%intf3/32                  directly connected          -           intf3
ff02::%lo1/32                    ::1                        -            lo1
ff02::%intf3/32                  directly connected          -           intf3
```

Configuring Neighbor Discovery Parameters

Neighbor Discovery CLI commands allow you to configure an X-Pedition router's behavior relative to the IPv6 Neighbor Discovery and Stateless Autoconfiguration protocols.

For the complete syntax of the Neighbor Discovery commands, see the "ndisc Commands" chapter in the *Enterasys X-Pedition NATIVE Command Line Interface Reference Manual*.

Configuring Router Advertisement Parameters

By default, an X-Pedition router sends Router Advertisement messages out an IPv6 interface when that interface is enabled with an IPv6 address, in response to a Router Solicitation message from a host, and when triggered by a periodic timer. The **ndisc set interface** command allows you to suppress the sending of Router Advertisements on one interface or all interfaces, and to set the interval for sending unsolicited Router Advertisements.

Other Router Advertisement parameters that you can configure with the **ndisc set interface** command include:

- Specifying whether the router will support being a default gateway router over a particular interface
- Specifying whether hosts receiving the router's advertisements should use stateful address configuration or stateless address autoconfiguration
- Specifying whether stateful configuration should be used for other, non-address-related information
- Specifying the length of time a node receiving the router's advertisements should assume a neighbor remains reachable
- Specifying the length of time a node receiving the router's advertisements should wait between sending Neighbor Solicitation messages

This syntax description illustrates all the parameters of the **ndisc set interface** command:

ndisc set interface <*interfacename-or-ipv6addr* >|**all** [**suppress-ra**] [**max-ra-interval** <*num*>] [**min-ra-interval** <*num*>] [**managed-config-flag**] [**other-config-flag**] [**ra-lifetime** <*num*>] [**reachable-time** <*num*>] [**ns-interval** <*num*>]

The following example changes the interval for unsolicited multicast Router Advertisements by setting the minimum interval to 4 minutes (240 seconds) for all IPv6 interfaces on the X-Pedition router that have not been specifically set. If an individual interface has been configured separately, this command will not change its configuration parameters.

```
xp(config)# ndisc set interface all min-ra-interval 240
```

The following example configures the existing IPv6 interface named “IPv6_1” to advertise itself as not being available as a default router. The example also sets the “Reachable Time” value sent in Router Advertisements to 10,000 milliseconds. This parameter is used by the Neighbor Unreachability Detection algorithm by the on-link hosts receiving advertisements from this router and also by the router itself.

```
xp(config)# ndisc set interface IPv6_1 ra-lifetime 0 reachable-time 10000
```

Configuring Prefixes

The **ndisc add prefix** command lets you configure prefixes to be included in IPv6 Router Advertisements transmitted on a specific interface. Such address prefixes are used by on-link nodes performing stateless address autoconfiguration. By default, the prefixes configured as addresses on an interface using the **interface create ipv6** or **interface add ipv6** commands, with the exception of link-local addresses, are advertised in IPv6 Router Advertisements. If you configure prefixes for advertisement using the **ndisc add prefix** command, then only the configured prefixes are advertised.

Note that link-local addresses are *not* advertised in the set of address prefixes in the Router Advertisement messages.

The following parameters associated with a prefix may be configured:

- The value of the prefix and its length.
- The interface on which the prefix is advertised.
- The “Valid Lifetime” value for the prefix in IPv6 Router Advertisements. This value defines the period of validity of the prefix for on-link determination and also the period of validity of any host address derived from the prefix via stateless autoconfiguration.
- The “Preferred Lifetime” value for the prefix in IPv6 Router Advertisements. This value defines the period of time that an address derived from the prefix via stateless autoconfiguration remains in a preferred state.
- Whether or not the prefix is included in IPv6 Router Advertisements.
- The status of the “on-link” flag for the prefix in IPv6 Router Advertisements. When this flag is set, it indicates that the addresses implied by the prefix are directly reachable by the specified interface. When it is cleared, a receiving host may not assume that the addresses implied by the prefix are available on the receiving interface.
- The status of the “autonomous address-configuration” flag for the prefix in IPv6 Router Advertisements. When set, this flag indicates that this prefix can be used for stateless address autoconfiguration. When cleared, this prefix may not be used for stateless autoconfiguration.

This syntax description illustrates all the parameters of the **ndisc add prefix** command:

```
ndisc add prefix <ipv6-prefix>/<length> interface <interfacename-or-ipv6addr>
[valid-lifetime <num> | <date-time> | infinite] [preferred-lifetime <num>| <date-time> | infinite]
[no-advertise] [off-link] [no-autoconfig]
```

The following example configures the prefix 12AB:34CD:56EF:7891::/64 to be advertised on the IPv6 interface named “ipv6_2,” using the default value for valid lifetime (30 days) but configuring a preferred lifetime of 1,209,600 seconds (14 days):

```
xp(config)# ndisc add prefix 22AB:34CD:56EF:7891::/64 interface ipv6_2 preferred-lifetime 1209600
```

Configuring Static Entries in the Neighbor Cache

The **ndisc add neighbor** command allows you to configure static entries in the IPv6 Neighbor Cache. The IPv6 address of the specified host is associated with the specified MAC address in the cache. This command is similar to the IPv4 **arp add** command. If a dynamic entry for the specified host (learned through the IPv6 Neighbor Discovery process) already exists in the Neighbor Cache, the entry is converted to a static entry. Static entries in the cache are not modified by the neighbor discovery process. Neighbor unreachability detection is not performed on static entries in the Neighbor Cache.

Use the **no** form of **ndisc add neighbor** command to delete a previously configured static entry. The **ndisc clear neighbor** command clears dynamic entries only, and cannot be used to delete static entries in the Neighbor Cache. You can use the **ndisc show neighbor** command to view both static and dynamic entries in the IPv6 Neighbor Cache.

The following example adds a static entry to the Neighbor Cache for an IPv6 host with the address 2134::2 and a MAC address 00-01-F4-34-56-78:

```
xp(config)# ndisc add neighbor 2134::2 mac-addr 00-01-F4-34-56-78
```

Configuring Duplicate Address Detection

The **ndisc set dad-attempts** command allows you to configure the number of consecutive IPv6 neighbor solicitation messages sent by an interface when duplicate address detection (DAD) is performed on a unicast IPv6 interface address assigned to that interface. If you enter a value of 0 with this command, you disable DAD on all IPv6 interfaces. The **ndisc show dad-attempts** command displays the value of the configured value.

The following example changes the number of consecutive DAD attempts for all IPv6 interfaces on the X-Pedition router, from the default of 1 to 3 attempts:

```
xp(config)# ndisc set dad-attempts 3
```

Displaying Neighbor Discovery Information

Use the following commands to display Neighbor Discovery information:

ndisc show interface	Displays configured IPv6 Neighbor Discovery protocol information for a single interface or all interfaces.
ndisc show prefix	Displays information about prefixes that are configured for inclusion in IPv6 Router Advertisements.
ndisc show neighbor	Displays IPv6 Neighbor Discovery Cache information for a specific neighbor or for all known neighbors.
ndisc show dad-attempts	Displays the value of the IPv6 neighbor discovery "dad-attempts" parameter for all IPv6 interfaces on the X-Pedition router.

Configuring ICMPv6 Parameters

Use the **ipv6 set icmp-rate-limit** command to limit the rate at which ICMPv6 error messages are generated by an X-Pedition router. You can specify the maximum rate at which error messages are sent, in units of packets per second. For example, to change the retransmission rate to 10 packets per second (from the default of 100):

```
xp(config)# ipv6 set icmp-rate-limit 10
```

Use the **ipv6 disable icmp-redirect** command to disable the generation of ICMPv6 Redirect messages when a packet is forwarded through the same interface on which it was received. Sending of Redirects is enabled by default.

Use the **system show active-config** command to display the current settings of these parameters.

Determining Network Connectivity

You can use the CLI commands **ping** and **traceroute** for determining network connectivity and topology.

```
xp# ping 22::2e0:63ff:fe13:2003
PING6(64=40+8+16 bytes) 20::2e0:63ff:fe13:2043 --> 22::2e0:63ff:fe13:2003
24 bytes from 22::2e0:63ff:fe13:2003, icmp_seq=0 time=222.413 ms
24 bytes from 22::2e0:63ff:fe13:2003, icmp_seq=1 time=111.879 ms
24 bytes from 22::2e0:63ff:fe13:2003, icmp_seq=2 time=96.472 ms
24 bytes from 22::2e0:63ff:fe13:2003, icmp_seq=3 time=98.02 ms
24 bytes from 22::2e0:63ff:fe13:2003, icmp_seq=4 time=102.593 ms

--- 22::2e0:63ff:fe13:2003 ping6 statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 96.472/126.275/222.413/48.368 ms
```

```
xp# traceroute 23::2e0:63ff:fe13:2043
traceroute6 to 23::2e0:63ff:fe13:2043 (23::2e0:63ff:fe13:2043) from 20::2e0:63ff:fe13:2003, 64 hops max,
12 byte packets
 1  23::2e0:63ff:fe13:2043  100 ms  120 ms  110 ms
```

Chapter 14

VRRP Configuration Guide

This chapter explains how to set up and monitor the Virtual Router Redundancy Protocol (VRRP) on the X-Pedition router. VRRP is defined in RFC 2338.

VRRP Overview

End host systems on a LAN are often configured to send packets to a statically configured default router. If this default router becomes unavailable, all the hosts that use it as their first hop router become isolated on the network. VRRP provides a way to ensure the availability of an end host's default router.

This is done by assigning IP addresses that end hosts use as their default route to a "virtual router." A Master router is assigned to forward traffic designated for the virtual router. If the Master router should become unavailable, a backup router takes over and begins forwarding traffic for the virtual router. As long as one of the routers in a VRRP configuration is up, the IP addresses assigned to the virtual router are always available, and the end hosts can send packets to these IP addresses without interruption.

Note: In some instances, a heavy load on a VRRP master may delay VRRP packet transmission and cause the backup router to assume the role of master.

Note: The X-Pedition router supports only 512 instances of VRRP. An instance is defined as one virtual router running on one interface. Running a single virtual router on four interfaces is considered four instances of VRRP, as is running four virtual routers on a single interface.

Configuring VRRP

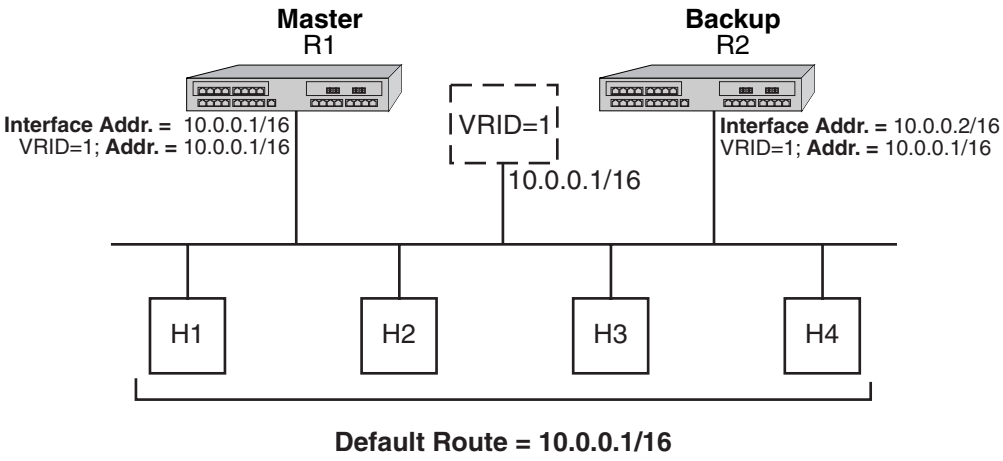
This section presents three sample VRRP configurations:

- A basic VRRP configuration with one virtual router
- A symmetrical VRRP configuration with two virtual routers
- A multi-backup VRRP configuration with three virtual routers

Basic VRRP Configuration

Figure 20 shows a basic VRRP configuration with a single virtual router. Routers R1 and R2 are both configured with one virtual router (VRID=1). Router R1 serves as the Master and Router R2 serves as the Backup. The four end hosts are configured to use 10.0.0.1/16 as the default route. IP address 10.0.0.1/16 is associated with virtual router VRID=1.

Figure 20. Basic VRRP Configuration



If Router R1 should become unavailable, Router R2 would take over virtual router VRID=1 and its associated IP addresses. Packets sent to 10.0.0.1/16 would go to Router R2. When Router R1 comes up again, it would take over as Master, and Router R2 would revert to Backup.

Notes:

- In some instances, a heavy load on a VRRP master may delay VRRP packet transmission and cause the backup router to assume the role of master.
- The X-Pedition router supports only 512 instances of VRRP. An instance is defined as one virtual router running on one interface. Running a single virtual router on four interfaces is considered four instances of VRRP, as is running four virtual routers on a single interface.
- Do not use an IP address for VRRP that is already configured for load-balancing.

Configuration of Router R1

The following is the configuration file for Router R1 in Figure 20.

```
1: interface create ip test address-netmask 10.0.0.1/16 port et.1.1
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
4: ip-redundancy start vrrp 1 interface test
```

Line 1 adds IP address 10.0.0.1/16 to interface test, making Router R1 the owner of this IP address. Line 2 creates virtual router VRID=1 on interface test. Line 3 associates IP address 10.0.0.1/16 with virtual router VRID=1. Line 4 starts VRRP on interface test.

In VRRP, the router that owns the IP address associated with the virtual router is the Master. Any other routers that participate in this virtual router are Backups. In this configuration, Router R1 is the Master for virtual router VRID=1 because it owns 10.0.0.1/16, the IP address associated with virtual router VRID=1.

Configuration for Router R2

The following is the configuration file for Router R2 in Figure 20.

```
1: interface create ip test address-netmask 10.0.0.2/16 port et.1.1
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
4: ip-redundancy start vrrp 1 interface test
```

The configuration for Router R2 is nearly identical to Router R1. The difference is that Router R2 does not own IP address 10.0.0.1/16. Since Router R2 does not own this IP address, it is the Backup. It will take over from the Master if it should become unavailable.

Ping the Backup VRRP Router

When enterprise customers run an X-Pedition router in a VRRP configuration, the customers may not know if a problem exists with the Backup router. As a result, the X-Pedition router feature set includes the ability to ping the Backup router while the router is in a non-Master state. When a Backup VRRP router assumes mastership, RFC 2338 specifies that it must not answer to ICMP echo requests (pings) destined to the associated virtual address. In some network situations, however, you may want to permit the Backup router to respond with an ICMP Echo Response when it is in the Master state. Use the following command to enable ICMP Echo Response:

ip-redundancy set vrrp *<vrID>* **interface** *<interface>* **icmp-response**

For more information, see the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Symmetrical Configuration

Figure 21 shows a VRRP configuration with two routers and two virtual routers. Routers R1 and R2 are both configured with two virtual routers (VRID=1 and VRID=2).

Note: Do not use an IP address for VRRP that is already configured for load-balancing.

Router R1 serves as:

- Master for VRID=1
- Backup for VRID=2

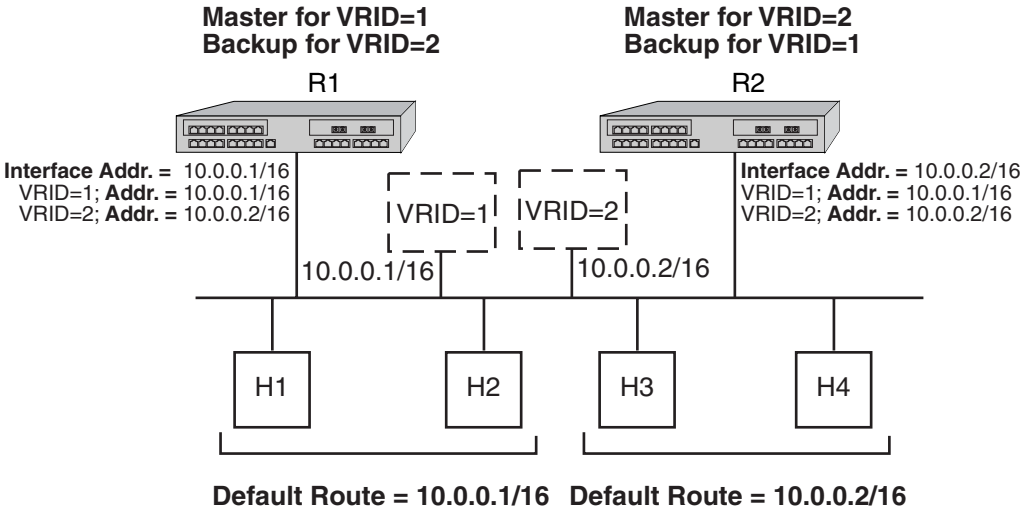
Router R2 serves as:

- Master for VRID=2
- Backup for VRID=1

This configuration allows you to load-balance traffic coming from the hosts on the 10.0.0.0/16 subnet and provides a redundant path to either virtual router.

Note: This is the recommended configuration on a network using VRRP.

Figure 21. Symmetrical VRRP Configuration



In this configuration, half the hosts use 10.0.0.1/16 as their default route, and half use 10.0.0.2/16. IP address 10.0.0.1/16 is associated with virtual router VRID=1, and IP address 10.0.0.2/16 is associated with virtual router VRID=2.

If Router R1, the Master for virtual router VRID=1, goes down, Router R2 would take over the IP address 10.0.0.1/16. Similarly, if Router R2, the Master for virtual router VRID=2, goes down, Router R1 would take over the IP address 10.0.0.2/16.

Configuration of Router R1

The following is the configuration file for Router R1 in Figure 21.

```
1: interface create ip test address-netmask 10.0.0.1/16 port et.1.1
 !
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy create vrrp 2 interface test
 !
4: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
5: ip-redundancy associate vrrp 2 interface test address 10.0.0.2/16
 !
6: ip-redundancy start vrrp 1 interface test
7: ip-redundancy start vrrp 2 interface test
```

Router R1 is the owner of IP address 10.0.0.1/16. Line 4 associates this IP address with virtual router VRID=1, so Router R1 is the Master for virtual router VRID=1.

On line 5, Router R1 associates IP address 10.0.0.2/16 with virtual router VRID=2. However, since Router R1 does not own IP address 10.0.0.2/16, it is not the default Master for VRID=2.

Configuration of Router R2

The following is the configuration file for Router R2 in Figure 21.

```
1: interface create ip test address-netmask 10.0.0.2/16 port et.1.1
 !
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy create vrrp 2 interface test
 !
4: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
5: ip-redundancy associate vrrp 2 interface test address 10.0.0.2/16
 !
6: ip-redundancy start vrrp 1 interface test
7: ip-redundancy start vrrp 2 interface test
```

On line 1, Router R2 is made owner of IP address 10.0.0.2/16. Line 5 associates this IP address with virtual router VRID=2, so Router R2 is the Master for virtual router VRID=2. Line 4 associates IP address 10.0.0.1/16 with virtual router VRID=1, making Router R2 the Backup for VRID=1.

Ping the Backup VRRP Router

When running an X-Pedition router in a VRRP configuration, the enterprise customers may not know if a problem exists with the Backup router. As a result, the X-Pedition feature set includes the ability to ping the Backup router while the router is in a non-Master state. When a Backup VRRP router assumes mastership, RFC 2338 specifies that it must not answer to ICMP echo requests (pings) destined to the associated virtual address. In some network situations, however, you may want to permit the Backup router to respond with an ICMP Echo Response when it is in the Master state.

Use the following command to enable ICMP Echo Response:

ip-redundancy set vrrp <*vrID*> **interface** <*interface*> **icmp-response**

For more information, see the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Multi-Backup Configuration

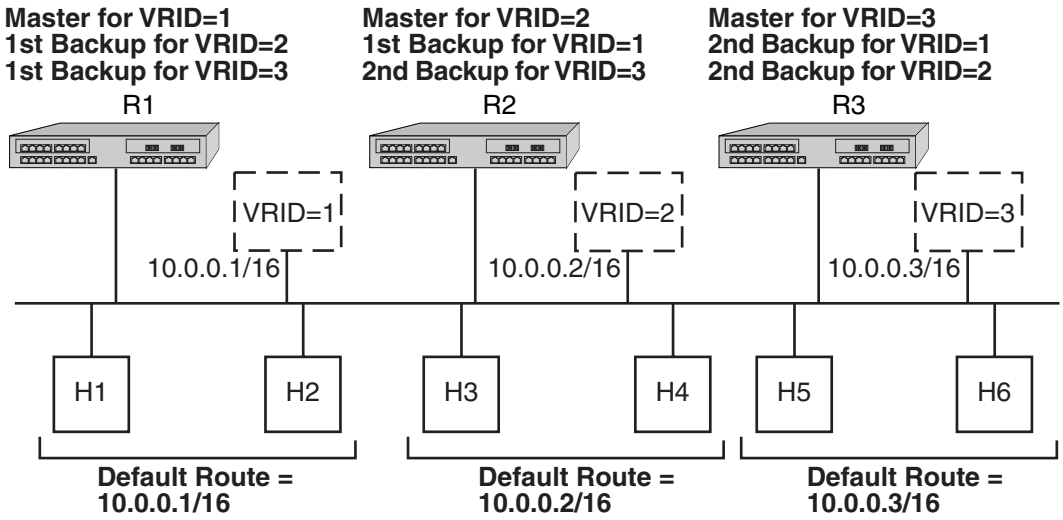
Figure 22 shows a VRRP configuration with three routers and three virtual routers. Each router serves as a Master for one virtual router and as a Backup for each of the others. When a Master router goes down, one of the Backups takes over the IP addresses of its virtual router.

In a VRRP configuration where more than one router is backing up a Master, you can specify which Backup router takes over when the Master goes down by setting the priority for the Backup routers.

Notes:

- In some instances, a heavy load on a VRRP master may delay VRRP packet transmission and cause the backup router to assume the role of master.
- The X-Pedition router supports only 512 instances of VRRP. An instance is defined as one virtual router running on one interface. Running a single virtual router on four interfaces is considered four instances of VRRP, as is running four virtual routers on a single interface.
- Do not use an IP address for VRRP that is already configured for load-balancing.

Figure 22. Multi-Backup VRRP Configuration



In this configuration, Router R1 is the Master for virtual router VRID=1 and the primary Backup for virtual routers VRID=2 and VRID=3. If Router R2 or R3 were to go down, Router R1 would assume the IP addresses associated with virtual routers VRID=2 and VRID=3.

Router R2 is the Master for virtual router VRID=2, the primary backup for virtual router VRID=1, and the secondary Backup for virtual router VRID=3. If Router R1 should fail, Router R2 would become the Master for virtual router VRID=1. If both Routers R1 and R3 should fail, Router R2 would become the Master for all three virtual routers. Packets sent to IP addresses 10.0.0.1/16, 10.0.0.2/16, and 10.0.0.3/16 would all go to Router R2.

Router R3 is the secondary Backup for virtual routers VRID=1 and VRID=2. It would become a Master router only if both Routers R1 and R2 should fail. In such a case, Router R3 would become the Master for all three virtual routers.

Configuration of Router R1

The following is the configuration file for Router R1 in Figure 22.

```
1: interface create ip test address-netmask 10.0.0.1/16 port et.1.1
 !
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy create vrrp 2 interface test
4: ip-redundancy create vrrp 3 interface test
 !
5: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
6: ip-redundancy associate vrrp 2 interface test address 10.0.0.2/16
7: ip-redundancy associate vrrp 3 interface test address 10.0.0.3/16
 !
8: ip-redundancy set vrrp 2 interface test priority 200
9: ip-redundancy set vrrp 3 interface test priority 200
 !
10: ip-redundancy start vrrp 1 interface test
11: ip-redundancy start vrrp 2 interface test
12: ip-redundancy start vrrp 3 interface test
```

Router R1’s IP address on interface test is 10.0.0.1. There are three virtual routers on this interface:

- VRID=1 – IP address=10.0.0.1/16
- VRID=2 – IP address=10.0.0.2/16
- VRID=3 – IP address=10.0.0.3/16

Since the IP address of virtual router VRID=1 is the same as the interface’s IP address (10.0.0.1), then the router automatically becomes the address owner of virtual router VRID=1.

A priority is associated with each of the virtual routers. The priority determines whether the router will become the Master or the Backup for a particular virtual router. Priorities can have values between 1 and 255. When a Master router goes down, the router with the next-highest priority takes over the virtual router. If more than one router has the next-highest priority, the router that has the highest-numbered interface IP address becomes the Master.

If a router is the address owner for a virtual router, then its priority for that virtual router is 255 and cannot be changed. If a router is ***not*** the address-owner for a virtual-router, then its priority for that virtual router is 100 by default, and can be changed by the user.

Since Router R1 is the owner of the IP address associated with virtual router VRID=1, it has a priority of 255 (the highest) for virtual router VRID=1. Lines 8 and 9 set Router R1's priority for virtual routers VRID=2 and VRID=3 at 200. If no other routers in the VRRP configuration have a higher priority, Router R1 will take over as Master for virtual routers VRID=2 and VRID=3, should Router R2 or R3 go down. The following table shows the priorities for each virtual router configured on Router R1.

Virtual Router	Default Priority	Configured Priority
VRID=1 – IP address=10.0.0.1/16	255 (address owner)	255 (address owner)
VRID=2 – IP address=10.0.0.2/16	100	200 (see line 8)
VRID=3 – IP address=10.0.0.3/16	100	200 (see line 9)

Configuration of Router R2

The following is the configuration file for Router R2 in Figure 22.

```
1: interface create ip test address-netmask 10.0.0.2/16 port et.1.1
 !
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy create vrrp 2 interface test
4: ip-redundancy create vrrp 3 interface test
 !
5: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
6: ip-redundancy associate vrrp 2 interface test address 10.0.0.2/16
7: ip-redundancy associate vrrp 3 interface test address 10.0.0.3/16
 !
8: ip-redundancy set vrrp 1 interface test priority 200
9: ip-redundancy set vrrp 3 interface test priority 100
 !
10: ip-redundancy start vrrp 1 interface test
11: ip-redundancy start vrrp 2 interface test
12: ip-redundancy start vrrp 3 interface test
```

Line 8 sets the backup priority for virtual router VRID=1 to 200. Since this number is higher than Router R3's backup priority for virtual router VRID=1, Router R2 is the primary Backup, and Router R3 is the secondary Backup for virtual router VRID=1.

On line 9, the backup priority for virtual router VRID=3 is set to 100. Since Router R1's backup priority for this virtual router is 200, Router R1 is the primary Backup, and Router R2 is the secondary Backup for virtual router VRID=3. The following table shows the priorities for each virtual router configured on Router R2.

Virtual Router	Default Priority	Configured Priority
VRID=1 – IP address=10.0.0.1/16	100	200 (see line 8)
VRID=2 – IP address=10.0.0.2/16	255 (address owner)	255 (address owner)
VRID=3 – IP address=10.0.0.3/16	100	100 (see line 9)

Note: Since 100 is the default priority, line 9, which sets the priority to 100, is actually unnecessary. It is included for illustration purposes only.

Ping the Backup VRRP Router

When enterprise customers run an X-Pedition router in a VRRP configuration, the customers may not know if a problem exists with the Backup router. As a result, the X-Pedition router feature set includes the ability to ping the Backup router while the router is in a non-Master state. When a Backup VRRP router assumes mastership, RFC 2338 specifies that it must not answer to ICMP echo requests (pings) destined to the associated virtual address. In some network situations, however, you may want to permit the Backup router to respond with an ICMP Echo Response when it is in the Master state. Use the following command to enable ICMP Echo Response:

ip-redundancy set vrrp <*vrID*> **interface** <*interface*> **icmp-response**

For more information, see the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Configuration of Router R3

The following is the configuration file for Router R3 in Figure 22.

```
1: interface create ip test address-netmask 10.0.0.3/16 port et.1.1
 !
2: ip-redundancy create vrrp 1 interface test
3: ip-redundancy create vrrp 2 interface test
4: ip-redundancy create vrrp 3 interface test
 !
5: ip-redundancy associate vrrp 1 interface test address 10.0.0.1/16
6: ip-redundancy associate vrrp 2 interface test address 10.0.0.2/16
7: ip-redundancy associate vrrp 3 interface test address 10.0.0.3/16
 !
8: ip-redundancy set vrrp 1 interface test priority 100
9: ip-redundancy set vrrp 2 interface test priority 100
 !
10: ip-redundancy start vrrp 1 interface test
11: ip-redundancy start vrrp 2 interface test
12: ip-redundancy start vrrp 3 interface test
```

Lines 8 and 9 set the backup priority for Router R3 at 100 for virtual routers VRID=1 and VRID=2. Since Router R1 has a priority of 200 for backing up virtual router VRID=2, and Router R2 has a priority of 200 for backing up virtual router VRID=1, Router R3 is the secondary Backup for both virtual routers VRID=1 and VRID=2. The following table shows the priorities for each virtual router configured on Router R3.

Virtual Router	Default Priority	Configured Priority
VRID=1 – IP address=10.0.0.1/16	100	100 (see line 8)
VRID=2 – IP address=10.0.0.2/16	100	100 (see line 9)
VRID=3 – IP address=10.0.0.3/16	255 (address owner)	255 (address owner)

Note: Since 100 is the default priority, lines 8 and 9, which set the priority to 100, are actually unnecessary. They are included for illustration purposes only.

Ping the Backup VRRP Router

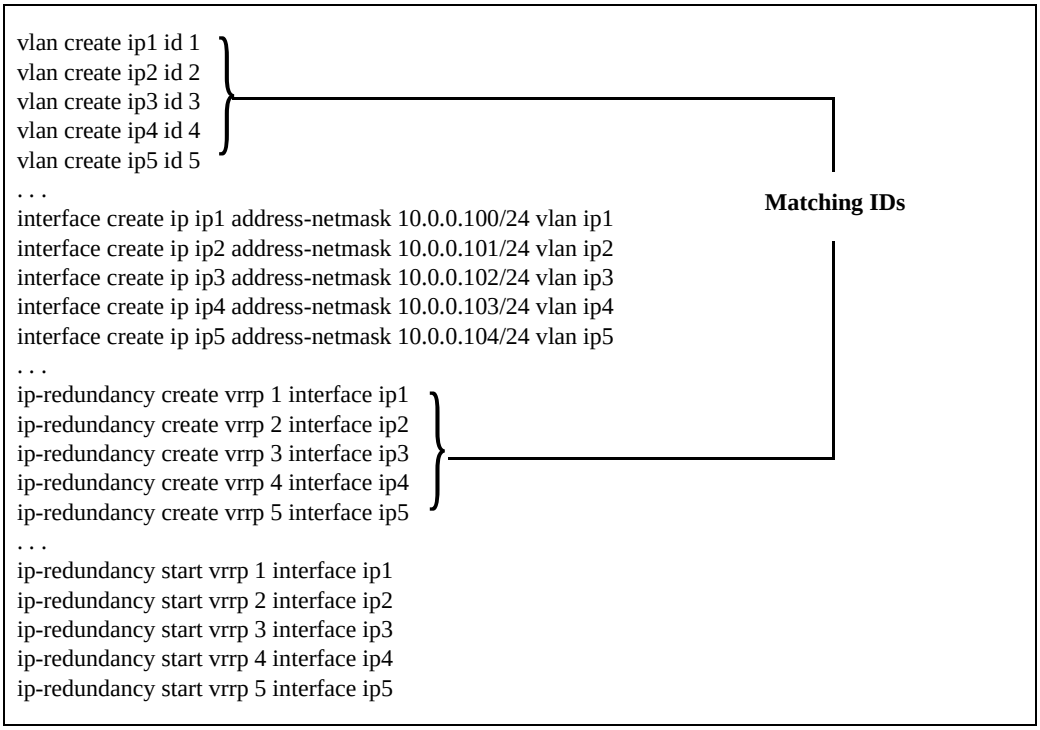
When enterprise customers run an X-Pedition router in a VRRP configuration, the customers may not know if a problem exists with the Backup router. As a result, the X-Pedition router feature set includes the ability to ping the Backup router while the router is in a non-Master state. When a Backup VRRP router assumes mastership, RFC 2338 specifies that it must not answer to ICMP echo requests (pings) destined to the associated virtual address. In some network situations, however, you may want to permit the Backup router to respond with an ICMP Echo Response when it is in the Master state. Use the following command to enable ICMP Echo Response:

ip-redundancy set vrrp <*vrID*> **interface** <*interface*> **icmp-response**

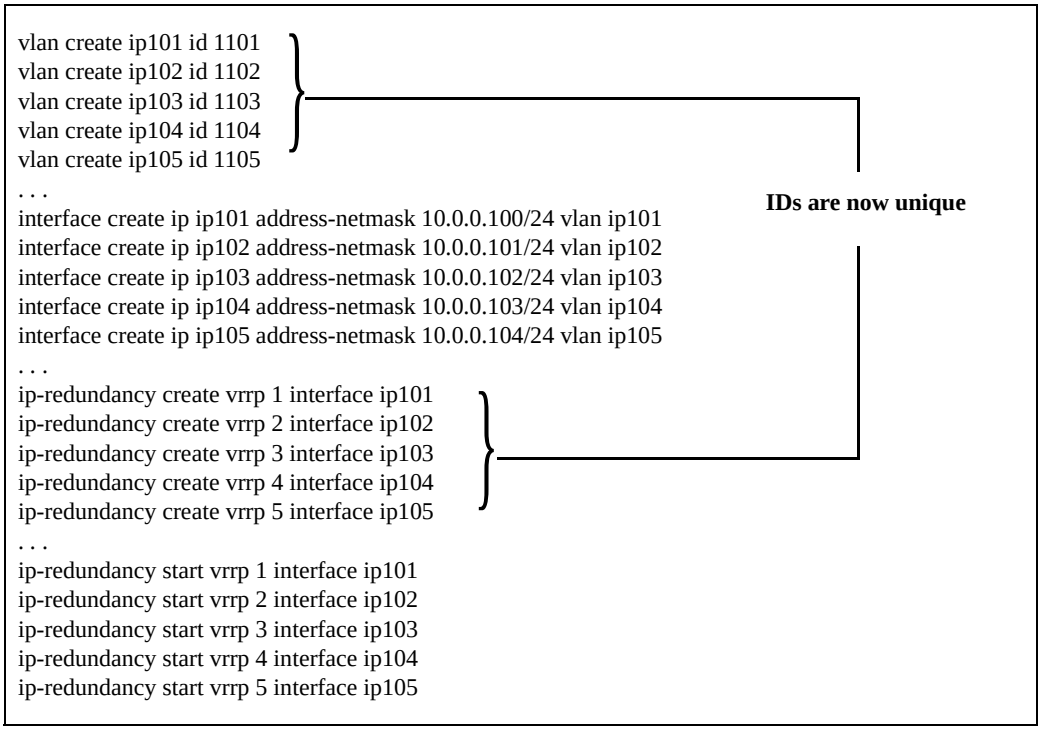
For more information, see the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Using VLANs with VRRP

A natural method of assigning VLAN and VRRP IDs is to start at 1, 10, 100, or something similar and increase incrementally through a numeric range. However, if you assign the configuration to the router using the same sequence of numbers for VLAN IDs as for VRRP IDs, a problem will result. Part of the process of creating VRRP entries is to mark the L2 table entry associated with the VRRP and VLAN permanent in memory to prevent entries from aging out. When too many entries are marked permanent and *hash* into the same area (i.e., what happens when the VLAN and VRRP entries match and are sequential), the hashed address space used to store and retrieve L2 addresses fills up and the router cannot create more VRRPs (see *Hashing* on page 213 for an explanation of how the X-Pedition router stores and accesses this information). In firmware versions prior to E8.3.0.9, this will cause the X-Pedition router to crash—later versions may report an error and will not create all of the VRRPs. The following example depicts a repeated series of ID numbers:



To avoid this problem, assign VLAN IDs that differ from your VRRP IDs. If you typically use the same numbers for both VLAN and VRRP IDs, a simple way to correct this problem is to adjust one set of numbers by adding a digit to the front of the ID (e.g., 101–105 becomes 1101–1105) or to increment the ID's first digit (e.g., 101–105 becomes 201–205). The following example shows the corrected IDs.



Hashing

Hashing is a method used to store and access items without having to create a separate storage location for each item. Similar to a mail system that uses single slots for A-K, L-O, P-S, and T-Z, L2 hashing uses “buckets” to store up to four MAC addresses in the same location. Typically, once all of the buckets are filled, the router will remove an old item to make room for a new one.

Auto-hashing determines optimal hashing algorithms for populating L2 and L3 tables. It detects overflows and adjusts the hash algorithm accordingly within 3 seconds and determines the best algorithm within 24 seconds when traffic is such that all algorithms result in overflows. By default, auto-hashing is disabled.

If entries are made permanent, they cannot be removed and the system cannot create a place to store additional VRRP MAC address information. This prevents the router from starting more VRRPs. To correct this problem, renumber the IDs or change the way hashing is implemented to prevent VLAN and VRRP IDs from hashing to the same location. To configure hashing to use different locations, change the hash variant to **m1**.

```
vlan create ip0 id 1
vlan create ip1 id 2
vlan create ip2 id 3
vlan create ip3 id 4
vlan create ip4 id 5
. . .
interface create ip ip1 address-netmask 10.0.0.100/24 vlan ip1
interface create ip ip2 address-netmask 10.0.0.101/24 vlan ip2
interface create ip ip3 address-netmask 10.0.0.102/24 vlan ip3
interface create ip ip4 address-netmask 10.0.0.103/24 vlan ip4
interface create ip ip5 address-netmask 10.0.0.104/24 vlan ip5
. . .
ip-redundancy create vrrp 1 interface ip1
ip-redundancy create vrrp 2 interface ip2
ip-redundancy create vrrp 3 interface ip3
ip-redundancy create vrrp 4 interface ip4
ip-redundancy create vrrp 5 interface ip5
. . .
ip-redundancy start vrrp 1 interface ip1
ip-redundancy start vrrp 2 interface ip2
ip-redundancy start vrrp 3 interface ip3
ip-redundancy start vrrp 4 interface ip4
ip-redundancy start vrrp 5 interface ip5
. . .
port set all-ports hash-mode m1
```

Change hash variant to m1

Additional Configuration

This section covers settings you can modify in a VRRP configuration, including backup priority, advertisement interval, pre-empt mode, and authentication key.

Setting the Backup Priority

As described in *Multi-Backup Configuration* on page 206, you can specify which Backup router takes over when the Master router goes down by setting the priority for the Backup routers. To set the priority for a Backup router, enter the following command in Configure mode:

To specify 200 as the priority used by virtual router 1 on interface int1:

```
xp(config)# ip-redundancy set vrrp 1 interface int1 priority 200
```

The priority can be between 1 (lowest) and 254. The default is 100. The priority for the IP address owner is 255 and cannot be changed.

Setting the Warm-up Period

When the Master router goes down, the Backup router takes over. When an interface comes up, the Master router may become available and take over from the Backup router. Before the Master router takes over, it may have to update its routing tables. You can specify a warm-up period, in seconds, during which the Master router can update its routing information before it preempts the existing Master router.

To specify a warm-up period for a Master router before it takes over:

```
xp(config)# ip-redundancy set vrrp 1 warmup-period 20
```

The warm-up period can be between 1 and 180 seconds. The default is 30 seconds.

Setting the Advertisement Interval

The VRRP Master router sends periodic advertisement messages to let the other routers know that the Master is up and running. By default, advertisement messages are sent once each second. To change the VRRP advertisement interval, enter the following command in Configure mode:

To set the advertisement interval to 3 seconds:

```
xp(config)# ip-redundancy set vrrp 1 interface int1 adv-interval 3
```

Setting Pre-empt Mode

When a Master router goes down, the Backup with the highest priority takes over the IP addresses associated with the Master. By default, when the original Master comes back up again, it takes over from the Backup router that assumed its role as Master. When a VRRP router does this, it is said to be in *pre-empt mode*. Pre-empt mode is enabled by default on the X-Pedition router. You can prevent a VRRP router from taking over from a lower-priority Master by disabling pre-empt mode. To do this, enter the following command in Configure mode:

To prevent a Backup router from taking over as Master from a Master router that has a lower priority:

```
xp(config)# ip-redundancy set vrrp 1 interface int1 preempt-mode disabled
```

Note: If the IP address owner is available, it will always take over as the Master—regardless of whether pre-empt mode is on or off.

Setting an Authentication Key

By default, no authentication of VRRP packets is performed on the X-Pedition router. You can specify a clear-text password to be used to authenticate VRRP exchanges. To enable authentication, enter the following command in Configure mode

To authenticate VRRP exchanges on virtual router 1 on interface int1 with a password of ‘yago’:

```
xp(config)# ip-redundancy set vrrp 1 interface int1 auth-type text auth-key yago
```

Note: The X-Pedition router does not currently support the IP Authentication Header method of authentication.

Setting ICMP Response

When a backup VRRP router assumes mastership, RFC 2338 specifies that it must not answer to ICMP echo requests (pings) destined to the associated virtual address. In some network situations, however, it may be desirable to have the ability to allow the backup router respond with an ICMP Echo Response when it is in the Master state. This feature allows the configuration of such a response. Use the following command to enable ICMP Echo Response:

ip-redundancy set vrrp <*vrID*> **interface** <*interface*> **icmp-response**

Monitoring VRRP

The X-Pedition router provides two commands for monitoring a VRRP configuration: **ip-redundancy trace**, which displays messages when VRRP events occur, and **ip-redundancy show**, which reports statistics about virtual routers.

ip-redundancy trace

The **ip-redundancy trace** command is used for troubleshooting purposes. This command causes messages to be displayed when certain VRRP events occur on the X-Pedition router. To trace VRRP events, enter the following commands in Enable mode:

Display a message when any VRRP event occurs. (Disabled by default.)	**ip-redundancy trace vrrp events enabled**
Display a message when a VRRP router changes from one state to another; for example Backup to Master. (Enabled by default.)	**ip-redundancy trace vrrp state-transitions enabled**
Display a message when a VRRP packet error is detected. (Enabled by default.)	**ip-redundancy trace vrrp packet-errors enabled**
Enable all VRRP tracing.	**ip-redundancy trace vrrp all enabled**

ip-redundancy show

The **ip-redundancy show** command reports information about a VRRP configuration.

To display information about all virtual routers on interface int1:

```
xp# ip-redundancy show vrrp interface int1

VRRP Virtual Router 100 - Interface int1
----------------------------------------
Uptime               0 days, 0 hours, 0 minutes, 17 seconds.
State              Backup
Priority            100 (default value)
Virtual MAC address   00005E:000164
Advertise Interval    1 sec(s) (default value)
Preempt Mode         Enabled (default value)
Authentication       None (default value)
Primary Address      10.8.0.2
Associated Addresses  10.8.0.1
                    100.0.0.1

VRRP Virtual Router 200 - Interface int1
----------------------------------------
Uptime               0 days, 0 hours, 0 minutes, 17 seconds.
State              Master
Priority            255 (default value)
Virtual MAC address   00005E:0001C8
Advertise Interval    1 sec(s) (default value)
Preempt Mode         Enabled (default value)
Authentication       None (default value)
Primary Address      10.8.0.2
Associated Addresses  10.8.0.2
```

To display VRRP statistics for virtual router 100 on interface int1:

```
xp# ip-redundancy show vrrp 1 interface int1 verbose

VRRP Virtual Router 100 - Interface int1
-----------------------------------------
Uptime                0 days, 0 hours, 0 minutes, 17 seconds.
State                 Backup
Priority              100 (default value)
Virtual MAC address   00005E:000164
Advertise Interval    1 sec(s) (default value)
Preempt Mode          Enabled (default value)
Authentication        None (default value)
Primary Address       10.8.0.2
Associated Addresses  10.8.0.1
                      100.0.0.1

Stats:
 Number of transitions to master state          2
 VRRP advertisements rcvd                       0
 VRRP packets sent with 0 priority              1
 VRRP packets rcvd with 0 priority              0
 VRRP packets rcvd with IP-address list mismatch 0
 VRRP packets rcvd with auth-type mismatch      0
 VRRP packets rcvd with checksum error          0
 VRRP packets rcvd with invalid version         0
 VRRP packets rcvd with invalid VR-Id           0
 VRRP packets rcvd with invalid adv-interval    0
 VRRP packets rcvd with invalid TTL             0
 VRRP packets rcvd with invalid 'type' field    0
 VRRP packets rcvd with invalid auth-type       0
 VRRP packets rcvd with invalid auth-key        0
```

To display VRRP information, enter the following commands in Enable mode.

Display information about all virtual routers.	**ip-redundancy show vrrp**

VRRP Configuration Notes

- The X-Pedition router supports only 512 instances of VRRP. An instance is defined as one virtual router running on one interface. Running a single virtual router on four interfaces is considered four instances of VRRP, as is running four virtual routers on a single interface.
- Do not use an IP address for VRRP that is already configured for load-balancing.
- The Master router sends keep-alive advertisements. The frequency of these keep-alive advertisements is determined by setting the Advertisement interval parameter. The default value is 1 second.
- If a Backup router doesn't receive a keep-alive advertisement from the current Master within a certain period of time, it will transition to the Master state and start sending advertisements itself. The amount of time that a Backup router will wait before it becomes the new Master is based on the following equation:

  Master-down-interval = (3 * advertisement-interval) + skew-time

  The skew-time depends on the Backup router's configured priority:

  Skew-time = ((256 - Priority) / 256)

  Therefore, the higher the priority, the faster a Backup router will detect that the Master is down. For example:

  – Default advertisement-interval = 1 second

  – Default Backup router priority = 100

  – Master-down-interval = time it takes a Backup to detect the Master is down

    = (3 * adv-interval) + skew-time

    = (3 * 1 second) + ((256 - 100) / 256)

    = 3.6 seconds

  Note: In some instances, a heavy load on a VRRP master may delay VRRP packet transmission and cause the backup router to assume the role of master.
- If a Master router is manually rebooted, or if its interface is manually brought down, it will send a special keep-alive advertisement that lets the Backup routers know that a new Master is needed immediately.
- A virtual router will respond to ARP requests with a virtual MAC address. This virtual MAC depends on the virtual router ID:

  virtual MAC address = 00005E:0001*XX*

  where *XX* is the virtual router ID

  This virtual MAC address is also used as the source MAC address of the keep-alive Advertisements transmitted by the Master router.

- These MAC addresses, when active, use entries in the port's Routing Address Table (RAT). Since the RAT has eight entries and other resources also use the RAT, the system will limit the number of unique virtual routing ID's to six or fewer on a per port basis.
- If multiple virtual routers are created on a single interface, the virtual routers must have unique identifiers. If virtual routers are created on different interfaces, you can reuse virtual router IDs.

  For example, the following configuration is valid:

```
ip-redundancy create vrrp 1 interface test-A
ip-redundancy create vrrp 1 interface test-B
```

- As specified in RFC 2338, a Backup router that has transitioned to Master will not respond to pings, accept telnet sessions, or field SNMP requests directed at the virtual router's IP address.

  Not responding allows network management to notice that the original Master router (i.e., the IP address owner) is down.

Chapter 15

RIP Configuration Guide

This chapter describes how to configure the Routing Information Protocol (RIP) on the Enterasys X-Pedition router. RIP is a distance-vector routing protocol for use in small networks. RIP is described in RFC 2453. A router running RIP broadcasts updates at set intervals. Each update contains paired values where each pair consists of an IP network address and an integer distance to that network. RIP uses a hop count metric to measure the distance to a destination.

The Enterasys X-Pedition router provides support for RIP Version 1 and 2. The X-Pedition router implements plain text and MD5 authentication methods for RIP Version 2.

The protocol independent features that apply to RIP are described in Chapter 12, *IP Routing Configuration Guide*.

Note: For information on RIPng, refer to Chapter 16, *RIPng Configuration Guide*.

Configuring RIP

By default, RIP is disabled on the X-Pedition router and on each of the attached interfaces. To configure RIP on the X-Pedition router, follow these steps:

1. Start the RIP process by entering the **rip start** command.
2. Use the **rip add interface** command to inform RIP about the attached interfaces.

Enabling and Disabling RIP

To enable or disable RIP, enter one of the following commands in Configure mode.

Enable RIP.	**rip start**
Disable RIP.	**rip stop**

Configuring RIP Interfaces

To configure RIP in the X-Pedition router, you must first add interfaces to inform RIP about attached interfaces. To add RIP interfaces, enter the following commands in Configure mode.

Add interfaces to the RIP process.	**rip add interface** <*interfacename-or-IPaddr*>
Add gateways from which the X-Pedition router will accept RIP updates.	**rip add trusted-gateway** <*interfacename-or-IPaddr*>
Define the list of routers to which RIP sends packets directly, not through multicast or broadcast.	**rip add source-gateway** <*interfacename-or-IPaddr*>

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring RIP Parameters

No further configuration is required, and the system default parameters will be used by RIP to exchange routing information. These default parameters may be modified to suit your needs by using the **rip set interface** command.

RIP Parameter	Default Value
Version number	RIP v1
Check-zero for RIP reserved parameters	Enabled
Whether RIP packets should be broadcast	Choose
Preference for RIP routes	100
Metric for incoming routes	1
Metric for outgoing routes	0
Authentication	None
Update interval	30 seconds

To change RIP parameters, enter the following commands in Configure mode.

Set RIP Version on an interface to RIP V1.	**rip set interface** <*interfacename-or-IPaddr*>**|all version 1**
Set RIP Version on an interface to RIP V2.	**rip set interface** <*interfacename-or-IPaddr*>**|all version 2**
Specify that RIP V2 packets should be multicast on this interface.	**rip set interface** <*interfacename-or-IPaddr*>**|all type multicast**
Specify that RIP V2 packets that are RIP V1-compatible should be broadcast on this interface.	**rip set interface** <*interfacename-or-IPaddr*>**|all type broadcast**
Change the metric on incoming RIP routes.	**rip set interface** <*interfacename-or-IPaddr*>**|all metric-in** <*num*>
Change the metric on outgoing RIP routes.	**rip set interface** <*interfacename-or-IPaddr*>**|all metric-out** <*num*>
Set the authentication method to simple text up to 8 characters.	**rip set interface** <*interfacename-or-IPaddr*>**|all authentication-method simple**
Set the authentication method to MD5.	**rip set interface** <*interfacename-or-IPaddr*>**|all authentication-method md5**
Specify the metric to be used when advertising routes that were learned from other protocols.	**rip set default-metric** <*num*>
Enable automatic summarization and redistribution of RIP routes. **Note:** The **rip set auto-summary** command must be enabled if the router will act as a border gateway using RIP Version 1.	**rip set auto-summary disable|enable**
Specify broadcast of RIP packets regardless of number of interfaces present.	**rip set broadcast-state always|choose|never**
Check that reserved fields in incoming RIP V1 packets are zero.	**rip set check-zero disable|enable**
Enable acceptance of RIP routes that have a metric of zero.	**rip set check-zero-metric disable|enable**
Enable poison reverse, as specified by RFC 1058.	**rip set poison-reverse disable|enable**

Specify the maximum number of RIP routes maintained in the routing information base (RIB). The default is 4.	**rip set max-routes** <*number*>
Disable multipath route calculation for RIP routes.	**Rip set multipath off**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring RIP Route Preference

You can set the preference of routes learned from RIP.

To configure RIP route preference, enter the following command in Configure mode.

Set the preference of routes learned from RIP.	**rip set preference** <*num*>

Configuring RIP Route Default-Metric

You can define the metric used when advertising routes via RIP that were learned from other protocols. The default value for this parameter is 16 (unreachable). To export routes from other protocols into RIP, you must explicitly specify a value for the default-metric parameter. The metric specified by the default-metric parameter may be overridden by a metric specified in the export command.

To configure default-metric, enter the following command in Configure mode.

Define the metric used when advertising routes via RIP that were learned from other protocols.	**rip set default-metric** <*num*>

For <*num*>, you must specify a number between 1 and 16.

Monitoring RIP

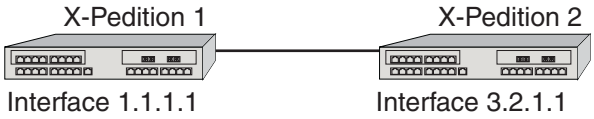
The *rip trace* command can be used to trace all rip request and response packets.

To monitor RIP information, enter the following commands in Enable mode.

Show all RIP information.	**rip show all**
Show RIP export policies.	**rip show export-policy**
Show RIP global information.	**rip show globals**
Show RIP import policies.	**rip show import-policy**
Show RIP information on the specified interface.	**rip show interface** *<Name or IP-addr>*
Show RIP interface policy information.	**rip show interface-policy**
Show detailed information of all RIP packets.	**rip trace packets detail**
Show detailed information of all packets received by the router.	**rip trace packets receive**
Show detailed information of all packets sent by the router.	**rip trace packets send**
Show detailed information of all request received by the router.	**rip trace request receive**
Show detailed information of all response received by the router.	**rip trace response receive**
Show detailed information of response packets sent by the router.	**rip trace response send**
Show RIP timer information.	**rip show timers**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuration Example




```
! Example configuration
!
! Create interface X-Pedition1-if1 with ip address 1.1.1.1/16 on port et.1.1 on X-Pedition1
interface create ip X-Pedition1-if1 address-netmask 1.1.1.1/16 port et.1.1
!
! Configure rip on X-Pedition1
rip add interface X-Pedition1-if1
rip set interface X-Pedition1-if1 version 2
rip start
!
!
! Set authentication method to md5
rip set interface X-Pedition1-if1 authentication-method md5
!
! Change default metric-in
rip set interface X-Pedition1-if1 metric-in 2
!
! Change default metric-out
rip set interface X-Pedition1-if1 metric-out 3
```

Chapter 16

RIPng Configuration Guide

This chapter describes how to configure the Routing Information Protocol Next Generation (RIPng) on the Enterasys X-Pedition router for use with IPv6.

The Enterasys X-Pedition router also provides support for RIP Version 1 and 2 for IPv4. Refer to Chapter 15, *RIP Configuration Guide* for configuration information for RIP v1 and v2.

IPv6 configuration information is provided in Chapter 13, *IPv6 Configuration Guide*.

RIPng Overview

RIPng is described in RFC 2080. Most of the concepts of RIPng come from RIPv1, described in RFC 1058, and RIPv2, described in RFC 2453.

RIPng is a simple extension of the RIP protocol to support IPv6 prefixes. RIPng functions much like RIPv2, using a distance vector scheme to find the shortest path through a network. RIPng tries to avoid loops by using split-horizon and poison-reverse, if configured. RIPng uses the counting-to-infinity method of discovering loops in certain situations. It listens on UDP port 521 for RIPng IPv6 packets, and communicates with IPv6 routers and hosts using the All-RIP-Routers Multicast Address (FF02::9) and unicast destinations.

Configuring RIPng

By default, RIPng is disabled on the X-Pedition router and on each of the attached interfaces. To configure RIPng on the X-Pedition router, follow these steps:

1. Start the RIPng process by entering the **ripng start** command.
2. Use the **ripng add interface** command to enable RIPng on one or more attached interfaces.
3. Optionally, change the default RIPng interface parameters, using the **ripng set interface** command.
4. Optionally, change the default global RIPng parameter values, using the appropriate **ripng set** commands.

Enabling and Disabling RIPng

To enable or disable RIP, enter the following commands in Configure mode.

Enable RIPng	**ripng start**
Disable RIPng	**no ripng start**

Configuring RIPng Interfaces

To configure RIPng in the X-Pedition router, you must first add interfaces to inform RIPng about attached interfaces. After adding RIPng interfaces, you can change the default values of the interface parameters, if desired. Enter the following commands in Configure mode.

Add interfaces to the RIPng process	**ripng add interface** <*interfacename*>| **all**
Optionally, change default values of interface parameters	**ripng set interface** <*interfacename*>| **all** [*option list*]

The following table lists the interface parameters that can be changed using the options of the **ripng set interface** command.

RIPng Interface Parameter	Default Value	Command Option
Whether the interface will accept RIPng updates	Enabled	**receive-rip** {**enable**	**disable**}
Whether the interface will send RIPng updates	Enabled	**send-rip** {**enable**	**disable**}

RIPng Interface Parameter	Default Value	Command Option
Metric to be added to incoming RIPng updates	1	**metric-in** <*num*> (<*num*> can range from 1 -16)
Metric to be added to outgoing RIPng updates	0	**metric-out** <*num*> (<*num*> can range from 1 - 16)

Configuring Global RIPng Parameters

You can also change the following global RIPng parameters using the listed commands in Configure mode.

RIPng Global Parameter	Command
Specify the metric to be used when advertising routes via RIPng that were learned from other protocols. Default is 1. Range is from 1 to 16.	**ripng set default-metric** <*num*>
Specify the expiration time for routes received via RIPng. Default is 180 seconds. Range is 5 to 180.	**ripng set expire-time** <*num*>
Enable poison reverse (see RFC 1058). Default is disabled.	**ripng set poison-reverse disable|enable**
Specify the preference for routes learned through RIPng. Default is 100. Range is 0 to 255.	**ripng set preference** <*num*>
Specify the update time for unsolicited route responses. Default is 30 seconds. Range is 1 to 30.	**ripng set update-time** <*num*>

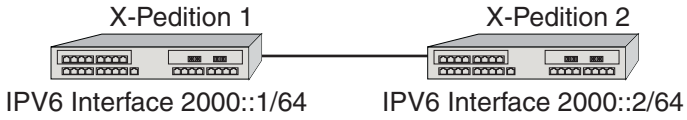
Monitoring RIPng

The *ripng trace* command can be used to trace all rip request and response packets.

To monitor RIPng information, enter the following commands in Enable mode.

Show all RIPng information.	**ripng show all**
Show RIPng global information.	**ripng show globals**
Show RIPng timer information.	**ripng show timers**
Show RIPng information on the specified interface.	**ripng show interface** *<Name or IPv6addr>*
Show active gateways running RIPng.	**ripng show active-gateways**
Show RIPng interface policy information.	**ripng show interface-policies**
Show RIPng import policies.	**ripng show import-policies**
Show RIPng export policies.	**ripng show export-policies**
Show detailed information of all RIPng packets.	**ripng trace packets detail**
Show detailed information of all packets received by router.	**ripng trace packets receive**
Show detailed information of all packets sent by router.	**ripng trace packets send**
Show detailed information of all request packets received by the router.	**ripng trace request receive**
Show details information of request packets sent by the router.	**ripng trace request send**
Show detailed information of all response packets received by the router.	**ripng trace response receive**
Show detailed information of response packets sent by the router.	**ripng trace response send**

Configuration Example




```
! Example configuration
!
! Create interface intf1 with ipv6 address 2000::1/64 on port et.1.1 on X-Pedition1
interface create ipv6 intf1 address-prefix 2000::1/64 port et.1.1
!
! Configure ripng on X-Pedition1
ripng add interface intf1
ripng start
!
! Change default metric
ripng set default-metric 2
!
! Enable poison reverse
ripng set poison-reverse enable
```

The interface in the example could be replaced with an IPv6-in-IPv4 tunnel interface.

Static and direct routes can be redistributed via RIPng using the **ipv6-router policy redistribute** command. Refer to the *X-Pedition NATIVE Command Line Interface Reference Manual* for a description of the **ipv6-router policy redistribute** command.

Chapter 17

OSPF Configuration Guide

OSPF Overview

Open Shortest Path First Routing (OSPF) is a shortest path first or link-state protocol. The X-Pedition router supports OSPF Version 2.0, as defined in RFC 2328. OSPF is an interior gateway protocol that distributes routing information between routers in a single autonomous system. OSPF chooses the least-cost path as the best path. OSPF is suitable for complex networks with a large number of routers because it provides equal-cost multi-path routing where packets to a single destination can be sent via more than one interface simultaneously.

In a link-state protocol, each router maintains a database that describes the entire AS topology, which it builds out of the collected link state advertisements of all routers. Each participating router distributes its local state (i.e., the router's usable interfaces and reachable neighbors) throughout the AS by flooding. Each multi-access network that has at least two attached routers has a designated router and a backup designated router. The designated router floods a link state advertisement for the multi-access network and has other special responsibilities. The designated router concept reduces the number of adjacencies required on a multi-access network.

OSPF allows networks to be grouped into areas. Routing information passed between areas is abstracted, potentially allowing a significant reduction in routing traffic. OSPF uses four different types of routes, listed in order of preference:

- Intra-area
- Inter-area
- Type 1 ASE
- Type 2 ASE

Intra-area paths have destinations within the same area. Inter-area paths have destinations in other OSPF areas. Both types of Autonomous System External (ASE) routes are routes to destinations external to OSPF (and usually external to the AS). Routes exported into OSPF ASE as type 1 ASE routes are supposed to be from interior gateway protocols (e.g., RIP) whose external metrics are directly comparable to OSPF metrics. When a routing decision is being made, OSPF will add the internal cost to the AS border router to the external metric. Type 2 ASEs are used for exterior gateway protocols whose metrics are not comparable to OSPF metrics. In this case, only the external OSPF cost from the AS border router is used in the routing decision.

The X-Pedition router supports the following OSPF functions:

- Stub Areas: Definition of stub areas is supported.
- NSSA: Definition of Not So Stubby Areas (NSSAs) is supported.
- Authentication: Simple password and MD5 authentication methods are supported within an area.
- Virtual Links: Virtual links are supported.
- Route Redistribution: Routes learned via RIP, BGP, or any other sources can be redistributed into OSPF. OSPF routes can be redistributed into RIP or BGP.
- Interface Parameters: Parameters that can be configured include interface output cost, retransmission interval, interface transit delay, router priority, router dead and hello intervals, and authentication key.

OSPF Multipath

The X-Pedition router also supports OSPF and static Multi-path. If multiple equal-cost OSPF or static routes have been defined for any destination, then the X-Pedition router “discovers” and uses all of them. The X-Pedition router will automatically learn up to four equal-cost OSPF or static routes and retain them in its forwarding information base (FIB). The forwarding module then installs flows for these destinations in a round-robin fashion.

Configuring OSPF

To configure OSPF on the X-Pedition router, you must enable OSPF, create OSPF areas, assign interfaces to OSPF areas, and, if necessary, specify any of the OSPF interface parameters.

To configure OSPF, you may need to perform some or all of the following tasks:

- Enable OSPF.
- Create OSPF areas.
- Create an IP interface or assign an IP interface to a VLAN.
- Add IP interfaces to OSPF areas.
- Configure OSPF interface parameters, if necessary.
- Add IP networks to OSPF areas.
- Create virtual links, if necessary.

Enabling OSPF

OSPF is disabled by default on the X-Pedition router. To enable or disable OSPF, enter one of the following commands in Configure mode.

Enable OSPF.	**ospf start**
Disable OSPF.	**ospf stop**

Configuring OSPF Interface Parameters

You can configure the OSPF interface parameters shown in the table below.

Table 9. OSPF Interface Parameters

OSPF Parameter	Default Value
Interface OSPF State (Enable/Disable)	Enable (except for virtual links)
Cost	See *Default Cost of an OSPF Interface* below.
No multicast	Default is using multicast mechanism.
Retransmit interval	5 seconds
Transit delay	1 second
Priority	1
Hello interval	10 seconds (broadcast), 30 (non broadcast)
Router dead interval	4 times the hello interval

Table 9. OSPF Interface Parameters (Continued)

OSPF Parameter	Default Value
Poll Interval	120 seconds
Key chain	N/A
Authentication Method	None

Default Cost of an OSPF Interface

The default cost of an OSPF interface is calculated using its bandwidth. A VLAN that is attached to an interface could have several ports of differing speeds. The bandwidth of an interface is represented by the highest bandwidth port that is part of the associated VLAN. The cost of an OSPF interface is inversely proportional to this bandwidth. The cost is calculated using the following formula:

Cost = 2000000000 / *speed (in bps)*

Table 10 lists the port types and the OSPF default cost associated with each type.

Table 10. OSPF Default Cost Per Port Type

Port Media Type	Speed	OSPF Default Cost
Ethernet 1000	1000 Mbps	2
Ethernet 10/100	100 Mbps	20
Ethernet 10/100	10 Mbps	200
WAN (T1)	1.5 Mbps	1333
WAN (T3)	45 Mbps	44

To configure OSPF interface parameters, enter one of the following commands in Configure mode:

Enable OSPF state on interface.	**ospf set interface** <*name-or-IPaddr*>**|all state disable|enable**
Specify the cost of sending a packet on an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all cost** <*num*>
Specify the priority for determining the designated router on an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all priority** <*num*>
Specify the interval between OSPF hello packets on an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all hello-interval** <*num*>
Configure the retransmission interval between link state advertisements for adjacencies belonging to an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all retransmit-interval** <*num*>
Specify the number of seconds required to transmit a link state update on an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all transit-delay** <*num*>
Specify the time a neighbor router will listen for OSPF hello packets before declaring the router down.	**ospf set interface** <*name-or-IPaddr*>**|all router-dead-interval** <*num*>
Disable IP multicast for sending OSPF packets to neighbors on an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all no-multicast**
Specify the poll interval on an OSPF interface.	**ospf set interface** <*name-or-IPaddr*>**|all poll-interval** <*num*>
Specify the identifier of the key chain containing the authentication keys.	**ospf set interface** <*name-or-IPaddr*>**|all key-chain** <*num-or-string*>
Specify the authentication method to be used on this interface.	**ospf set interface** <*name-or-IPaddr*>**|all authentication-method none|simple|md5**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring an OSPF Area

OSPF areas are a collection of subnets that are grouped in a logical fashion. These areas communicate with other areas via the backbone area. Once OSPF areas are created, you can add interfaces, stub hosts, and summary ranges to the area.

To reduce the amount of routing information propagated between areas, you can configure summary-ranges on Area Border Routers (ABRs). On the X-Pedition router, summary-ranges are created using the **ospf add summary-range** command—the networks specified using this command describe the scope of an area. Intra-area Link State Advertisements (LSAs) that fall within the specified ranges are not advertised into other areas as inter-area routes. Instead, the specified ranges are advertised as summary network LSAs.

Note: Although this does not apply to most changes to OSPF and other routing-based entries in the configuration file, the following actions force the OSPF Link State Databases (LSDB) to re-initialize:

- Adding a network to or removing one from an area.
- Changing an area's type.
- Adding a summary range to or removing one from an Area Border Router.

To create areas and assign interfaces, enter the following commands in the Configure mode.

Create an OSPF area.	**ospf create area** <*area-num*>**|backbone**
Add an interface to an OSPF area.	**ospf add interface** <*interfacename-or-IPaddr*> **to-area** <*area-addr*> **|backbone [type broadcast|non-broadcast| point-to-multipoint]**
Add a stub host to an OSPF area.	**ospf add stub-host** <*IPaddr*> **to-area** <*area-addr*> **| backbone cost** <*num*>
Add a network to an OSPF area for summarization.	**ospf add network|summary-range** <*IPaddr/mask*> **to-area** <*area-addr*> **[restrict] [host-net]**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring OSPF Area Parameters

The X-Pedition router allows configuration of various OSPF area parameters, including stub areas, stub cost, authentication method, and not-so-stubby areas (NSSAs). Information about routes which are external to the OSPF routing domain is not sent into a stub area. Instead, there is a default external route generated by the ABR into the stub area for destinations outside the OSPF routing domain.

Stub cost specifies the cost to be used to inject a default route into a stub area. An authentication method for OSPF packets can be specified on a per-area basis. To configure OSPF area parameters, enter the following commands in the Configure mode.

Specify an OSPF area to be a stub area.	**ospf set area** <*area-num*> **stub**
Specify the cost to be used to inject a default route into a stub area. **Note:** If this command is not specified, no default route is injected into the OSPF stub area.	**ospf set area** <*area-num*> **stub-cost** <*num*>
Specify the authentication method to be used by neighboring OSPF routers.	**ospf set area** <*area-num*> **[stub] [authentication-method none|simple|md5]**

Note: Although this does not apply to most changes to OSPF and other routing-based entries in the configuration file, the following actions force the OSPF Link State Databases (LSDB) to re-initialize:

- Adding a network to or removing one from an area.
- Changing an area's type.
- Adding a summary range to or removing one from an Area Border Router.

Creating Virtual Links

In OSPF, virtual links can be established:

- To connect an area via a transit area to the backbone
- To create a redundant backbone connection via another area

Each Area Border Router must be configured with the same virtual link. Note that virtual links cannot be configured through a stub area.

To configure virtual links, enter the following commands in the Configure mode.

Create a virtual link.	**ospf add virtual-link** <*number-or-string*> **neighbor** <*IPaddr*> **transit-area** <*area-num*>
Set virtual link parameters.	**ospf set virtual-link** <*number-or-string*> [**no-multicast**] [**retransmit-interval** <*num*>] [**transit-delay** <*num*>] [**priority** <*num*>] [**hello-interval** <*num*>] [**router-dead-interval** <*num*>] [**poll-interval** <*num*>] **[key-chain** <*num*>] [**authentication-method none**| **simple**| **md5**] [**passive**] [**advertise-subnet on**|**off**]

Configuring Autonomous System External (ASE) Link Advertisements

An autonomous system boundary router (ASBR) advertises external destinations throughout the OSPF autonomous system. In many cases, external link states make up a large percentage of the link states in the databases of every router. A stub area is an area in which you do not allow advertisements of external routes, thus reducing the size of the database even more. Instead, a default summary route (0.0.0.0) is inserted into the stub area in order to reach these external routes. The following table lists the restrictions associated with each type of area:

Area	**Restrictions**
Normal	None
Stub area	No Type 5 AS-external LSA allowed
Totally Stub area	No Type 3, 4 or 5 LSAs allowed except the default summary route
NSSA	No Type 5 AS-external LSAs allowed, but Type 7 LSAs that convert to Type 5 at the NSSA ABR can traverse
NSSA Totally Stub area	No Type 3, 4 or 5 LSAs except the default summary route, but Type 7 LSAs that convert to Type 5 at the NSSA ABR are allowed

The X-Pedition router generates and floods a batch of ASE link state advertisements into OSPF one time per second. Each batch contains 100 ASEs. To specify AS external link advertisements parameters, enter the following command in the Configure mode:

Specifies AS external link advertisement default parameters.	**ospf set ase-defaults {**[**preference** <*num*>]	[**cost** <*num*>]	[**type** <*num*>]	[**inherit-metric**]	[**tag** [**as**] <*num*>]	[**multicast**]**}**

Configuring OSPF for Different Types of Interfaces

The X-Pedition router can run OSPF over a variety of physical connections: Serial connections, LAN interfaces, ATM, or Frame Relay. The OSPF configuration supports four different types of interfaces.

- LAN. An example of a LAN interface is an Ethernet. The X-Pedition router will use multicast packets on LAN interfaces to reach other OSPF routers. By default, an IP interface attached to a VLAN that contains LAN ports is treated as an OSPF broadcast network.
- Point-to-Point. A point-to-point interface can be a serial line using PPP. By default, an IP interface associated with a serial line that is using PPP is treated as an OSPF point-to-point network. If an IP interface that is using PPP is to be treated as an OSPF broadcast network, then use the **type broadcast** option of the **interface create** command.
- Non-Broadcast Multiple Access (NBMA). An example of a NBMA network is a fully-meshed Frame Relay or ATM network with virtual circuits. Because there is no general multicast for these networks, each neighboring router that is reachable over the NBMA network must be specified, so that routers can poll each other. The X-Pedition router will unicast packets to the routers in the NBMA network.
- Point-to-Multipoint (PMP). Point-to-multipoint connectivity is used when the network does not provide full connectivity to all routers in the network. As in the case of NBMA networks, a list of neighboring routers reachable over a PMP network should be configured so that the router can discover its neighbors.

To configure OSPF for NBMA networks, enter the following command in Configure mode:

Specify an OSPF NBMA neighbor.	**ospf add nbma-neighbor** <*IPaddr*> **to-interface** <*interfacename-or-IPaddr*> [**eligible**]

Note: When you assign an interface with the **ospf add interface** command, you must specify **type non-broadcast**.

Note: The X-Pedition router displays interface names up to 32 characters in length.

To configure OSPF for point-to-multipoint networks, enter the following command in Configure mode:

Specify an OSPF point-to-multipoint neighbor.	**ospf add pmp-neighbor** <*IPaddr*> **to-interface** <*interfacename-or-IPaddr*>

Note: When you assign an interface with the **ospf add interface** command, you must specify **type point-to-multipoint** (instead of **type non-broadcast)**.

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Displaying OSPF Information

The **ospf show** commands allow you to display detailed versions of the various OSPF tables. The **ospf show** commands can only display OSPF tables for the router on which the commands are being entered.

To display OSPF information, enter the following commands in Enable mode.

Show the link state database.	**ospf show lsdb area** <*ipAddr*> | **backbone** | **all**
Show information about all the OSPF routing neighbors.	**ospf show neighbors**
Show OSPF Autonomous System External database entries.	**ospf show as-external-lsdb**
Show all OSPF tables.	**ospf show all** [**to-file** | **to-terminal**]
Show all OSPF areas.	**ospf show areas**
Show OSPF errors.	**ospf show errors** <*interface*>
Show information about OSPF export policies.	**ospf show export-policies**
Show OSPF routes.	**ospf show routes**
Show all OSPF global parameters.	**ospf show globals**
Show information about OSPF import policies.	**ospf show import-policies**
Show OSPF interfaces.	**ospf show interfaces**
Show OSPF statistics.	**ospf show statistics** <*interface*>
Show OSPF timers.	**ospf show timers**
Show OSPF virtual-links.	**ospf show virtual-links**

Show link state advertisement information	**ospf show lsa area-id** <*IPaddr*> **type router-links** | **network-links** | **summary-networks** | **summary-asb**| **as-external ls-id** <*IPaddr*> **adv-rtr** <*IPaddr*>
Show enabled OSPF traps	**ospf show enabled-traps**

OSPF Configuration Examples

For all examples in this section, refer to the configuration shown in Figure 23 on page 247.

The following configuration commands for router R1:

- Determine the IP address for each interface
- Specify the static routes configured on the router
- Determine its OSPF configuration

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Create the various IP interfaces.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 interface create ip to-r2 address-netmask 120.190.1.1/16 port et.1.2
 interface create ip to-r3 address-netmask 130.1.1.1/16 port et.1.3
 interface create ip to-r41 address-netmask 140.1.1.1/24 port et.1.4
 interface create ip to-r42 address-netmask 140.1.2.1/24 port et.1.5
 interface create ip to-r6 address-netmask 140.1.3.1/24 port et.1.6
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route 202.1.0.0/16 gateway 120.1.1.2
 ip add route 160.1.5.0/24 gateway 120.1.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! OSPF Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ospf start
 ospf create area 140.1.0.0
 ospf create area backbone
 ospf set ase-defaults cost 4
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! OSPF Interface Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ospf add interface 140.1.1.1 to-area 140.1.0.0
 ospf add interface 140.1.2.1 to-area 140.1.0.0
 ospf add interface 140.1.3.1 to-area 140.1.0.0
 ospf add interface 130.1.1.1 to-area backbone
```

Exporting All Interfaces and Static Routes to OSPF

Router R1 has several static routes. We would export these static routes as type-2 OSPF routes. The interface routes would be redistributed as type-1 OSPF routes. You may accomplish this using either of the following examples.

Example 1

1. Create a OSPF export destination for type-1 routes since we would like to redistribute certain routes into OSPF as type 1 OSPF-ASE routes.

```
ip-router policy create ospf-export-destination ospfExpDstType1 type 1 metric 1
```

2. Create a OSPF export destination for type-2 routes since we would like to redistribute certain routes into OSPF as type 2 OSPF-ASE routes.

```
ip-router policy create ospf-export-destination ospfExpDstType2 type 2 metric 4
```

3. Create a Static export source since we would like to export static routes.

```
ip-router policy create static-export-source statExpSrc
```

4. Create a Direct export source since we would like to export interface/direct routes.

```
ip-router policy create direct-export-source directExpSrc
```

5. Create the Export-Policy for redistributing all interface routes and static routes into OSPF.

```
ip-router policy export destination ospfExpDstType1 source directExpSrc network all
ip-router policy export destination ospfExpDstType2 source statExpSrc network all
```

Example 2

Rather than create two multiple exports and another command to use them (as in the previous example), the following commands will produce the same result:

```
ip-router policy redistribute from-proto direct to-proto ospf ase-type 1 metric 1 network all
ip-router policy redistribute from-proto static to-proto ospf ase-type 2 metric 4 network all
```

Exporting all RIP, Interface, and Static Routes to OSPF

Note: Also export interface, static, RIP, OSPF, and OSPF-ASE routes into RIP.

In the configuration shown in Figure 23 on page 247, RIP Version 2 is configured on the interfaces of routers R1 and R2, attached to the sub-network 120.190.0.0/16. Suppose you would like to redistribute these *RIP* routes as OSPF type-2 routes and associate the tag 100 with them. Furthermore, you would like to redistribute *static* routes from R1 as OSPF type-2 routes, *interface* routes as OSPF type-1 routes, and OSPF, OSPF-ASE, RIP, Static, and Interface/Direct routes into RIP. You may accomplish this change using either of the following examples.

Example 1

1. Enable RIP on interface 120.190.1.1/16.

```
rip add interface 120.190.1.1
rip set interface 120.190.1.1 version 2 type multicast
rip start
```

2. Create a OSPF export destination for type-1 routes.

```
ip-router policy create ospf-export-destination ospfExpDstType1 type 1 metric 1
```

3. Create a OSPF export destination for type-2 routes.

```
ip-router policy create ospf-export-destination ospfExpDstType2 type 2 metric 4
```

4. Create a OSPF export destination for type-2 routes with a tag of 100.

```
ip-router policy create ospf-export-destination ospfExpDstType2t100 type 2 tag 100 metric 4
```

5. Create a RIP export source.

```
ip-router policy create rip-export-source ripExpSrc
```

6. Create a Static export source.

```
ip-router policy create static-export-source statExpSrc
```

7. Create a Direct export source.

```
ip-router policy create direct-export-source directExpSrc
```

8. Create the Export-Policy for redistributing all interface, RIP and static routes into OSPF.

```
ip-router policy export destination ospfExpDstType1 source directExpSrc network all
ip-router policy export destination ospfExpDstType2 source statExpSrc network all
ip-router policy export destination ospfExpDstType2t100 source ripExpSrc network all
```

9. Create a RIP export destination.

```
ip-router policy create rip-export-destination ripExpDst
```

Note: When you use the *gateway* option in the **ip-router policy create rip-export-destination** command in conjunction with **ip-router policy export destination**, you must use **rip add source-gateways** for each address indicated in the gateway option.

10. Create an OSPF export source.

```
ip-router policy create ospf-export-source ospfExpSrc metric 1
```

11. Create an OSPF-ASE export source.

```
ip-router policy create ospfase-export-source ospfAseExpSrc metric 1
```

12. Create the Export-Policy for redistributing all interface, RIP, static, OSPF and OSPF-ASE routes into RIP.

```
ip-router policy export destination ripExpDst source statExpSrc network all
ip-router policy export destination ripExpDst source ripExpSrc network all
ip-router policy export destination ripExpDst source directExpSrc network all
ip-router policy export destination ripExpDst source ospfExpSrc network all
ip-router policy export destination ripExpDst source ospfAseExpSrc network all
```

Example 2

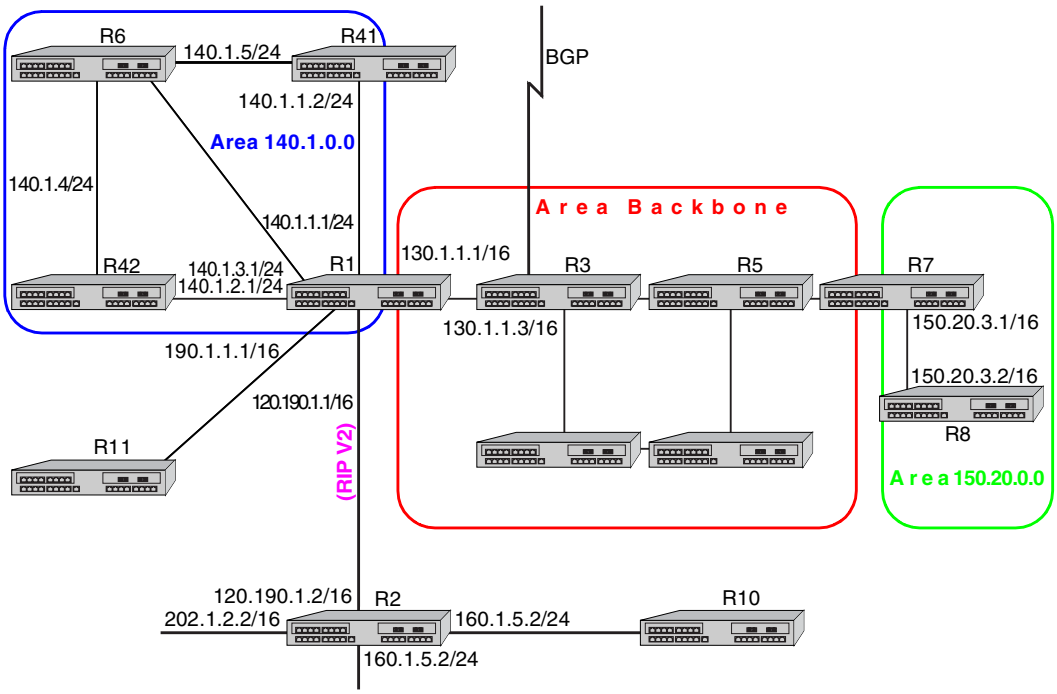
To simplify the process described in the previous example, the following commands produce the same result as steps 2-8:

```
ip-router policy redistribute from-proto direct to-proto ospf ase-type 1 metric 1 network all
ip-router policy redistribute from-proto static to-proto ospf ase-type 2 metric 4 network all
ip-router policy redistribute from-proto rip to-proto ospf ase-type 2 tag 100 metric 4 network all
```

To reproduce the result of steps 9-12:

```
ip-router policy redistribute from-proto static to-proto rip network all
ip-router policy redistribute from-proto rip to-proto rip network all
ip-router policy redistribute from-proto direct to-proto rip network all
ip-router policy redistribute from-proto ospf to-proto rip network all
ip-router policy redistribute from-proto ospf-ase to-proto rip network all
```

Figure 23. Exporting to OSPF



Chapter 18

BGP Configuration Guide

BGP Overview

The Border Gateway Protocol (BGP) is an exterior gateway protocol that allows IP routers to exchange network reachability information. BGP became an internet standard in 1989 (RFC 1105) and the current version, BGP-4, was published in 1994 (RFC 1771). BGP typically runs between Internet Service Providers. It is also frequently used by multi-homed ISP customers and large commercial networks.

Notes:

- BGP management traps are not supported.
- The X-Pedition router does not currently follow "Breaking Ties (Phase2)," Section 9.1.2.1 (p. 37-38) of RFC 1771. Instead, the router follows "Breaking Ties (Phase2)," Section 9.1.2.2 (p. 49-50) of Draft-ietf-ier-bgp-4-17.

Autonomous systems that wish to connect their networks must agree on a method of exchanging routing information. Interior gateway protocols such as RIP and OSPF may be inadequate for this task since they were not designed to handle multi-AS, policy, and security issues. Similarly, using static routes may not be the best choice for exchanging AS-AS routing information because there may be a large number of routes, or the routes may change often.

Note: This chapter uses the term *Autonomous System* (AS) throughout. An AS is defined as a set of routers under a central technical administration that has a coherent interior routing plan and accurately portrays to other ASs what routing destinations are reachable by way of it.

In an environment where using static routes is not feasible, BGP is often the best choice for an AS-AS routing protocol. BGP prevents the introduction of routing loops created by multi-homed and meshed AS topologies. BGP also provides the ability to create and enforce policies at the AS level, such as selectively determining which AS routes are to be accepted or what routes are to be advertised to BGP peers.

The X-Pedition BGP Implementation

The X-Pedition routing protocol implementation is based on GateD 4.0.3 code (www.gated.org). GateD is a modular software program consisting of core services, a routing database, and protocol modules supporting multiple routing protocols (RIP versions 1 and 2, OSPF version 2, BGP version 2 through 4, and Integrated IS-IS).

Since the X-Pedition IP routing code is based upon GateD, BGP can also be configured using a GateD configuration file (gated.conf) instead of the X-Pedition Command Line Interface (CLI). Additionally, even if the X-Pedition router is configured using the CLI, the gated.conf equivalent can be displayed by entering the **ip-router show configuration-file** command at the X-Pedition Enable prompt.

VLANs, interfaces, ACLs, and many other X-Pedition configurable entities and functionality can only be configured using the X-Pedition CLI. Therefore, a gated.conf file is dependent upon some X-Pedition CLI configuration.

Note: The **rip set auto-summary** command must be enabled if the X-Pedition router will act as a border gateway using RIP Version 1.

Basic BGP Tasks

This section describes the basic tasks necessary to configure BGP on the X-Pedition router. Due to the abstract nature of BGP, many BGP designs can be extremely complex. For any one BGP design challenge, there may only be one solution out of many that is relevant to common practice.

When designing a BGP configuration, it may be prudent to refer to information in RFCs, Internet drafts, and books about BGP. Some BGP designs may also require the aid of an experienced BGP network consultant.

Basic BGP configuration involves the following tasks:

- Setting the autonomous system number
- Setting the router ID
- Creating a BGP peer group
- Adding and removing a BGP peer host
- Starting BGP
- Using AS path regular expressions
- Using AS path prepend

Setting the Autonomous System Number

An autonomous system number identifies your autonomous system to other routers. To set the X-Pedition router's autonomous system number, enter the following command in Configure mode.

Set the X-Pedition router's autonomous system number.	**ip-router global set autonomous-system** <*num1*> **loops** <*num2*>

The a**utonomous-system** <*num1*> parameter sets the AS number for the router. Specify a number from 1–65534. The **loops** <*num2*> parameter controls the number of times the AS may appear in the as-path. The default is 1.

Setting the Router ID

The router ID uniquely identifies the X-Pedition router. To set the router ID to be used by BGP, enter the following command in Configure mode.

Set the X-Pedition router's router ID.	**ip-router global set router-id** <*hostname-or-IPaddr*>

If you do not explicitly specify the router ID, then an ID is chosen implicitly by the X-Pedition router. A secondary address on the loopback interface (the primary address being 127.0.0.1) is the most preferred candidate for selection as the X-Pedition router ID. If there are no secondary addresses on the loopback interface, the default router ID is set to the address of the first interface the X-Pedition router encounters in the up state (except the interface en0, which is the Control Module's interface). The address of a non point-to-point interface is preferred over the local address of a point-to-point interface. If the router ID is implicitly chosen to be the address of a non-loopback interface, and if that interface were to go down, then the router ID is changed. When the router ID changes, an OSPF router has to flush all its LSAs from the routing domain.

If you explicitly specify a router ID, then it would not change, even if all interfaces were to go down.

Configuring a BGP Peer Group

A BGP peer group is a group of neighbor routers that have the same update policies. To configure a BGP peer group, enter the following command in Configure mode:

Configure a BGP peer group.	**bgp create peer-group** <*number-or-string*> **type external** | **internal** | **routing** [**autonomous-system** <*number*>] [**proto any** | **rip** | **ospf** | **static**] [**interface** <*interface-name-or-ipaddr*> | **all**]

Note: The X-Pedition router displays interface names up to 32 characters in length.

where:

peer-group <*number-or-string*>
Is a group ID, which can be a number or a character string.

type Specifies the type of BGP group you are adding. You can specify one of the following:

external In the classic external BGP group, full policy checking is applied to all incoming and outgoing advertisements. The external neighbors must be directly reachable through one of the machine's local interfaces.

routing An internal group which uses the routes of an interior protocol to resolve forwarding addresses. Type Routing groups will determine the immediate next hops for routes by using the next hop received with a route from a peer as a forwarding address, and using this to look up an immediate next hop in an IGP's routes. Such groups support distant peers, but need to be informed of the IGP whose routes they are using to determine immediate next hops. This implementation comes closest to the IBGP implementation of other router vendors.

internal An internal group operating where there is no IP-level IGP, for example an SMDS network. Type Internal groups expect all peers to be directly attached to a shared subnet so that, like external peers, the next hops received in BGP advertisements may be used directly for forwarding. All Internal group peers should be L2 adjacent.

autonomous-system <*number*>
Specifies the autonomous system of the peer group. Specify a number from 1 –65534.

proto Specifies the interior protocol to be used to resolve BGP next hops. Specify one of the following:

any Use any igp to resolve BGP next hops.

rip Use RIP to resolve BGP next hops.

ospf Use OSPF to resolve BGP next hops.

static Use static to resolve BGP next hops.

interface <*name-or-IPaddr*> | **all**
Interfaces whose routes are carried via the IGP for which third-party next hops may be used instead. Use only for type Routing group. Specify the interface or **all** for all interfaces.

Note: The X-Pedition router displays interface names up to 32 characters in length.

Adding and Removing a BGP Peer

There are two ways to add BGP peers to peer groups. You can explicitly add a peer host, or you can add a network. Adding a network allows for peer connections from any addresses in the range of network and mask pairs specified in the **bgp add network** command.

To add BGP peers to BGP peer groups, enter one of the following commands in Configure mode.

Add a host to a BGP peer group.	**bgp add peer-host** <*ipaddr*> **group** <*number-or-string*>
Add a network to a BGP peer group.	**bgp add network** <*ip-addr-mask*>**|all** **group** <*number-or-string*>

You may also remove a BGP peer from a peer group. To do so, enter the following command in Configure mode:

Remove a host from a BGP peer group.	**bgp clear peer-host** <*ipaddr*>

Starting BGP

BGP is disabled by default. To start BGP, enter the following command in Configure mode.

Start BGP.	**bgp start**

Using AS-Path Regular Expressions

An AS-path regular expression is a regular expression where the alphabet is the set of AS numbers. An AS-path regular expression is composed of one or more AS-path expressions. An AS-path expression is composed of AS path terms and AS-path operators.

An AS path term is one of the following three objects:

autonomous_system
: Is any valid autonomous system number, from 1 through 65534 inclusive.

. (dot)
: Matches any autonomous system number.

(aspath_regexp)
: Parentheses group subexpressions. An operator, such as * or ? works on a single element or on a regular expression enclosed in parentheses.

An AS-path operator is one of the following:

aspath_term {m,n}
: A regular expression followed by {m,n} (where m and n are both non-negative integers and m <= n) means at least m and at most n repetitions.

aspath_term {m}
: A regular expression followed by {m} (where m is a positive integer) means exactly m repetitions.

aspath_term {m,}
: A regular expression followed by {m,} (where m is a positive integer) means m or more repetitions.

aspath_term *
: An AS path term followed by * means zero or more repetitions. This is shorthand for {0,}.

aspath_term +
: A regular expression followed by + means one or more repetitions. This is shorthand for {1,}.

aspath_term ?
: A regular expression followed by ? means zero or one repetition. This is shorthand for {0,1}.

aspath_term | aspath_term
: Matches the AS term on the left, or the AS term on the right.

For example:

(4250 .*) Means anything beginning with 4250.

(.* 6301 .*) Means anything with 6301.

(.* 4250) Means anything ending with 4250.

(. * 1104|1125|1888|1135 .*)
Means anything containing 1104 or 1125 or 1888 or 1135.

AS-path regular expressions are used as one of the parameters for determining which routes are accepted and which routes are advertised.

AS-Path Regular Expression Examples

To import MCI routes with a preference of 165:

```
ip-router policy create bgp-import-source mciRoutes aspath-regular-expression “(.* 3561 .*)” origin any
    sequence-number 10
ip-router policy import source mciRoutes network all preference 165
```

To import all routes (.* matches all AS paths) with the default preference:

```
ip-router policy create bgp-import-source allOthers aspath-regular-expression “(.*)” origin any
        sequence-number 20
ip-router policy import source allOthers network all
```

To export all active routes from 284 or 813 or 814 or 815 or 816 or 3369 or 3561 to autonomous system 64800.

```
ip-router policy create bgp-export-destination to-64800 autonomous-system 64800
ip-router policy create aspath-export-source allRoutes aspath-regular-expression
        “(.*(284|813|814|815|816|3369|3561) .*)” origin any protocol bgp
ip-router policy export destination to-64800 source allRoutes network all
```

Using the AS Path Prepend Feature

When BGP compares two advertisements of the same prefix that have differing AS paths, the default action is to prefer the path with the lowest number of transit AS hops; in other words, the preference is for the shorter AS path length. The AS path prepend feature is a way to manipulate AS path attributes to influence downstream route selection. AS path prepend involves inserting the originating AS into the beginning of the AS prior to announcing the route to the exterior neighbor.

Lengthening the AS path makes the path less desirable than would otherwise be the case. However, this method of influencing downstream path selection is feasible only when comparing prefixes of the same length because an instance of a more specific prefix always is preferable.

On the X-Pedition router, the number of instances of an AS that are put in the route advertisement is controlled by the **as-count** option of the **bgp set peer-host** command.

The following is an example:

```
#
# insert two instances of the AS when advertising the route to this peer
#
bgp set peer-host 194.178.244.33 group nlnet as-count 2
#
# insert three instances of the AS when advertising the route to this
# peer
#
bgp set peer-host 194.109.86.5 group webnet as-count 3
```

Notes on Using the AS Path Prepend Feature

- Use the **as-count** option for external peer-hosts only.
- If the **as-count** option is entered for an active BGP session, routes will ***not*** be resent to reflect the new setting. To have routes reflect the new setting, you must restart the peer session. To do this:

  a. Enter Configure mode.

  b. Negate the command that adds the peer-host to the peer-group. (If this causes the number of peer-hosts in the peer-group to drop to zero, then you must also negate the command that creates the peer group.)

  c. Exit Configure mode.

  d. Re-enter Configure mode.

  e. Add the peer-host back to the peer-group.

  If the **as-count** option is part of the startup configuration, the above steps are unnecessary.

BGP Configuration Examples

This section presents sample configurations illustrating BGP features. The following features are demonstrated:

- BGP peering
- Internal BGP (IBGP)
- External BGP (EBGP) multihop
- BGP community attribute
- BGP local preference (local_pref) attribute
- BGP Multi-Exit Discriminator (MED) attribute
- BGP multipath configuration
- EBGP aggregation
- Route reflection

BGP Peering Session Example

The router process used for a specific BGP peering session is known as a *BGP speaker*. A single router can have several BGP speakers. Successful BGP peering depends on the establishment of a neighbor relationship between BGP speakers. The first step in creating a BGP neighbor relationship is the establishment of a TCP connection (using TCP port 179) between peers.

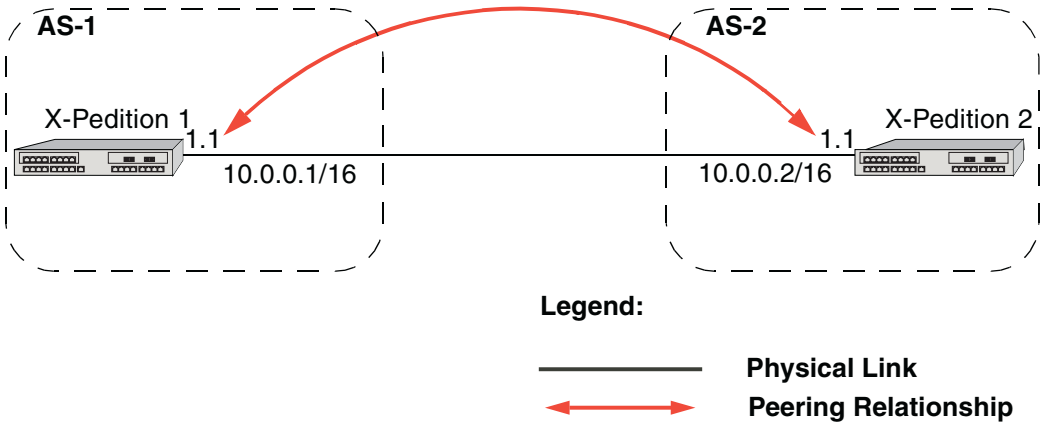
A BGP Open message can then be sent between peers across the TCP connection to establish various BGP variables (BGP Version, AS number (ASN), hold time, BGP identifier, and optional parameters). Upon successful completion of the BGP Open negotiations, BGP Update messages containing the BGP routing table can be sent between peers.

BGP does not require a periodic refresh of the entire BGP routing table between peers. Only incremental routing changes are exchanged. Therefore, each BGP speaker is required to retain the entire BGP routing table of their peer for the duration of the peer's connection.

BGP "keepalive" messages are sent between peers periodically to ensure that the peers stay connected. If one of the routers encounters a fatal error condition, a BGP notification message is sent to its BGP peer, and the TCP connection is closed.

Figure 24 illustrates a sample BGP peering session.

Figure 24. Sample BGP Peering Session



The CLI configuration for router 1 is as follows:

```
interface create ip et.1.1 address-netmask 10.0.0.1/16 port et.1.1
#
# Set the AS of the router
#
ip-router global set autonomous-system 1
#
# Set the router ID
#
ip-router global set router-id 10.0.0.1
#
# Create EBGP peer group pg1w2 for peering with AS 2
#
bgp create peer-group pg1w2 type external autonomous-system 2
#
# Add peer host 10.0.0.2 to group pg1w2
#
bgp add peer-host 10.0.0.2 group pg1w2
bgp start
```

The gated.conf file for router 1 is as follows:

```
autonomoussystem 1 ;
routerid 10.0.0.1 ;
bgp yes {
        group type external peeras 2
        {
                peer 10.0.0.2
                        ;
         };
};
```

The CLI configuration for router 2 is as follows:

```
interface create ip et.1.1 address-netmask 10.0.0.2/16 port et.1.1
ip-router global set autonomous-system 2
ip-router global set router-id 10.0.0.2
bgp create peer-group pg2w1 type external autonomous-system 1
bgp add peer-host 10.0.0.1 group pg2w1
bgp start
```

The gated.conf file for router 2 is as follows:

```
autonomoussystem 2 ;
routerid 10.0.0.2 ;
bgp yes {
        group type external peeras 1
        {
                peer 10.0.0.1
                        ;
         };
};
```

IBGP Configuration Example

Connections between BGP speakers within the same AS are referred to as internal links. A peer in the same AS is an internal peer. Internal BGP is commonly abbreviated IBGP; external BGP is EBGP.

An AS that has two or more EBGP peers is referred to as a multihomed AS. A multihomed AS can "transit" traffic between two ASs by advertising to one AS routes that it learned from the other AS. To successfully provide transit services, all EBGP speakers in the transit AS must have a consistent view of all of the routes reachable through their AS.

Multihomed transit ASs can use IBGP between EBGP-speaking routers in the AS to synchronize their routing tables. IBGP requires a full-mesh configuration; all EBGP speaking routers must have an IBGP peering session with every other EBGP speaking router in the AS.

An IGP, like OSPF, could possibly be used instead of IBGP to exchange routing information between EBGP speakers within an AS. However, injecting full Internet routes (50,000+ routes) into an IGP puts an expensive burden on the IGP routers. Additionally, IGPs cannot communicate all of the BGP attributes for a given route. It is, therefore, recommended that an IGP not be used to propagate full Internet routes between EBGP speakers. IBGP should be used instead.

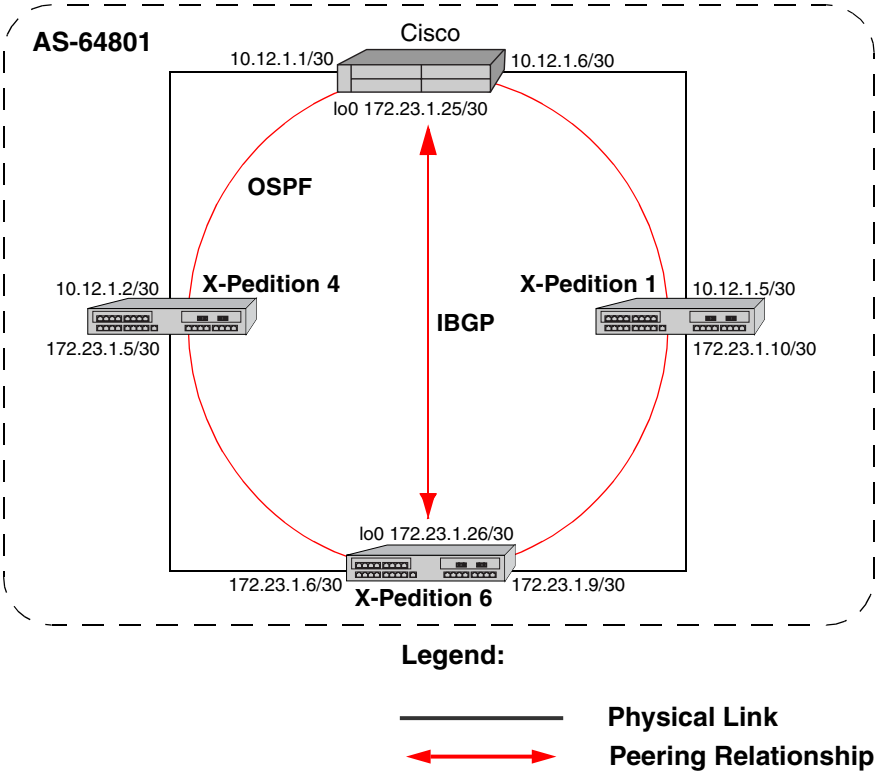
IBGP Routing Group Example

An IBGP Routing group uses the routes of an interior protocol to resolve forwarding addresses. An IBGP Routing group will determine the immediate next hops for routes by using the next hop received with a route from a peer as a forwarding address, and using this to look up an immediate next hop in an IGP's routes. Such groups support distant peers, but need to be informed of the IGP whose routes they are using to determine immediate next hops. This implementation comes closest to the IBGP implementation of other router vendors.

You should use the IBGP Routing group as the mechanism to configure the X-Pedition router for IBGP. If the peers are directly connected, then IBGP using group-type Internal can also be used. Note that for running IBGP using group-type Routing you must run an IGP such as OSPF to resolve the next hops that come with external routes. You could also use protocol **any** so that all protocols are eligible to resolve the BGP forwarding address.

Figure 25 shows a sample BGP configuration that uses the Routing group type.

Figure 25. Sample IBGP Configuration (Routing Group Type)



In this example, OSPF is configured as the IGP in the autonomous system. The following lines in the router 6 configuration file configure OSPF:

```
#
# Create a secondary address for the loopback interface
#
interface add ip lo0 address-netmask 172.23.1.26/30
ospf create area backbone
ospf add interface to-XP4 to-area backbone
ospf add interface to-XP1 to-area backbone
#
# This line is necessary because we want CISCO to peer with our loopback
# address.This will make sure that the loopback address gets announced
# into OSPF domain
#
ospf add stub-host 172.23.1.26 to-area backbone cost 1
ospf set interface to-XP4 priority 2
ospf set interface to-XP1 priority 2
ospf set interface to-XP4 cost 2
ospf start
```

The following lines in the Cisco router configure OSPF:

```
The following lines on the CISCO 4500 configures it for OSPF.
router ospf 1
 network 10.12.1.1 0.0.0.0 area 0
 network 10.12.1.6 0.0.0.0 area 0
 network 172.23.1.14 0.0.0.0 area 0
```

The following lines in router 6 set up peering with the Cisco router using the Routing group type.

```
# Create a internal routing group.
bgp create peer-group ibgp1 type routing autonomous-system 64801 proto any interface all
# Add CISCO to the above group
bgp add peer-host 172.23.1.25 group ibgp1
# Set our local address. This line is necessary because we want CISCO to
# peer with our loopback
bgp set peer-group ibgp1 local-address 172.23.1.26
# Start BGP
bgp start
```

The following lines on the Cisco router set up IBGP peering with router 6.

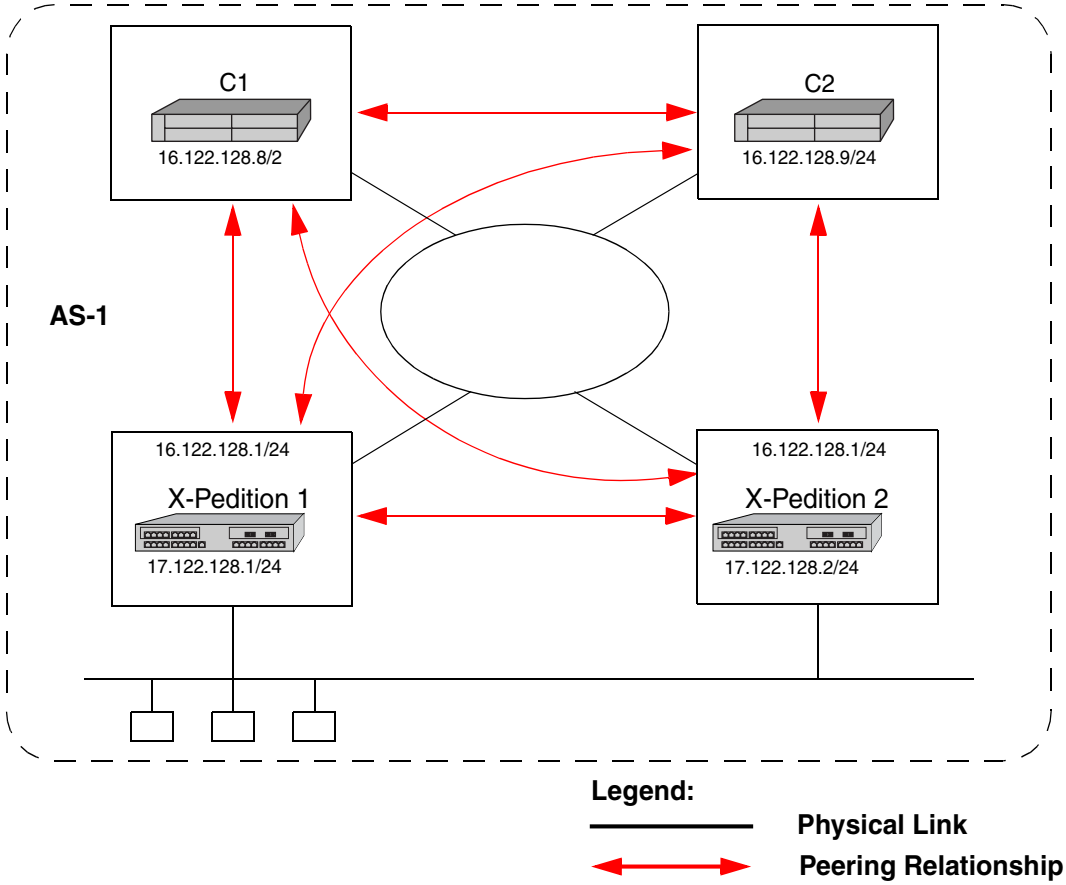
```
router bgp 64801
!
! Disable synchronization between BGP and IGP
!
 no synchronization
neighbor 172.23.1.26 remote-as 64801
!
! Allow internal BGP sessions to use any operational interface for TCP
! connections
!
 neighbor 172.23.1.26 update-source Loopback0
```

IBGP Internal Group Example

The IBGP Internal group expects all peers to be directly attached to a shared subnet so that, like external peers, the next hops received in BGP advertisements may be used directly for forwarding. All Internal group peers should be L2 adjacent.

Figure 26 illustrates a sample IBGP Internal group configuration.

Figure 26. Sample IBGP Configuration (Internal Group Type)



The CLI configuration for router 1 is as follows:

```
ip-router global set autonomous-system 1
bgp create peer-group int-ibgp-1 type internal autonomous-system 1
bgp add peer-host 16.122.128.2 group int-ibgp-1
bgp add peer-host 16.122.128.8 group int-ibgp-1
bgp add peer-host 16.122.128.9 group int-ibgp-1
```

The gated.conf file for router 1 is as follows:

```
autonomoussystem 1 ;

routerid 16.122.128.1 ;

bgp yes {
        traceoptions aspath detail packets detail open detail update ;

        group type internal peeras 1
        {
                peer 16.122.128.2
                        ;
                peer 16.122.128.8
                        ;
                peer 16.122.128.9
                        ;
        };
};
```

The CLI configuration for router 2 is as follows:

```
ip-router global set autonomous-system 1
bgp create peer-group int-ibgp-1 type internal autonomous-system 1
bgp add peer-host 16.122.128.1 group int-ibgp-1
bgp add peer-host 16.122.128.8 group int-ibgp-1
bgp add peer-host 16.122.128.9 group int-ibgp-1
```

The gated.conf file for router 2 is as follows:

```
autonomoussystem 1 ;

routerid 16.122.128.2 ;

bgp yes {
        traceoptions aspath detail packets detail open detail update ;

        group type internal peeras 1
        {
                peer 16.122.128.1
                        ;
                peer 16.122.128.8
                        ;
                peer 16.122.128.9
                        ;
        };
};
```

The configuration for router C1 (a Cisco router) is as follows:

```
router bgp 1
 no synchronization
 network 16.122.128.0 mask 255.255.255.0
 network 17.122.128.0 mask 255.255.255.0
 neighbor 16.122.128.1 remote-as 1
 neighbor 16.122.128.1 next-hop-self
 neighbor 16.122.128.1 soft-reconfiguration inbound
 neighbor 16.122.128.2 remote-as 1
 neighbor 16.122.128.2 next-hop-self
 neighbor 16.122.128.2 soft-reconfiguration inbound
 neighbor 16.122.128.9 remote-as 1
 neighbor 16.122.128.9 next-hop-self
 neighbor 16.122.128.9 soft-reconfiguration inbound
 neighbor 18.122.128.4 remote-as 4
```

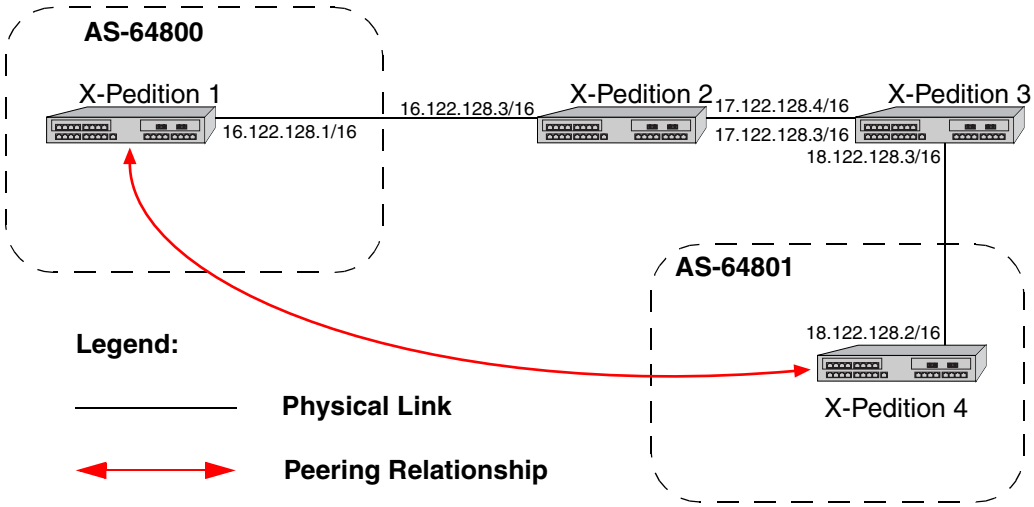
The configuration for router C2 (a Cisco router) is as follows:

```
router bgp 1
 no synchronization
 network 16.122.128.0 mask 255.255.255.0
 network 17.122.128.0 mask 255.255.255.0
 neighbor 14.122.128.5 remote-as 5
 neighbor 16.122.128.1 remote-as 1
 neighbor 16.122.128.1 next-hop-self
 neighbor 16.122.128.1 soft-reconfiguration inbound
 neighbor 16.122.128.2 remote-as 1
 neighbor 16.122.128.2 next-hop-self
 neighbor 16.122.128.2 soft-reconfiguration inbound
 neighbor 16.122.128.8 remote-as 1
 neighbor 16.122.128.8 next-hop-self
 neighbor 16.122.128.8 soft-reconfiguration inbound
```

EBGP Multihop Configuration Example

EBGP Multihop refers to a configuration where external BGP neighbors are not connected to the same subnet. Such neighbors are logically, but not physically connected. For example, BGP can be run between external neighbors across non-BGP routers. Some additional configuration is required to indicate that the external peers are not physically attached.

This sample configuration shows External BGP peers, routers 1 and 4, which are not connected to the same subnet. In this example, a hold time of 30 seconds for keepalive messages is configured for both routers 1 and 4. The sample configuration assumes that IP communication has already been established between router 1 and 4.



The CLI configuration for router 1 is as follows:

```
bgp create peer-group ebgp_multihop autonomous-system 64801 type external
bgp add peer-host 18.122.128.2 group ebgp_multihop
!
! Specify the gateway option, which indicates EBGP multihop. Set the
! gateway option to the address of the router that has a route to the
! peer.
!
bgp set peer-host 18.122.128.2 gateway 16.122.128.3 group ebgp_multihop hold-time 30
```

The gated.conf file for router 1 is as follows:

```
autonomoussystem 64800 ;

routerid 0.0.0.1 ;

bgp yes {
        traceoptions state ;

        group type external peeras 64801
        {
                peer 18.122.128.2
                        gateway 16.122.128.3
                                holdtime 30
;
        };
};

static {
        18.122.0.0 masklen 16
                gateway 16.122.128.3
                ;
};
```

The CLI configuration for router 2 is as follows:

```
interface create ip to-R1 address-netmask 16.122.128.3/16 port et.1.1
interface create ip to-R3 address-netmask 17.122.128.3/16 port et.1.2
#
# Static route needed to reach 18.122.0.0/16
#
ip add route 18.122.0.0/16 gateway 17.122.128.4
```

The gated.conf file for router 2 is as follows:

```
static {
        18.122.0.0 masklen 16
                gateway 17.122.128.4
                ;
};
```

The CLI configuration for router 3 is as follows:

```
interface create ip to-R2 address-netmask 17.122.128.4/16 port et.4.2
interface create ip to-R4 address-netmask 18.122.128.4/16 port et.4.4
ip add route 16.122.0.0/16 gateway 17.122.128.3
```

The gated.conf file for router 3 is as follows:

```
static {
        16.122.0.0 masklen 16
                gateway 17.122.128.3
                ;
};
```

The CLI configuration for router 4 is as follows:

```
bgp create peer-group ebgp_multihop autonomous-system 64801 type external
bgp add peer-host 18.122.128.2 group ebgp_multihop
!
! Specify the gateway option, which indicates EBGP multihop. Set the
! gateway option to the address of the router that has a route to the
! peer.
!
bgp set peer-host 18.122.128.2 gateway 16.122.128.3 group ebgp_multihop hold-time 30
```

The gated.conf file for router 4 is as follows:

```
autonomoussystem 64800 ;

routerid 0.0.0.1 ;

bgp yes {
        traceoptions state ;

        group type external peeras 64801
        {
                peer 18.122.128.2
                        gateway 16.122.128.3
                                holdtime 30
```

Community Attribute Example

The following configuration illustrates the BGP community attribute. Community is specified as one of the parameters in the **ip-router policy create optional-attributes-list** command.

Figure 27 shows a BGP configuration where the specific community attribute is used. Figure 28 shows a BGP configuration where the well-known community attribute is used.

Figure 27. Sample BGP Configuration (Specific Community)

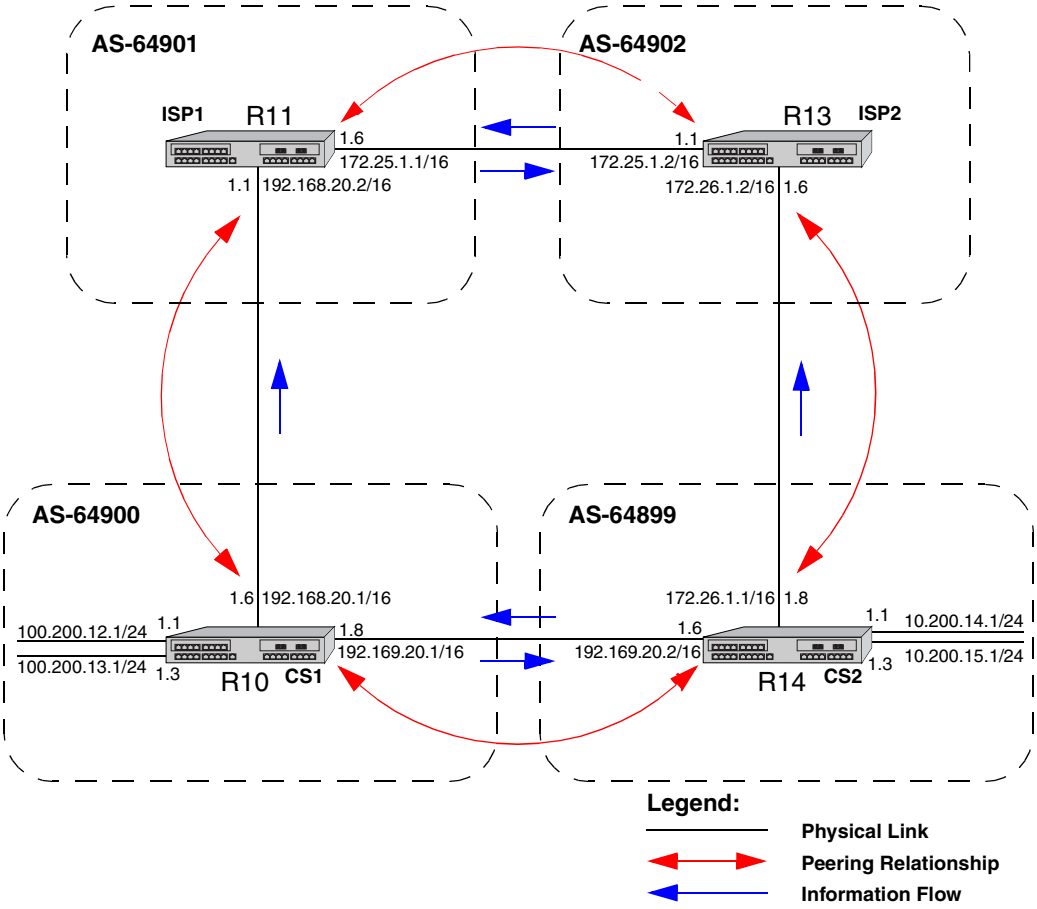
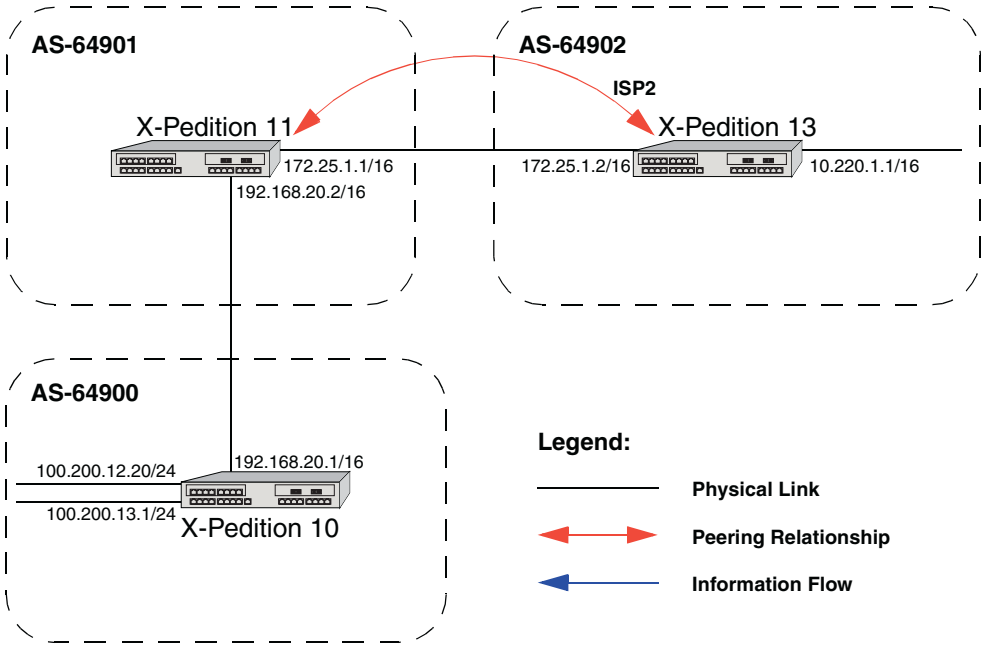


Figure 28. Sample BGP Configuration (Well-Known Community)



The Community attribute can be used in three ways:

1. In a BGP Group statement: Any packets sent to this group of BGP peers will have the communities attribute in the BGP packet modified to be this communities attribute value from this AS.

2. In an Import Statement: Any packets received from a BGP peer will be checked for the community attribute. The **ip-router policy create optional-attributes-list** command allows the specification of an import policy based on optional path attributes (for instance, the community attribute) found in the BGP update. If multiple communities are specified in the **optional-attributes-list**, only updates carrying all of the specified communities will be matched. If **none** is specified, only updates lacking the community attribute will be matched.

   Note that it is quite possible for several BGP import clauses to match a given update. If more than one clause matches, the first matching clause will be used; all later matching clauses will be ignored. For this reason, it is generally desirable to order import clauses from most to least specific. An import clause without an **optional-attributes-list** option will match any update with any (or no) communities.

In Figure 28, router 11 has the following configuration:

```
#
# Create an optional attribute list with identifier color1 for a community
# attribute (community-id 160 AS 64901)
#
ip-router policy create optional-attributes-list color1 64901:160
#
# Create an optional attribute list with identifier color2 for a community
# attribute (community-id 155 AS 64901)
#
ip-router policy create optional-attributes-list color2 64901:155
#
# Create a BGP import source for importing routes from AS 64900 containing the
# community attribute (community-id 160 AS 64901). This import source is given an
# identifier 901color1 and sequence-number 1.
#
ip-router policy create bgp-import-source 901color1 optional-attributes-list color1 autonomous-system 64900
        sequence-number 1
ip-router policy create bgp-import-source 901color2 optional-attributes-list color2 autonomous-system 64900
        sequence-number 2
ip-router policy create bgp-import-source 901color3 optional-attributes-list color1 autonomous-system 64902
        sequence-number 3
ip-router policy create bgp-import-source 901color4 optional-attributes-list color2 autonomous-system 64902
        sequence-number 4
#
# Import all routes matching BGP import source 901color1 (from AS 64900 having
# community attribute with ID 160 AS 64901) with a preference of 160
#
ip-router policy import source 901color1 network all preference 160
ip-router policy import source 901color2 network all preference 155
ip-router policy import source 901color3 network all preference 160
ip-router policy import source 901color4 network all preference 155
```

In Figure 28, router 13 has the following configuration:

```
ip-router policy create optional-attributes-list color1 64902:160
ip-router policy create optional-attributes-list color2 64902:155
ip-router policy create bgp-import-source 902color1 optional-attributes-list color1 autonomous-system 64899
        sequence-number 1
ip-router policy create bgp-import-source 902color2 optional-attributes-list color2 autonomous-system 64899
        sequence-number 2
ip-router policy create bgp-import-source 902color3 optional-attributes-list color1 autonomous-system 64901
        sequence-number 3
ip-router policy create bgp-import-source 902color4 optional-attributes-list color2 autonomous-system 64901
        sequence-number 4
ip-router policy import source 902color1 network all preference 160
ip-router policy import source 902color2 network all preference 155
ip-router policy import source 902color3 network all preference 160
ip-router policy import source 902color4 network all preference 155
```

3. In an Export Statement: The **add-optional-attributes-list** option of the **ip-router policy create bgp-export-destination** command may be used to send the BGP community attribute. Any communities specified with the **add-optional-attributes-list** option are sent in addition to any received in the route or specified with the group.

   In Figure 28, router 10 has the following configuration:

```
#
# Create an optional attribute list with identifier color1 for a community
# attribute (community-id 160 AS 64902)
#
ip-router policy create optional-attributes-list color1 64902:160
#
# Create an optional attribute list with identifier color2 for a community
# attribute (community-id 155 AS 64902)
#
ip-router policy create optional-attributes-list color2 64902:155
#
# Create a direct export source
#
ip-router policy create direct-export-source 900toanydir metric 10
#
# Create BGP export-destination for exporting routes to AS 64899 containing the
# community attribute (community-id 160 AS 64902). This export-destination has an
# identifier 900to899dest
#
ip-router policy create bgp-export-destination 900to899dest autonomous-system 64899 add-optional-attributes-list color1
ip-router policy create bgp-export-destination 900to901dest autonomous-system 64901 add-optional-attributes-list color2
#
# Export routes to AS 64899 with the community attribute (community-id 160 AS
# 64902)
#
ip-router policy export destination 900to899dest source 900toanydir network all
ip-router policy export destination 900to901dest source 900toanydir network all
```

   In Figure 28, router 14 has the following configuration:

```
ip-router policy create bgp-export-destination 899to900dest autonomous-system 64900 add-optional-attributes-list color1
ip-router policy create bgp-export-destination 899to902dest autonomous-system 64902 add-optional-attributes-list color2
ip-router policy create bgp-export-source 900toany autonomous-system 64900 metric 10
ip-router policy create optional-attributes-list color1 64901:160
ip-router policy create optional-attributes-list color2 64901:155
ip-router policy export destination 899to900dest source 899toanydir network all
ip-router policy export destination 899to902dest source 899toanydir network all
```

   Any communities specified with the **add-optional-attributes-list** option are sent in addition to any received with the route or associated with a BGP export destination.

   The community attribute may be a single community or a set of communities. A maximum of 10 communities may be specified.

The community attribute can take any of the following forms:

- Specific community

  The specific community consists of the combination of the AS-value and community ID.

- no-export

  no-export is a special community which indicates that the routes associated with this attribute must not be advertised outside a BGP confederation boundary. Since the X-Pedition router's implementation does not support Confederations, this boundary is an AS boundary.

  For example, router 10 in Figure 28 has the following configuration:

```
ip-router policy create optional-attributes-list noexport no-export
ip-router policy create bgp-export-destination 900to901dest autonomous-system 64901
      add-optional-attributes-list noexport
ip-router policy export destination 900to901dest source 900to901src network all
ip-router policy export destination 900to901dest source 900to901dir network all
```

- no-advertise

  no-advertise is a special community indicating that the routes associated with this attribute must not be advertised to other BGP peers. A packet can be modified to contain this attribute and passed to its neighbor. However, if a packet is received with this attribute, it cannot be transmitted to another BGP peer.

- no-export-subconfed

  no-export-subconfed is a special community indicating the routes associated with this attribute must not be advertised to external BGP peers. (This includes peers in other members' autonomous systems inside a BGP confederation.)

  A packet can be modified to contain this attribute and passed to its neighbor. However, if a packet is received with this attribute, the routes (prefix-attribute pair) cannot be advertised to an external BGP peer.

- none

  This is not actually a community, but rather a keyword that specifies that a received BGP update is only to be matched if no communities are present. It has no effect when originating communities.

Notes on Using Communities

When originating BGP communities, the set of communities that is actually sent is the union of the communities received with the route (if any), those specified in group policy (if any), and those specified in export policy (if any).

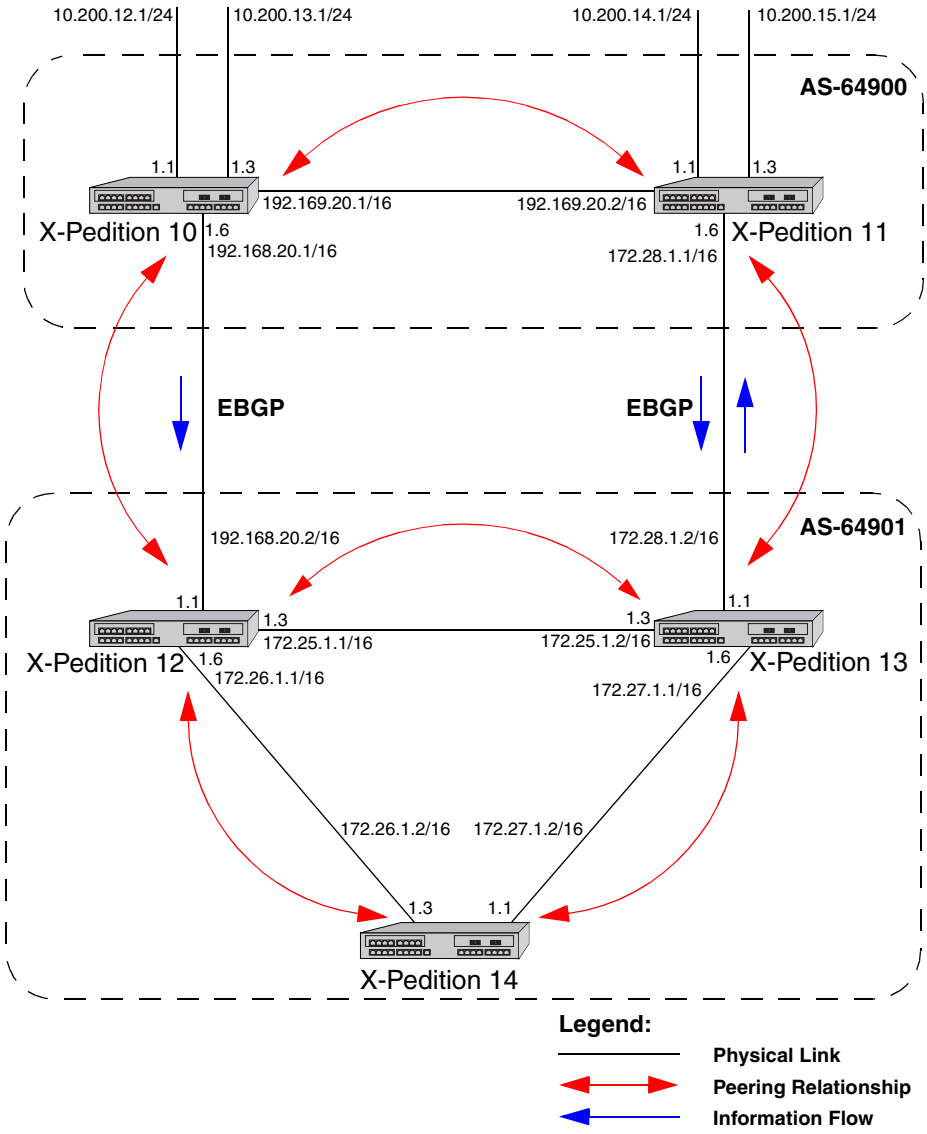
When receiving BGP communities, the update is only matched if all communities specified in the **ip-router policy create optional-attributes-list** command are present in the BGP update. (If additional communities are also present in the update, it will still be matched.)

Local Preference Example

The **bgp set internal-set-pref** command allows GateD to set the local preference to reflect GateD's own internal preference for the route, as given by the global protocol preference value. Note that in this case, local preference is a function of the GateD preference and set-pref options.

Figure 29 shows a BGP configuration that uses the BGP local preference attribute in a sample BGP configuration with two autonomous systems. All traffic exits Autonomous System 64901 through the link between router 13 and router 11. This is accomplished by configuring a higher local preference on router 13 than on router 12. Because local preference is exchanged between the routers within the AS, all traffic from AS 64901 is sent to router 13 as the exit point.

Figure 29. Sample BGP Configuration (Local Preference)



The formula used to compute the local preference is as follows:

Local_Pref = 254 – (global protocol preference for this route) + internal-set-pref metric

Note: A value greater than 254 will be reset to 254. GateD will only send Local_Pref values between 0 and 254.

In a mixed GateD and non-GateD network, the non-GateD IBGP implementation may send Local_Pref values that are greater than 254. When operating a mixed network of this type, you should make sure that all routers are restricted to sending Local_Pref values in the range metric to 254.

In router 12's CLI configuration file, the import preference is set to 160:

```
#
# Set the set-pref metric for the IBGP peer group
#
bgp set peer-group as901
bgp set internal-set-pref 100
ip-router policy create bgp-import-source as900 autonomous-system 64900 preference 160
```

Using the formula for local preference [Local_Pref = 254 - (global protocol preference for this route) + metric], the Local_Pref value put out by router 12 is 254 - 160+100 = 194.

For router 13, the import preference is set to 150. The Local_Pref value put out by router 13 is 254 - 150+100 = 204.

```
ip-router policy create bgp-import-source as900 autonomous-system 64900 preference 150
```

Note the following when using the **bgp set internal-set-pref** command:

- All routers in the same network that are running GateD and participating in IBGP should use the **bgp set internal-set-pref** command, and the metric MUST be set to the same value. For example, in Figure 29, routers 12, 13, and 14 use the following in their CLI configuration files:

  ```
  bgp set peer-group as901
  bgp set internal-set-pref 100
  ```

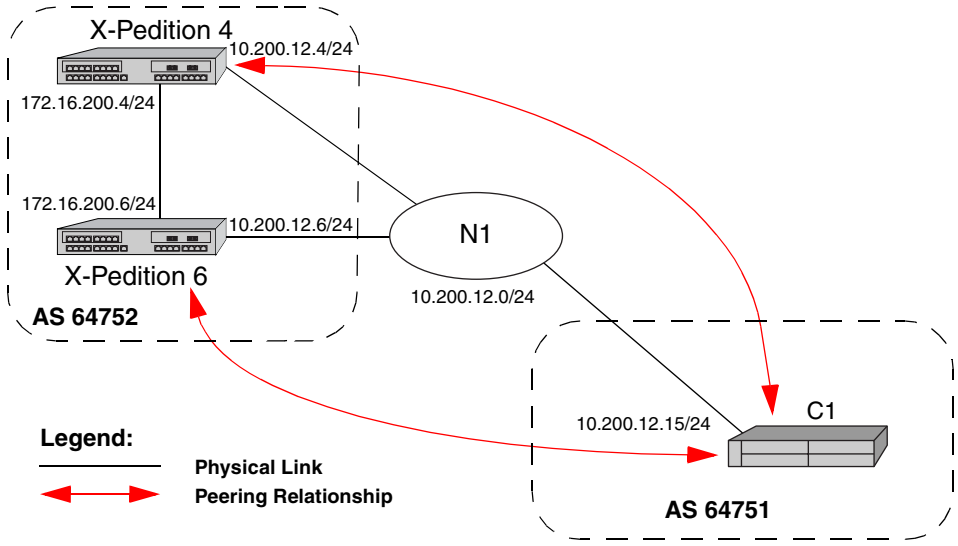
- The value of the **internal-set-pref** metric should be consistent with the import policy in the network. The metric value should be set high enough to avoid conflicts between BGP routes and IGP or static routes. For example, if the import policy sets GateD preferences ranging from 170 to 200, a metric of 170 would make sense. Users should set the metric high enough to avoid conflicts between BGP routes and IGP or static routes.

Multi-Exit Discriminator Attribute Example

Multi-Exit Discriminator (MED) is a BGP attribute that affects the route selection process. MED is used on external links to discriminate among multiple exit or entry points to the same neighboring AS. All other factors being equal, the exit or entry point with a lower metric should be preferred. If received over external links, the MED attribute may be propagated over internal links to other BGP speakers within the same AS. The MED attribute is never propagated to other BGP speakers in neighboring autonomous systems.

Figure 30 shows a sample BGP configuration where the MED attribute has been used.

Figure 30. Sample BGP Configuration (MED Attribute)



Routers 4 and 6 inform router C1 about network 172.16.200.0/24 through External BGP (EBGP). Router 6 announced the route with a MED of 10, whereas router 4 announces the route with a MED of 20. Of the two EBGP routes, router C1 chooses the one with a smaller MED. Thus router C1 prefers the route from router 6, which has a MED of 10. Router 4 has the following CLI configuration:

```
bgp create peer-group pg752to751 type external autonomous-system 64751
bgp add peer-host 10.200.12.15 group pg752to751
#
# Set the MED to be announced to peer group pg752to751
#
bgp set peer-group pg752to751 metric-out 20
bgp set peer-group pg752to751 med
```

Router 6 has the following CLI configuration:

```
bgp create peer-group pg752to751 type external autonomous-system 64751
bgp add peer-host 10.200.12.15 group pg752to751
bgp set peer-group pg752to751 metric-out 10
bgp set peer-group pg752to751 med
```

Note: Before the router can process and select the correct route based on the MED values received from other BGP peers, users must set the selection process in the active configuration of the router where the peer is defined. To set the selection process, enter one (or both) of the following commands in the configuration, depending on the type of BGP peer configured (i.e., peer group, peer host, or both):

bgp set peer-group <*group_name*> **med**
bgp set peer-host <*IP Address*> **med**

BGP Multipath Configuration

The BGP Multipath feature is used with external BGP4 peers to support up to four physical interfaces (Ethernet or serial lines), preferably with the same throughput as each other. Should an interface go down, the remaining links will assume the load of the downed interface by instantiating new flows across all links—based on the next selected gateway for a particular route. The gateway selected for each route rotates in circular fashion between all available gateways (up to a maximum of four). The nexthop selection for each route is determined randomly and subsequent route selections progress to the next highest gateway for each use of the route. This creates a load share across the available links.

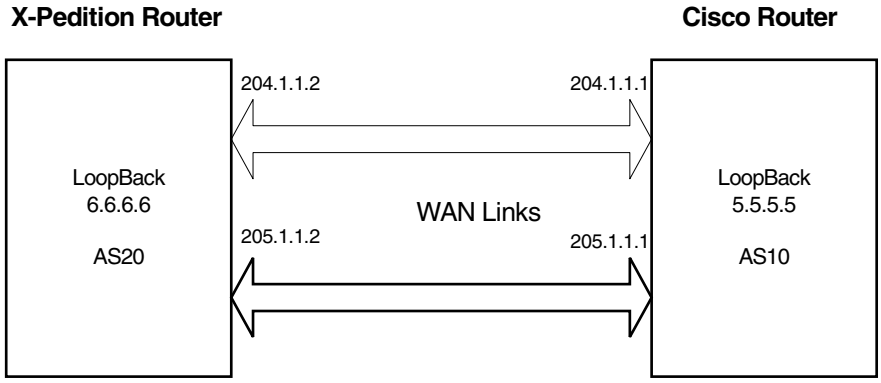
To configure BGP4 Multipath, the firmware requires that the BGP address and its peer are based on a loopback address. Because users may not delete a loopback address once it is created, the router maintains the BGP peer association as long as one or more nexthop gateways exists for the peer. To set up the configuration, each physical link must be included as a static route to the peer's loopback address. In the following example, 204.1.1.1 and 205.1.1.1 are peer gateways:

```
ip add route <peer-loopback-address>/32 gate-list "204.1.1.1  205.1.1.1" retain
```

Note: BGP Multipath is not supported by BGP v2 and v3.

Sample Configuration 1

The following example depicts a BGP multipath configuration with a Cisco router:



X-Pedition Configuration

```
port set se.3.1 wan-encapsulation ppp speed 64000
port set se.3.3 wan-encapsulation ppp speed 64000
interface create ip ser1 address-netmask 204.1.1.2 port se.3.1
interface create ip ser2 address-netmask 205.1.1.2 port se.3.3
interface add ip lo0 address-netmask 6.6.6.6/32
interface add ip en0 address-netmask 10.136.181.81/16

ip-router global set router-id 6.6.6.6
ip-router global set autonomous-system 20

ip add route 5.5.5.5/32 gate-list “204.1.1.1 205.1.1.1” retain

bgp create peer-group EBGP1 type external autonomous-system 10

bgp add peer-host 5.5.5.5 group EBGP1
bgp set peer-host 5.5.5.5 group EBGP1 gateway 6.6.6.6
bgp set peer-host 5.5.5.5 group EBGP1 local-address 6.6.6.6
bgp start

system set name AS20
```

Cisco Router Configuration

```
interface Loopback0
 ip address 5.5.5.5 255.255.255.255
!
interface Ethernet0
 ip address 204.1.1.1 255.255.255.0
 media-type 10BaseT
!
interface Ethernet1
 ip address 205.1.1.1 255.255.255.0
 media-type 10BaseT
!
interface Ethernet2
 ip address 10.10.10.3 255.255.255.0
 media-type 10BaseT
!
interface Ethernet3
 no ip address
 shutdown
 media-type 10BaseT
!
interface Ethernet4
 no ip address
 shutdown
 media-type 10BaseT
!
interface Ethernet5
 no ip address
 shutdown
 media-type 10BaseT
!
router bgp 10
 redistribute connected metric 20
 redistribute static
 neighbor 6.6.6.6 remote-as 20
 neighbor 6.6.6.6 ebgp-multihop 255
 neighbor 6.6.6.6 update-source Loopback0
 maximum-paths 2
!
ip classless
ip route 6.6.6.6 255.255.255.255 204.1.1.2
ip route 6.6.6.6 255.255.255.255 205.1.1.2
```

Additionally, the following BGP statements are required to enable this feature:

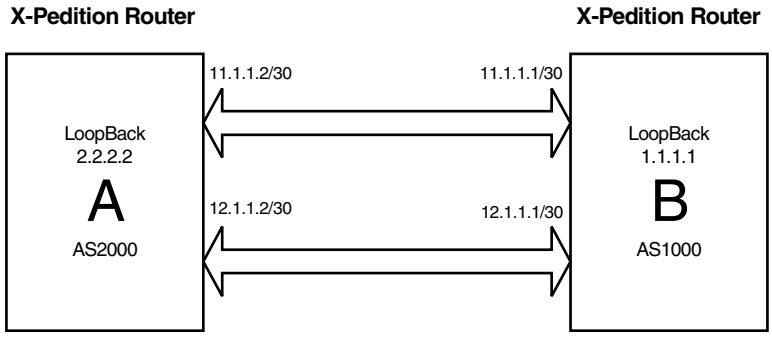
bgp set peer-host <*peer-loopback address*> **group** <*group-name*> **gateway** <*local loopback address*>
bgp set peer-host <*peer-loopback address*> **group** <*group-name*>
local-address <*local loopback address*>

The number of gateways provided in the static route configured to the peer-loopback address becomes the number of gateways the BGP4 Multipath uses to form the active links to its peer.

To display routes and active gateways in a multipath environment, use the **ip show routes** command. When viewing the output, you will note a number of active gateways to the peer's loopback address—these should be owned by BGP. If the router knows a route to a destination that is preferred over BGP, the BGP route will not be the "active" route.

Sample Configuration 2

The following example shows a BGP multipath configuration between two X-Pedition routers:



Router A Configuration

```
xp# system show version
Software Information
  Software Version   : E9.1.3.0A
  Copyright          : Copyright (c) 2003 Enterasys Networks
  Image Information  : Version E9.1.3.0A, built on Wed Apr 23 00:21:24 2003
  Image Boot Location: slot0:boot/xp9130A/
  Boot Prom Version  : prom-E3.2.0.0

  2 : interface create ip 11net address-netmask 11.1.1.2/30 port et.1.1
  3 : interface create ip 12net address-netmask 12.1.1.2/30 port et.1.2
  4 : interface add ip lo0 address-netmask 2.2.2.2/32
    !
  6 : ip-router global set router-id 2.2.2.2
  7 : ip-router global set autonomous-system 2000
    !
  9 : ip add route 1.1.1.1/32 gate-list "11.1.1.1 12.1.1.1"
    !
 11 : bgp create peer-group ebgp type external autonomous-system 1000
 13 : bgp add peer-host 1.1.1.1 group ebgp
 14 : bgp set peer-host 1.1.1.1 group ebgp local-address 2.2.2.2
 15 : bgp set peer-host 1.1.1.1 group ebgp gateway 2.2.2.2
 16 : bgp start
```

Router B Configuration

```
xp# system show version
Software Information
  Software Version   : E9.1.3.0A
  Copyright          : Copyright (c) 2003 Enterasys Networks
  Image Information  : Version E9.1.3.0A, built on Wed Apr 23 00:21:24 2003
  Image Boot Location: slot0:boot/xp9130A/
  Boot Prom Version  : prom-E3.2.0.0

  1 : interface create ip 10net address-netmask 10.1.1.2/30 port et.2.8
  2 : interface create ip 11net address-netmask 11.1.1.1/30 port et.1.1
  3 : interface create ip 12net address-netmask 12.1.1.1/30 port et.1.2
  4 : interface add ip lo0 address-netmask 1.1.1.1/32
  6 : ip-router global set router-id 1.1.1.1
  7 : ip-router global set autonomous-system 1000
    !
  9 : ip add route 2.2.2.2/32 gate-list "11.1.1.2 12.1.1.2"
    !
 11 : bgp create peer-group ebgp type external autonomous-system 2000
 13 : bgp add peer-host 2.2.2.2 group ebgp
 14 : bgp set peer-host 2.2.2.2 group ebgp local-address 1.1.1.1
 15 : bgp set peer-host 2.2.2.2 group ebgp gateway 1.1.1.1
 16 : bgp start
```

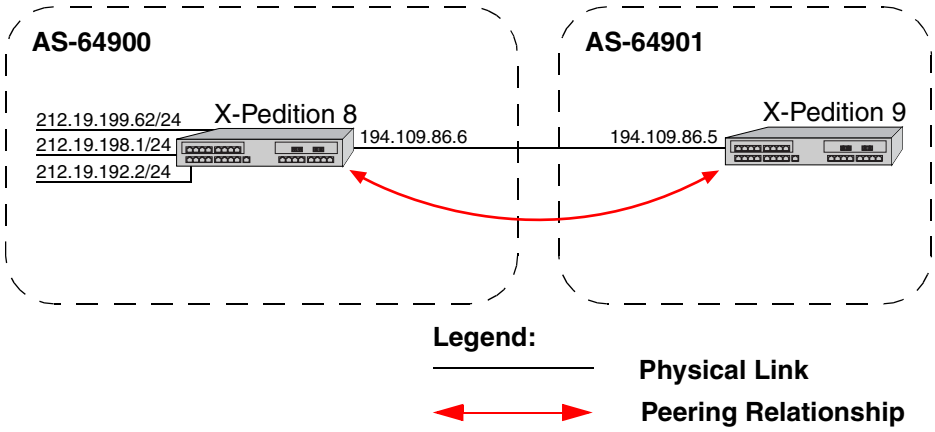
Router A's Router Table

```
xp# bgp show summary
Neighbor        V          AS       MsgRcvd  MsgSent  Up/Down      State
--------        -          -        -------  -------  -------      -----
1.1.1.1         4        1000         26        3      0d0h24m38    Established
BGP summary, 2 groups, 2 peers

xp# ip show routes

Destination  Gateway       Owner    Netif
-----------  -------       -----    -----
1.1.1.1      11.1.1.1      Static   11net
12.1.1.1                   Static   12net
2.2.2.2      2.2.2.2         -        lo0
10.1.1.0/30  11.1.1.1       BGP      11net
12.1.1.1                    BGP      12net
```

EBGP Aggregation Example

Figure 31 shows a simple EBGP configuration in which one peer is exporting an aggregated route to its upstream peer and restricting the advertisement of contributing routes to the same peer. The aggregated route is 212.19.192.0/19.

Figure 31. Sample BGP Configuration (Route Aggregation)



Router 8 has the following CLI configuration:

```
interface add ip xleapnl address-netmask 212.19.192.2/24
interface create ip hobbygate address-netmask 212.19.199.62/24 port et.1.2
interface create ip xenosite address-netmask 212.19.198.1/24 port et.1.7
interface add ip lo0 address-netmask 212.19.192.1/30
bgp create peer-group webnet type external autonomous system 64901
bgp add peer-host 194.109.86.5 group webnet
#
# Create an aggregate route for 212.19.192.0/19 with all its subnets as
# contributing routes
#
ip-router policy summarize route 212.19.192.0/19
ip-router policy redistribute from-proto aggregate to-proto bgp target-as 64901 network 212.19.192.0/19
ip-router policy redistribute from-proto direct to-proto bgp target-as 64901 network all restrict
```

Router 9 has the following CLI configuration:

```
bgp create peer-group rtr8 type external autonomous system 64900
bgp add peer-host 194.109.86.6 group rtr8
```

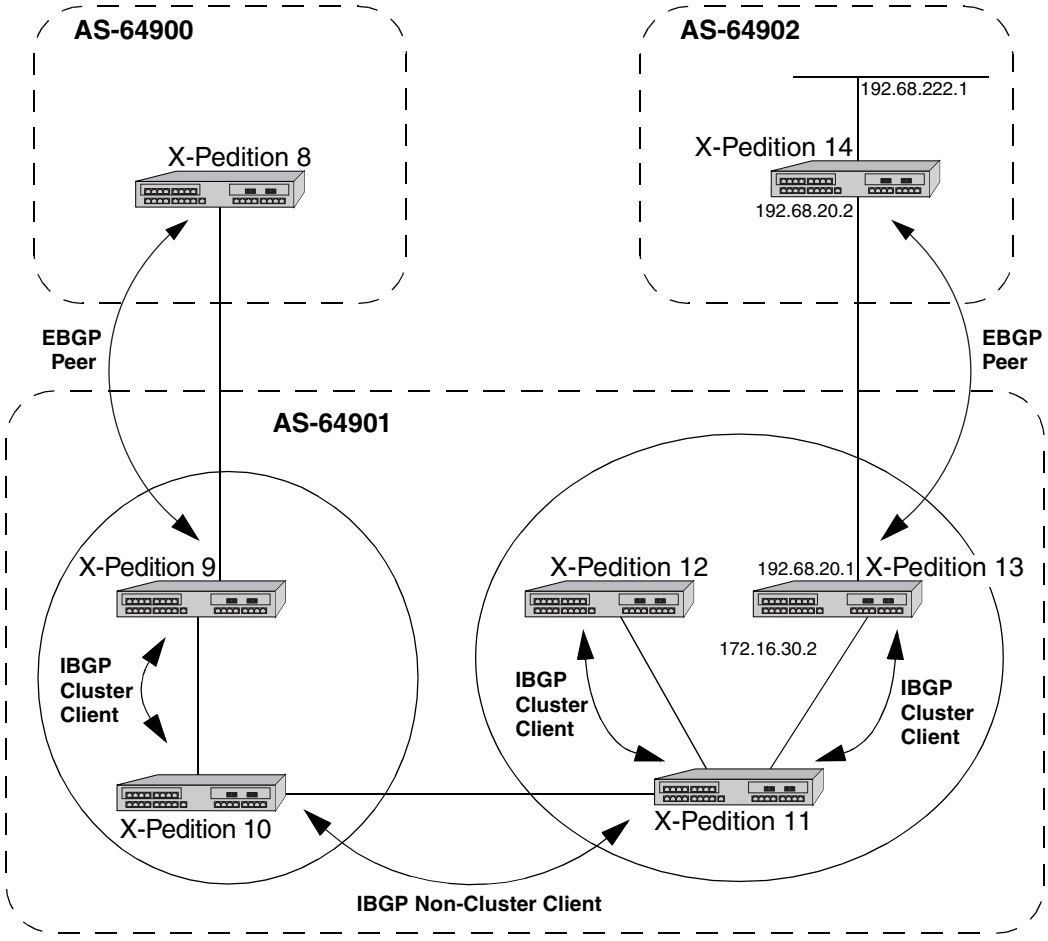
Route Reflection Example

In some ISP networks, the internal BGP mesh becomes quite large, and the IBGP full mesh does not scale well. For such situations, route reflection provides a way to alleviate the need for a full IBGP mesh. In route reflection, the clients peer with the route reflector and exchange routing information with it. In turn, the route reflector passes on (reflects) information between clients.

The IBGP peers of the route reflector fall under two categories: clients and non-clients. A route reflector and its clients form a cluster. All peers of the route reflector that are not part of the cluster are non-clients. The X-Pedition router supports client peers as well as non-client peers of a route reflector.

Figure 32 shows a sample configuration that uses route reflection.

Figure 32. Sample BGP Configuration (Route Reflection)



In this example, there are two clusters. Router 10 is the route reflector for the first cluster and router 11 is the route reflector for the second cluster. Router 10 has router 9 as a client peer and router 11 as a non-client peer. The following line in router 10's configuration file causes it to be a route reflector:

```
bgp set peer-group XP9 reflector-client
```

Router 11 has router 12 and router 13 as client peers and router 10 as non-client peer. The following line in router 11's configuration file specifies it to be a route reflector:

```
bgp set peer-group XP11 reflector-client
```

Even though the IBGP Peers are not fully meshed in AS 64901, the direct routes of router 14, that is, 192.68.222.0/24 in AS 64902 (which are redistributed in BGP) do show up in the route table of router 8 in AS64900, as shown below:

```
**********************************************
* Route Table (FIB) of Router 8          *
**********************************************
rtr-8# ip show routes

Destination          Gateway              Owner      Netif
-----------          -------              -----      -----
10.50.0.0/16         directly connected   -          en
127.0.0.0/8          127.0.0.1            Static     lo
127.0.0.1            127.0.0.1            -          lo
172.16.20.0/24       directly connected   -          mls1
172.16.70.0/24       172.16.20.2          BGP        mls1
172.16.220.0/24      172.16.20.2          BGP        mls1
192.68.11.0/24       directly connected   -          mls0
192.68.20.0/24       172.16.20.2          BGP        mls1
192.68.222.0/24      172.16.20.2          BGP        mls1
```

The direct routes of router 8, i.e., 192.68.11.0/24 in AS64900 (which are redistributed in BGP), do show up in the route table of router 14 in AS64902, as shown below:

```
************************************************************
*  Route Table (FIB) of Router 14              *
************************************************************
rtr-14# ip show routes

Destination          Gateway              Owner      Netif
-----------          -------              -----      -----
10.50.0.0/16         directly connected   -          en0
127.0.0.0/8          127.0.0.1            Static     lo0
127.0.0.1            127.0.0.1            -          lo0
172.16.20.0/24       192.68.20.1          BGP        mls1
172.16.30.0/24       192.68.20.1          BGP        mls1
172.16.90.0/24       192.68.20.1          BGP        mls1
192.68.11.0/24       192.68.20.1          BGP        mls1
192.68.20.0/24       directly connected   -          mls1
192.68.222.0/24      directly connected   -          mls0
```

Notes on Using Route Reflection

- Two types of route reflection are supported:
  - By default, all routes received by the route reflector from a client are sent to all internal peers (including the client's group, but not the client itself).
  - If the **no-client-reflec**t option is enabled, routes received from a route reflection client are sent only to internal peers that are not members of the client's group. In this case, the client's group must itself be fully meshed.

  In either case, all routes received from a non-client internal peer are sent to all route reflection clients.
- Typically, a single router acts as the reflector for a cluster of clients. However, for redundancy, two or more may also be configured to be reflectors for the same cluster. In this case, a cluster ID should be selected to identify all reflectors serving the cluster, using the **bgp set cluster-id** command. Gratuitous use of multiple redundant reflectors is not advised, since it can lead to an increase in the memory required to store routes on the redundant reflectors' peers.
- No special configuration is required on the route reflection clients. From a client's perspective, a route reflector is simply a normal IBGP peer. Any BGP version 4 speaker can be a reflector client.
- It is necessary to export routes from the local AS into the local AS when acting as a route reflector.

  To accomplish this, routers 10 and 11 have the following line in their configuration files:

```
ip-router policy redistribute from-proto bgp source-as 64901 to-proto bgp target-as 64901
```

- If the cluster ID is changed, all BGP sessions with reflector clients will be dropped and restarted.

Chapter 19

Routing Policy Configuration Guide

Route Import and Export Policy Overview

The X-Pedition family of routers supports extremely flexible routing policies. The X-Pedition router allows the network administrator to control import and export of routing information based on criteria including:

- Individual protocol
- Source and destination autonomous system
- Source and destination interface
- Previous hop router
- Autonomous system path
- Tag associated with routes
- Specific destination address

The network administrator can specify a preference level for each combination of routing information being imported by using a flexible masking capability.

The X-Pedition router also provides the ability to create advanced and simple routing policies. Simple routing policies provide a quick route redistribution between various routing protocols (RIP RIPng, and OSPF). Advanced routing policies provide more control over route redistribution.

Preference

Preference is the value the X-Pedition routing process uses to order preference of routes from one protocol or peer over another. Preference can be set using several different configuration commands. Preference can be set based on one network interface over another, from one protocol over another, or from one remote gateway over another. Preference may not be used to control the selection of routes within an Interior Gateway Protocol (IGP). This is accomplished automatically by the protocol based on metric.

Preference may be used to select routes from the same Exterior Gateway Protocol (EGP) learned from different peers or autonomous systems. Each route has only one preference value associated with it, even though the preference can be set at many places using configuration commands. The last or most specific preference value set for a route is the value used. A preference value is an arbitrarily assigned value used to determine the order of routes to the same destination in a single routing database. The active route is chosen by the lowest preference value.

A default preference is assigned to each source from which the X-Pedition routing process receives routes. Preference values range from 0 to 255 with the lowest number indicating the most preferred route.

The following table summarizes the default preference values for routes learned in various ways. The table lists the CLI commands that set preference, and shows the types of routes to which each CLI command applies. A default preference for each type of route is listed, and the table notes preference precedence between protocols. The narrower the scope of the statement, the higher precedence its preference value is given, but the smaller the set of routes it affects.

Note: The current IPv6 implementation supports only RIPng and static routes.

Table 11. Default Preference Values

Preference	Defined by CLI Command	Default
Direct connected networks	**ip-router global set interface**	0
OSPF routes	**ospf**	10
Static routes from config	**ip add route** **ipv6 add route**	60
DVMRP		70
RIP routes	**rip set preference**	100
RIPng routes	**ripng set preference**	100
Point-to-point interface		110
Routes to interfaces that are down	**ip-router global set interface down-preference**	255
Aggregate/generate routes	**ip-router policy aggr-gen**	130
OSPF AS external routes	**ospf set ase-defaults preference**	150
OSPF NSSA routes	**ospf set nssa-defaults preference**	150
BGP routes	**bgp set preference**	170

Import Policies

Import policies control the importation of routes from routing protocols and their installation in the routing databases (Routing Information Base and Forwarding Information Base). Import Policies determine which routes received from other systems are used by the X-Pedition routing process. Every import policy can have up to two components:

- Import-Source
- Route-Filter

Note: The X-Pedition router maintains separate Routing and Forwarding Information Bases (RIB and FIB) for IPv4 and IPv6.

Import-Source

This component specifies the source of the imported routes. It can also specify the preference to be associated with the routes imported from this source.

The routes to be imported can be identified by their associated attributes:

- Type of the source protocol (RIP, OSPF, BGP).
- Source interface or gateway from which the route was received.
- Source autonomous system from which the route was learned.
- AS path associated with a route. Besides autonomous system, BGP also supports importation of routes using AS path regular expressions and AS path options.
- If multiple communities are specified using the optional-attributes-list, only updates carrying all of the specified communities will be matched. If the specified optional-attributes-list has the value **none** for the **community** option, then only updates lacking the community attribute will be matched.

In some cases, a combination of the associated attributes can be specified to identify the routes to be imported.

Note: It is quite possible for several BGP import policies to match a given update. If more than one policy matches, the first matching policy will be used. All later matching policies will be ignored. For this reason, it is generally desirable to order import policies from most to least specific. An import policy with an optional-attributes-list will match any update with any (or no) communities.

The importation of RIP routes may be controlled by source interface and by source gateway. RIP does not support the use of preference to choose between RIP routes. That is left to the protocol metrics.

Due to the nature of OSPF, only the importation of ASE routes may be controlled. OSPF intra-and inter-area routes are always imported into the routing table with a preference of 10. If a tag is specified with the import policy, routes with the specified tag will only be imported.

It is only possible to restrict the importation of OSPF ASE routes when functioning as an AS border router. Like the other interior protocols, preference cannot be used to choose between OSPF ASE routes. That is done by the OSPF costs.

Route-Filter

This component specifies the individual routes which are to be imported or restricted. The preference to be associated with these routes can also be explicitly specified using this component.

The preference associated with the imported routes are inherited unless explicitly specified. If there is no preference specified with a route-filter, then the preference is inherited from the one specified with the import-source.

Every protocol (RIP OSPF, and BGP) has a configurable parameter that specifies the default-preference associated with routes imported to that protocol. If a preference is not explicitly specified with the route-filter, as well as the import-source, then it is inherited from the default-preference associated with the protocol for which the routes are being imported.

Export Policies

Export policies control the redistribution of routes to other systems. They determine which routes are advertised by the Unicast Routing Process to other systems. Every export policy can have up to three components:

- Export-Destination
- Export-Source
- Route-Filter

Export-Destination

This component specifies the destination where the routes are to be exported. It also specifies the attributes associated with the exported routes. The interface, gateway, or the autonomous system to which the routes are to be redistributed are a few examples of export-destinations. The metric, type, tag, and AS-Path are a few examples of attributes associated with the exported routes.

Export-Source

This component specifies the source of the exported routes. It can also specify the metric to be associated with the routes exported from this source.

The routes to be exported can be identified by their associated attributes:

- Their protocol type (RIP, OSPF, BGP, Static, Direct, Aggregate).
- Interface or the gateway from which the route was received.
- Autonomous system from which the route was learned.
- AS path associated with a route. When BGP is configured, all routes are assigned an AS path when they are added to the routing table. For interior routes, this AS path specifies IGP as the origin and no ASs in the AS path (the current AS is added when the route is exported). For BGP routes, the AS path is stored as learned from BGP.
- Tag associated with a route. OSPF and RIP version 2 currently support tags. All other protocols have a tag of zero.

In some cases, a combination of the associated attributes can be specified to identify the routes to be exported.

Route-Filter

This component specifies the individual routes which are to exported or restricted. The metric to be associated with these routes can also be explicitly specified using this component.

The metrics associated with the exported routes are inherited unless explicitly specified. If there is no metric specified with a route-filter, then the metric is inherited from the one specified with the export-source.

If a metric was not explicitly specified with both the route-filter and the export-source, then it is inherited from the one specified with the export-destination.

Every protocol (RIP, OSPF, and BGP) has a configurable parameter that specifies the default-metric associated with routes exported to that protocol. If a metric is not explicitly specified with the route-filter, export-source as well as export-destination, then it is inherited from the default-metric associated with the protocol to which the routes are being exported.

Specifying a Route Filter

Routes are filtered by specifying a route-filter that will match a certain set of routes by destination, or by destination and mask. Among other places, route filters are used with martians and in import and export policies.

The action taken when no match is found is dependent on the context. For instance, a route that does not match any of the route-filters associated with the specified import or export policies is rejected.

A route will match the most specific filter that applies. Specifying more than one filter with the same destination, mask, and modifiers generates an error.

There are three possible formats for a route filter. Not all of these formats are available in all places. In most cases, it is possible to associate additional options with a filter. For example, while creating a martian, it is possible to specify the **allow** option, while creating an import policy, one can specify a **preference**, and while creating an export policy one can specify a **metric**.

The three forms of a route-filter are:

- Network [exact | refines | between number,number]
- Network/mask [exact | refines | between number,number]
- Network/masklen [exact | refines | between number,number]

Matching usually requires both an address and a mask, although the mask is implied in the shorthand forms listed below. These three forms vary in how the mask is specified. In the first form, the mask is implied to be the natural mask of the network. In the second, the mask is explicitly specified. In the third, the mask is specified by the number of contiguous one bits.

If no optional parameters (exact, refines, or between) are specified, any destination that falls in the range given by the network and mask is matched, so the mask of the destination is ignored. If a natural network is specified, the network, any subnets, and any hosts will be matched. Three optional parameters that cause the mask of the destination to also be considered are:

- **Exact:** Specifies that the mask of the destination must match the supplied mask exactly. This is used to match a network, but no subnets or hosts of that network.
- **Refines:** Specifies that the mask of the destination must be more specified (i.e., longer) than the filter mask. This is used to match subnets and/or hosts of a network, but not the network.
- **Between number, number:** Specifies that the mask of the destination must be as or more specific (i.e., as long as or longer) than the lower limit (the first number parameter) and no more specific (i.e., as long as or shorter) than the upper limit (the second number). Note that exact and refines are both special cases of between.

Aggregates and Generates

Route aggregation is a method of generating a more general route, given the presence of a specific route. It is used, for example, at an autonomous system border to generate a route to a network to be advertised via BGP given the presence of one or more subnets of that network learned via OSPF. The routing process does not perform any aggregation unless explicitly requested.

Route aggregation is also used by regional and national networks to reduce the amount of routing information passed around. With careful allocation of network addresses to clients, regional networks can just announce one route to regional networks instead of hundreds.

Aggregate routes are not actually used for packet forwarding by the originator of the aggregate route, but only by the receiver (if it wishes). Instead of requiring a route-peer to know about individual subnets which would increase the size of its routing table, the peer is only informed about an aggregate-route which contains all the subnets.

Like export policies, aggregate-routes can have up to three components:

- Aggregate-Destination
- Aggregate-Source
- Route-Filter

Aggregate-Destination

This component specifies the aggregate/summarized route. It also specifies the attributes associated with the aggregate route. The preference to be associated with an aggregate route can be specified using this component.

Aggregate-Source

This component specifies the source of the routes contributing to an aggregate/summarized route. It can also specify the preference to be associated with the contributing routes from this source. This preference can be overridden by explicitly specifying a preference with the route-filter.

The routes contributing to an aggregate can be identified by their associated attributes:

- Protocol type (RIP, OSPF, BGP, Static, Direct, Aggregate).
- Autonomous system from which the route was learned.
- AS path associated with a route. When BGP is configured, all routes are assigned an AS path when they are added to the routing table. For interior routes, this AS path specifies IGP as the origin and no ASs in the AS path (the current AS is added when the route is exported). For BGP routes, the AS path is stored as learned from BGP.
- Tag associated with a route. OSPF and RIP version 2 currently support tags. All other protocols have a tag of zero.

In some cases, a combination of the associated attributes can be specified to identify the routes contributing to an aggregate.

Route-Filter

This component specifies the individual routes that are to be aggregated or summarized. The preference to be associated with these routes can also be explicitly specified using this component.

The contributing routes are ordered according to the aggregation preference that applies to them. If there is more than one contributing route with the same aggregating preference, the route's own preferences are used to order the routes. The preference of the aggregate route will be that of contributing route with the lowest aggregate preference.

A route may only contribute to an aggregate route that is more general than itself; it must match the aggregate under its mask. Any given route may only contribute to one aggregate route, which will be the most specific configured, but an aggregate route may contribute to a more general aggregate.

An aggregate-route only comes into existence if at least one of its contributing routes is active.

Authentication

Authentication guarantees that routing information is only imported from trusted routers. Many protocols like RIP v2 and OSPF provide mechanisms for authenticating protocol exchanges. A variety of authentication schemes can be used. Authentication has two components – an Authentication Method and an Authentication Key. Many protocols allow different authentication methods and keys to be used in different parts of the network.

Authentication Methods

There are mainly two authentication methods:

Simple Password: In this method, an authentication key of up to 8 characters is included in the packet. If this does not match what is expected, the packet is discarded. This method provides little security, as it is possible to learn the authentication key by watching the protocol packets.

MD5: This method uses the MD5 algorithm to create a crypto-checksum of the protocol packet and an authentication key of up to 16 characters. The transmitted packet does not contain the authentication key itself; instead, it contains a crypto-checksum, called the digest. The receiving router performs a calculation using the correct authentication key and discards the packet if the digest does not match. In addition, a sequence number is maintained to prevent the replay of older packets. This method provides a much stronger assurance that routing data originated from a router with a valid authentication key.

Many protocols allow the specification of two authentication keys per interface. Packets are always sent using the primary keys, but received packets are checked with both the primary and secondary keys before being discarded.

Authentication Keys and Key Management

An authentication key permits the generation and verification of the authentication field in protocol packets. In many situations, the same primary and secondary keys are used on several interfaces of a router. To make key management easier, the concept of a *key-chain* was introduced. Each key-chain has an identifier and can contain up to two keys. One key is the primary key and other is the secondary key. Outgoing packets use the primary authentication key, but incoming packets may match either the primary or secondary authentication key. In Configure mode, instead of specifying the key for each interface (which can be up to 16 characters long), you can specify a key-chain identifier.

The X-Pedition router supports MD5 specification of OSPF RFC 2178 which uses the MD5 algorithm and an authentication key of up to 16 characters. Thus there are now three authentication schemes available per interface: none, simple and RFC 2178 OSPF MD5 authentication. It is possible to configure different authentication schemes on different interfaces.

RFC 2178 allows multiple MD5 keys per interface. Each key has two times associated with the key:

- A time period that the key will be generated
- A time period that the key will be accepted

The X-Pedition router only allows one MD5 key per interface. Also, there are no options provided to specify the time period during which the key would be generated and accepted; the specified MD5 key is always generated and accepted.

Configuring Simple Routing Policies

Simple routing policies provide an efficient way for routing information to be exchanged between routing protocols. The **redistribute** command can be used to redistribute routes from one routing domain into another routing domain. Redistribution of routes between routing domains is based on route policies. A route policy is a set of conditions based on which routes are redistributed. While the **redistribute** command may fulfill the export policy requirement for most users, complex export policies may require the use of the commands discussed in *Export Policies* on page 288.

The general syntax of the IPv4 redistribute command is:

ip-router policy redistribute from-proto <*protocol*> **to-proto** <*protocol*>
[**network** <*ipAddr/mask*> | **all** | **default**] [**exact**|**refines**|**between** <*low-high*>]
[**metric** <*number*>] [**restrict**] [**source-as** <*number*>] [**target-as** <*number*>] [**tag**] [**ase-type**]

The **from-proto** parameter specifies the protocol of the source routes. The values for the **from-proto** parameter can be **rip**, **ospf**, **bgp**, **direct**, **static**, **aggregate** and **ospf-ase**. The **to-proto** parameter specifies the destination protocol where the routes are to be exported. The values for the **to-proto** parameter can be **rip**, **ospf**, **bgp**, or **ospfnssa**. The network parameter provides a means to define a filter for the routes to be distributed. The network parameter defines a filter that is made up of an IP address and a mask. Routes that match the filter are considered eligible for redistribution.

The general syntax of the IPv6 redistribute command is similar to the IPv4 redistribute command:

ipv6-router policy redistribute from-proto <*protocol*> **to-proto** <*protocol*> [**network** <*ipv6Addr/prefix*> | **default** | **all**] [**metric** <*number*>**| restrict**]

The **from-proto** parameter values can be **ripng**, **direct**, or **static**. The only valid value for the **to-proto** parameter is **ripng**.

Every protocol (RIP, RIPng, OSPF, and BGP) has a configurable parameter that specifies the default-metric associated with routes exported to that protocol. If a metric is not explicitly specified with the redistribute command, then it is inherited from the default-metric associated with the protocol to which the routes are being exported.

Redistributing Static Routes

Static routes may be redistributed to another routing protocol such as RIP, RIPng, or OSPF by the following commands. The **network** parameter specifies the set of static routes that will be redistributed by this command. If all static routes are to be redistributed, set the **network** parameter to **all**. Note that the **network** parameter is a filter that specifies routes that are to be redistributed.

To redistribute static routes, enter one of the following commands in Configure mode.

Redistribute static routes into RIP	**ip-router policy redistribute from-proto static to-proto rip network all**
Redistribute static routes into RIPng	**ipv6-router policy redistribute from-proto static to-proto ripng network all**
Redistribute static routes into OSPF	**ip-router policy redistribute from-proto static to-proto ospf network all**

Redistributing Directly Attached Networks

Routes to directly attached networks are redistributed to another routing protocol such as RIP or OSPF by the following commands. The **network** parameter specifies a set of routes that will be redistributed by this command. If all direct routes are to be redistributed set the **network** parameter to **all**. Note that the **network** parameter is a filter that specifies routes that are to be redistributed.

To redistribute direct routes, enter one of the following commands in Configure mode:

Redistribute direct routes into RIP	**ip-router policy redistribute from-proto direct to-proto rip network all**
Redistribute direct routes into RIPng	**ipv6-router policy redistribute from-proto direct to-proto ripng network all**
Redistribute direct routes into OSPF	**ip-router policy redistribute from-proto direct to-proto ospf network all**

Redistributing RIP/RIPng into RIP/RIPng

The X-Pedition routing process requires RIP/RIPng redistribution into RIP/RIPng if a protocol is redistributed into RIP/RIPng.

To redistribute RIP into RIP, enter the following in Configure mode:

Redistribute RIP into RIP	**ip-router policy redistribute from-proto rip to-proto rip**
Redistribute RIPng into RIPng	**ipv6-router policy redistribute from-proto ripng to-proto ripng**

Redistributing RIP into OSPF

RIP routes may be redistributed to OSPF. Note that the IPv6 implementation does not support OSPF.

To redistribute RIP into OSPF, enter the following command in Configure mode:

Redistribute RIP into OSPF	**ip-router policy redistribute from-proto rip to-proto ospf**

Redistributing OSPF to RIP

For the purposes of route redistribution and import-export policies, OSPF intra- and inter-area routes are referred to as **ospf** routes, and external routes redistributed into OSPF are referred to as **ospf-ase** routes. Examples of **ospf-ase** routes include **static** routes, **rip** routes, **direct** routes, **bgp** routes, or **aggregate** routes, which are redistributed into an OSPF domain.

OSPF routes may be redistributed into RIP. To redistribute OSPF into RIP, enter the following commands in Configure mode. Note that the IPv6 implementation does not support OSPF.

Redistribute ospf-ase routes into RIP	**ip-router policy redistribute from-proto ospf-ase to-proto rip**
Redistribute ospf routes into RIP	**ip-router policy redistribute from-proto ospf to-proto rip**

Redistributing Aggregate Routes

The **aggregate** parameter causes an aggregate route with the specified IP address and subnet mask to be redistributed.

Note: The aggregate route must first be created using the **aggr-gen** command**.** This command creates a specified aggregate route for routes that match the aggregate.

To redistribute aggregate routes, enter one of the following commands in Configure mode:

Redistribute aggregate routes into RIP	**ip-router policy redistribute from-proto aggregate to-proto rip**
Redistribute aggregate routes into OSPF	**ip-router policy redistribute from-proto aggregate to-proto ospf**

Simple Route Redistribution Examples

Example 1: Redistribution into RIP

For all examples given in this section, refer to the configurations shown in Figure 33 on page 308.

Note: These IPv4 examples could apply to IPv6, also, by using the comparable IPv6 commands.

The following configuration commands for router R1:

- Determine the IP address for each interface
- Specify the static routes configured on the router
- Determine its RIP configuration

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Create the various IP interfaces.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 interface create ip to-r2 address-netmask 120.190.1.1/16 port et.1.2
 interface create ip to-r3 address-netmask 130.1.1.1/16 port et.1.3
 interface create ip to-r41 address-netmask 140.1.1.1/24 port et.1.4
 interface create ip to-r42 address-netmask 140.1.2.1/24 port et.1.5
 interface create ip to-r6 address-netmask 160.1.1.1/16 port et.1.6
 interface create ip to-r7 address-netmask 170.1.1.1/16 port et.1.7
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure a default route through 170.1.1.7
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route default gateway 170.1.1.7
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure static routes to the 135.3.0.0 subnets reachable through
! R3.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route 135.3.1.0/24 gateway 130.1.1.3
 ip add route 135.3.2.0/24 gateway 130.1.1.3
 ip add route 135.3.3.0/24 gateway 130.1.1.3
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route 202.1.0.0/16 gateway 120.190.1.2
 ip add route 160.1.5.0/24 gateway 120.190.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! RIP Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 rip start
 rip set default-metric 2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! RIP Interface Configuration. Create a RIP interfaces, and set
! their type to (version II, multicast).
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 rip add interface to-r41
 rip add interface to-r42
```

```
rip add interface to-r6
rip set interface to-r41 version 2 type multicast
rip set interface to-r42 version 2 type multicast
rip set interface to-r6 version 2 type multicast
```

Exporting a Given Static Route to All RIP Interfaces

Router R1 has several static routes of which one is the default route. We would export this default route over all RIP interfaces.

```
ip-router policy redistribute from-proto static to-proto rip network default
```

Exporting All Static Routes to All RIP Interfaces

Router R1 has several static routes. We would export these routes over all RIP interfaces.

```
ip-router policy redistribute from-proto static to-proto rip network all
```

Exporting All Static Routes Except the Default Route to All RIP Interfaces

Router R1 has several static routes. We would export all these routes except the default route to all RIP interfaces.

```
ip-router policy redistribute from-proto static to-proto rip network all
ip-router policy redistribute from-proto static to-proto rip network default restrict
```

Example 2: Redistribution into OSPF

For all examples given in this section, refer to the configurations shown in Figure 34 on page 311.

Note: These examples apply to IPv4 networks, only.

The following configuration commands for router R1:

- Determine the IP address for each interface
- Specify the static routes configured on the router
- Determine its OSPF configuration

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Create the various IP interfaces.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  interface create ip to-r2 address-netmask 120.190.1.1/16 port et.1.2
  interface create ip to-r3 address-netmask 130.1.1.1/16 port et.1.3
  interface create ip to-r41 address-netmask 140.1.1.1/24 port et.1.4
  interface create ip to-r42 address-netmask 140.1.2.1/24 port et.1.5
  interface create ip to-r6 address-netmask 140.1.3.1/24 port et.1.6
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ip add route 202.1.0.0/16 gateway 120.1.1.2
  ip add route 160.1.5.0/24 gateway 120.1.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! OSPF Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ospf start
  ospf create area 140.1.0.0
  ospf create area backbone
  ospf set ase-defaults cost 4
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! OSPF Interface Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ospf add interface 140.1.1.1 to-area 140.1.0.0
  ospf add interface 140.1.2.1 to-area 140.1.0.0
  ospf add interface 140.1.3.1 to-area 140.1.0.0
  ospf add interface 130.1.1.1 to-area backbone
```

Exporting All Interface & Static Routes to OSPF

Router R1 has several static routes. We would like to export all these static routes and direct-routes (routes to connected networks) into OSPF.

```
ip-router policy redistribute from-proto static to-proto ospf
ip-router policy redistribute from-proto direct to-proto ospf
```

Note: The network parameter specifying the network-filter is optional. The default value for this parameter is **all**, indicating all networks. Since in the previous example, we would like to export all static and direct routes into OSPF, we have not specified this parameter.

Exporting All RIP, Interface & Static Routes to OSPF

Note: Also export interface, static, RIP, OSPF, and OSPF-ASE routes into RIP.

In the configuration shown in Figure 34 on page 311, suppose we decide to run RIP Version 2 on network 120.190.0.0/16, connecting routers R1 and R2.

Router R1 would like to export all RIP, interface, and static routes to OSPF.

```
ip-router policy redistribute from-proto rip to-proto ospf
ip-router policy redistribute from-proto direct to-proto ospf
ip-router policy redistribute from-proto static to-proto ospf
```

Router R1 would also like to export interface, static, RIP, OSPF, and OSPF-ASE routes into RIP.

```
ip-router policy redistribute from-proto direct to-proto rip
ip-router policy redistribute from-proto static to-proto rip
ip-router policy redistribute from-proto rip to-proto rip
ip-router policy redistribute from-proto ospf to-proto rip
ip-router policy redistribute from-proto ospf-ase to-proto rip
```

Configuring Advanced Routing Policies

Advanced Routing Policies are used for creating complex import/export policies that cannot be done using the redistribute command. Advanced export policies provide granular control over the targets where the routes are exported, the source of the exported routes, and the individual routes which are exported. They provide the capability to send different routes to the various route-peers. They can be used to provide the same route with different attributes to the various route-peers.

Import policies control the importation of routes from routing protocols and their installation in the routing database (Routing Information Base and Forwarding Information Base). Import policies determine which routes received from other systems are used by the X-Pedition routing process. Using import policies, it is possible to ignore route updates from an unreliable peer and give better preference to routes learned from a trusted peer.

Export Policies

Advanced export policies can be constructed from one or more of the following building blocks:

- Export Destinations - This component specifies the destination where the routes are to be exported. It also specifies the attributes associated with the exported routes. The interface, gateway or the autonomous system to which the routes are to be redistributed are a few examples of export-destinations. The metric, type, tag, and AS-Path are a few examples of attributes associated with the exported routes.
- Export Sources - This component specifies the source of the exported routes. It can also specify the metric to be associated with the routes exported from this source. The routes to be exported can be identified by their associated attributes, such as protocol type, interface or the gateway from which the route was received, and so on.
- Route Filter - This component provides the means to define a filter for the routes to be distributed. Routes that match a filter are considered as eligible for redistribution. This can be done using one of two methods:
  - Creating a route-filter (with the **ip-router policy create filter** command) and associating an identifier with it. A route-filter has several network specifications associated with it. Every route is checked against the set of network specifications associated with all route-filters to determine its eligibility for redistribution. The identifier associated with a route-filter is used in the **ip-router policy export** command.
  - Specifying the networks as needed in the **ip-router policy export** command.

  If you want to create a complex route-filter, and you intend to use that route-filter in several export policies, then the first method is recommended. It you do not have complex filter requirements, then use the second method.

After you create one or more building blocks, they are tied together by the **ip/ipv6-router policy export** command.

To create route export policies, enter the following in Configure mode:

Create an export policy for IPv4.	**ip-router policy export destination** <*exp-dest-id*> [**source** <*exp-src-id*>] [**filter** <*filter-id*>] [**network** <*ipAddr/mask*> | **all** | **default**] [**exact**|**refines**|**between** <*low-high*>] [**metric** <*number*>] | [**restrict**]

The <*exp-dest-id*> is the identifier of the export-destination which determines where the routes are to be exported. If no routes to a particular destination are to be exported, then no additional parameters are required.

The <*exp-src-id*>, if specified, is the identifier of the export-source which determines the source of the exported routes. If a export-policy for a given export-destination has more than one export-source, then the **ip-router policy export destination** <*exp-dest-id*> command should be repeated for each <*exp-src-id*>.

The <*filter-id*>, if specified, is the identifier of the route-filter associated with this export-policy. If there is more than one route-filter for any export-destination and export-source combination, then the **ip-router policy export destination** <*exp-dest-id*> **source** <*exp-src-id*> command should be repeated for each <*filter-id*>.

Creating an Export Destination

To create an export destination, enter one the following commands in Configure mode:

Create a RIP export destination.	**ip-router policy create rip-export-destination**
Create an OSPF export destination. (IPv4 only)	**ip-router policy create ospf-export-destination**
Create an OSPF NSSA export destination. (IPv4 only)	**ip-router policy create ospfnssa-export-destination**
Create a BGP export destination (IPv4 only)	**ip-router policy create bgp-export-destination**

Note: For IPv4 networks, when you use the **ip-router policy create rip-export-destination** command in conjunction with the **gateway** option in **ip-router policy export**, you must use **rip add source-gateways** for each address indicated in the gateway option.

Creating an Export Source

To create an export source, enter one of the following commands in Configure mode:

Create a RIP export source.	**ip-router policy create rip-export-source**
Create a BGP export source (IPv4 only).	**ip-router policy create bgp-export-source**
Create an OSPF export source (IPv4 only).	**ip-router policy create ospf-export-source** <*number-or-string*> {**metric** <*num*> | **restrict** | **noagg** | **tag** <*num*> | **sequence-number** <*num*>}
Create an OSPF ASE export source (IPv4 only).	**ip-router policy create ospfase-export-source** <*number-or-string*> {**metric** <*num*> | **restrict** | **noagg** | **tag** <*num*> | **sequence-number** <*num*>}
Create an AS path export source (IPv4 only).	**ip-router policy create aspath-export-source**
Create an IPv4 tag export source.	**ip-router policy create tag-export-source**
Create an IPv4 directly connected export source.	**ip-router policy create direct-export-source**
Create an IPv4 source for exporting static routes.	**ip-router policy create static-export-source**

Import Policies

Import policies can be constructed from one or more of the following building blocks:

- Import-source - This component specifies the source of the imported routes. It can also specify the preference to be associated with the routes imported from this source. The routes to be imported can be identified by their associated attributes, including source protocol, source interface, or gateway from which the route was received, and so on.
- Route Filter - This component provides the means to define a filter for the routes to be imported. Routes that match a filter are considered as eligible for importation. This can be done using one of two methods:
  - Creating a route-filter and associating an identifier with it. A route-filter has several network specifications associated with it. Every route is checked against the set of network specifications associated with all route-filters to determine its eligibility for importation. The identifier associated with a route-filter is used in the **ip-router policy import** command.
  - Specifying the networks as needed in the **ip-router policy import** command.

  If you want to create a complex route-filter, and you intend to use that route-filter in several import policies, then the first method is recommended. It you do not have complex filter requirements, then use the second method.

After you create one or more building blocks, they are tied together by the **ip-router policy import** command.

To create route import policies, enter the following in Configure mode:

Create an import policy (IPv4).	**ip-router policy import source** <*imp-src-id*> [**filter** <*filter-id*>] [**network** <*ipAddr/mask*> | **all** | **default**] [**exact**|**refines**|**between** <*low-high*>] [**preference** <*number*>] [**restrict**] [**fromribs unicast** | **multicast** | **both**] [**unicast**] [**multicast**]

The <*imp-src-id*> is the identifier of the import-source that determines the source of the imported routes. If no routes from a particular source are to be imported, then no additional parameters are required.

The <*filter-id*>, if specified, is the identifier of the route-filter associated with this import-policy. If there is more than one route-filter for any import-source, then the **ip-router policy import source** <*imp-src-id*> command should be repeated for each <*filter-id*>.

Creating an Import Source

Import sources specify the routing protocol from which the routes are imported. The source may be RIP, BGP, or OSPF.

To create an import source, enter one of the following commands in Configure mode:

Create a RIP import source.	**ip-router policy create rip-import-source**
Create a BGP import source (IPv4 only).	**ip-router policy create bgp-import-source**
Create an OSPF import source (IPv4 only).	**ip-router policy create ospf-import-source**
Create an IPv4 redirect import source.	**ip-router policy create redirect-import-source**

Creating a Route Filter

Route policies are defined by specifying a set of filters that will match a certain route by destination or by destination and mask.

To create route filters, enter the following in Configure mode:

Create a route filter (IPv4).	**ip-router policy create filter** <*number-or-string*> [**network** <*ipAddr/mask*> | **all** | **default**] [**exact** | **refines** | **between** <*low-high*>] [**host-net**]

Creating an Aggregate Route

Route aggregation is a method of generating a more general route, given the presence of a specific route. The routing process does not perform any aggregation unless explicitly requested. Aggregate-routes can be constructed from one or more of the following building blocks:

- Aggregate-Destination - This component specifies the aggregate/summarized route. It also specifies the attributes associated with the aggregate route. The preference to be associated with an aggregate route can be specified using this component.
- Aggregate-Source - This component specifies the source of the routes contributing to an aggregate/summarized route. It can also specify the preference to be associated with the contributing routes from this source. The routes contributing to an aggregate can be identified by their associated attributes, including protocol type, tag associated with a route, and so on.
- Route Filter - This component provides the means to define a filter for the routes to be aggregated or summarized. Routes that match a filter are considered as eligible for aggregation. This can be done using one of two methods:
  - Creating a route-filter and associating an identifier with it. A route-filter has several network specifications associated with it. Every route is checked against the set of network specifications associated with all route-filters to determine its eligibility for aggregation. The identifier associated with a route-filter is used in the **ip-router policy aggr-gen** command.
  - Specifying the networks as needed in the **ip-router policy aggr-gen** command.
- If you want to create a complex route-filter, and you intend to use that route-filter in several aggregates, then the first method is recommended. It you do not have complex filter requirements, then use the second method.

After you create one or more building blocks, they are tied together by the **ip-router policy aggr-gen** command. To create aggregates, enter the following in Configure mode:

Create an aggregate route (IPv4).	**ip-router policy aggr-gen destination** <*number-or-string*> [**source** <*number-or-string*>] [**filter** <*number-or-string*>] | [**network** <*ipAddr/mask*> | **all** | **default**] [**exact**| **refines** | **between** <*low-high*>] [**preference** <*number*>| **restrict**] [**multicast** | **unicast**]

The <*aggr-dest-id*> is the identifier of the aggregate-destination that specifies the aggregate/summarized route.

The <*aggr-src-id*> is the identifier of the aggregate-source that contributes to an aggregate route. If an aggregate has more than one aggregate-source, then the **ip-router policy aggr-gen destination** <*aggr-dest-id*> command should be repeated for each <*aggr-src-id*>.

The <*filter-id*> is the identifier of the route-filter associated with this aggregate. If there is more than one route-filter for any aggregate-destination and aggregate-source combination, then the **ip-router policy aggr-gen destination** <*aggr-dest-id*> **source** <*aggr-src-id*> command should be repeated for each <*filter-id*>.

Creating an Aggregate Destination

To create an aggregate destination, enter the following in Configure mode:

Create an IPv4 aggregate destination.	**ip-router policy create aggr-gen-dest** <*number-or-string*> [**network** <*ipAddr/mask*> | **default**] [**type aggregate**|**generation**] | [**preference** <*num*>] | **brief** | **bgp** | **noinstall** | **unicast** | **multicast**

Creating an Aggregate Source

To create an aggregate source, enter the following in Configure mode:

Create an IPv4 aggregate source.	**ip-router policy create aggr-gen-source** <*number-or-string*> **protocol** <*string*> | **autonomou7-system** <*num*> | **aspath-regular-expression** <*string*> | **tag** <*num*> | **preference** <*num*> | **origin** <*string*> | **optional-attributes-list** <*num-or-string*> | **restrict**

Examples of Import Policies

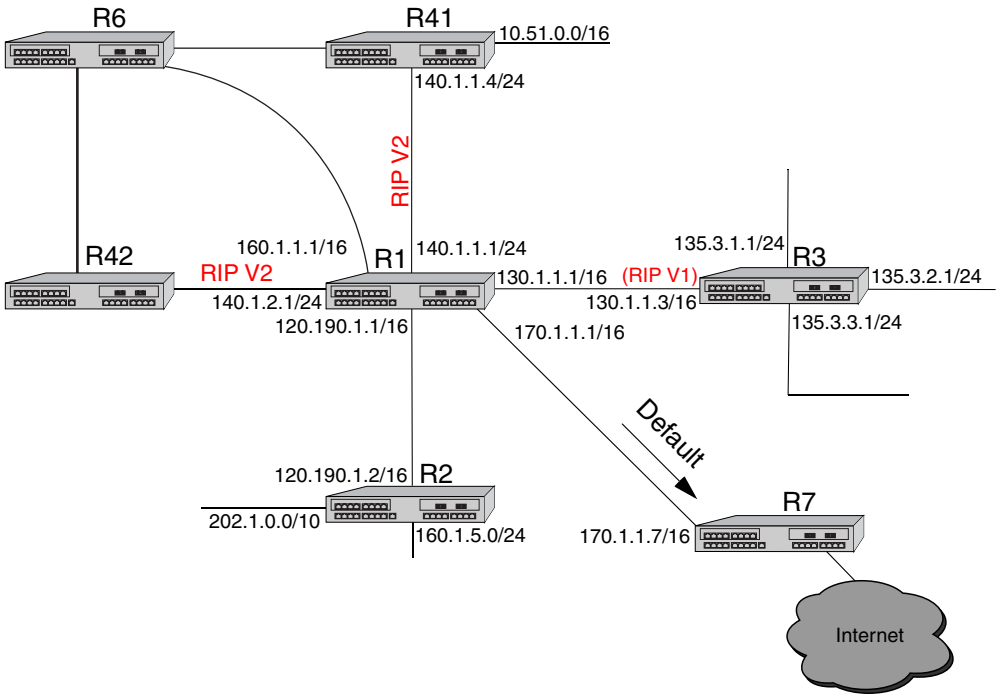
Example 1: Importing from RIP

The importation of RIP routes may be controlled by any of protocol, source interface, or source gateway. If more than one is specified, they are processed from most general (protocol) to most specific (gateway).

RIP does not support the use of preference to choose between routes of the same protocol. That is left to the protocol metrics.

For all examples in this section, refer to the configuration shown in Figure 33.

Figure 33. RIP Configuration



The following configuration commands for router R1:

- Determine the IP address for each interface.
- Specify the static routes configured on the router.
- Determine its RIP configuration.

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Create the various IP interfaces.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
interface create ip to-r2  address-netmask  120.190.1.1/16 port et.1.2
interface create ip to-r3  address-netmask  130.1.1.1/16 port et.1.3
interface create ip to-r41 address-netmask  140.1.1.1/24 port et.1.4
interface create ip to-r42 address-netmask  140.1.2.1/24 port et.1.5
interface create ip to-r6  address-netmask  160.1.1.1/16 port et.1.6
interface create ip to-r7  address-netmask  170.1.1.1/16 port et.1.7
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure a default route through 170.1.1.7
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ip add route default gateway 170.1.1.7
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure default routes to the 135.3.0.0 subnets reachable through
! R3.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ip add route 135.3.1.0/24 gateway 130.1.1.3
ip add route 135.3.2.0/24 gateway 130.1.1.3
ip add route 135.3.3.0/24 gateway 130.1.1.3
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ip add route 202.1.0.0/16 gateway 120.190.1.2
ip add route 160.1.5.0/24 gateway 120.190.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! RIP Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
rip start
rip set default-metric 2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! RIP Interface Configuration. Create a RIP interfaces, and set
! their type to (version II, multicast).
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
rip add interface to-r41
rip add interface to-r42
rip add interface to-r6
rip set interface to-r41 version 2 type multicast
rip set interface to-r42 version 2 type multicast
rip set interface to-r6  version 2 type multicast
```

Importing a Selected Subset of Routes from One RIP Trusted Gateway

Note: This example applies to IPv4 networks only.

Router R1 has several RIP peers. Router R41 has an interface on the network 10.51.0.0. By default, router R41 advertises network 10.51.0.0/16 in its RIP updates. Router R1 would like to import all routes except the 10.51.0.0/16 route from its peer R41.

1. Add the peer 140.1.1.41 to the list of trusted and source gateways.

```
rip add source-gateways 140.1.1.41
rip add trusted-gateways 140.1.1.41
```

2. Create a RIP import source with the gateway as 140.1.1.41 since we would like to import all routes except the 10.51.0.0/16 route from this gateway.

```
ip-router policy create rip-import-source ripImpSrc144 gateway 140.1.1.41
```

3. Create the Import-Policy, importing all routes except the 10.51.0.0/16 route from gateway 140.1.1.41.

```
ip-router policy import source ripImpSrc144 network all
ip-router policy import source ripImpSrc144 network 10.51.0.0/16 restrict
```

Importing a Selected Subset of Routes from All RIP Peers Accessible Over a Certain Interface

Router R1 has several RIP peers. Router R41 has an interface on the network 10.51.0.0. By default, router R41 advertises network 10.51.0.0/16 in its RIP updates. Router R1 would like to import all routes except the 10.51.0.0/16 route from all its peer which are accessible over interface 140.1.1.1.

1. Create a RIP import source with the interface as 140.1.1.1, since we would like to import all routes except the 10.51.0.0/16 route from this interface.

```
ip-router policy create rip-import-source ripImpSrc140 interface 140.1.1.1
```

2. Create the Import-Policy importing all routes except the 10.51.0.0/16 route from interface 140.1.1.1

```
ip-router policy import source ripImpSrc140 network all
ip-router policy import source ripImpSrc140 network 10.51.0.0/16 restrict
```

Example 2: Importing from OSPF

Note: The examples in this section apply to IPv4 networks only.

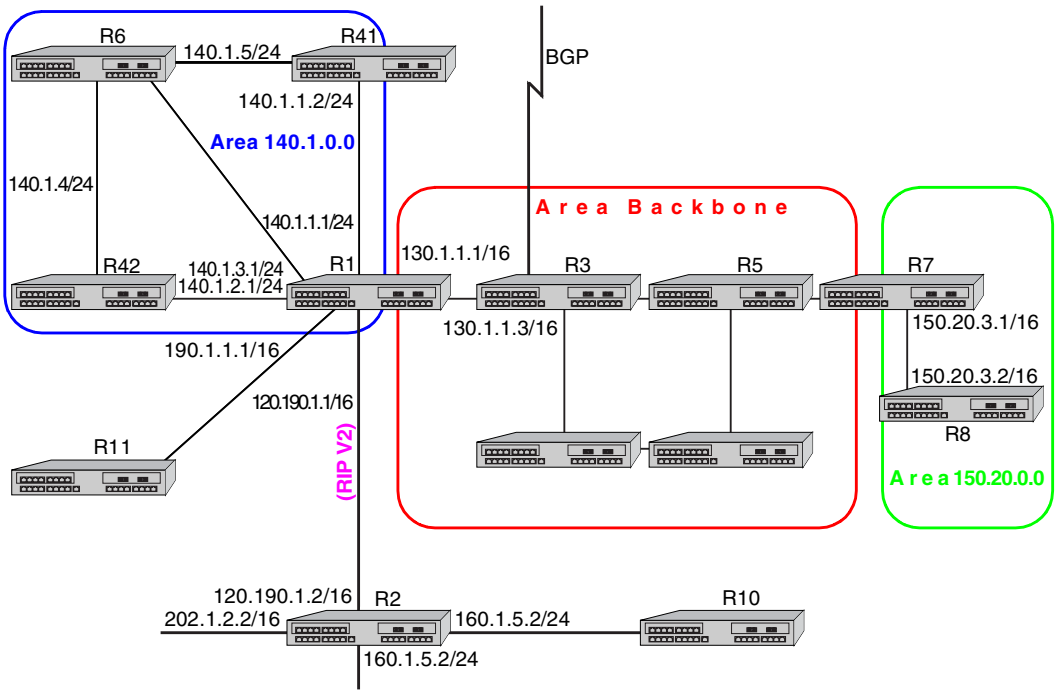
Due to the nature of OSPF, only the importation of ASE routes may be controlled. OSPF intra-and inter-area routes are always imported into the X-Pedition routing table with a preference of 10. If a tag is specified, the import clause will only apply to routes with the specified tag.

It is only possible to restrict the importation of OSPF ASE routes when functioning as an AS border router.

Like the other interior protocols, preference cannot be used to choose between OSPF ASE routes. That is done by the OSPF costs. Routes that are rejected by policy are stored in the table with a negative preference.

For all examples in this section, refer to the configuration shown in Figure 34.

Figure 34. OSPF Configuration



The following configuration commands for router R1:

- Determine the IP address for each interface
- Specify the static routes configured on the router
- Determine its OSPF configuration

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Create the various IP interfaces.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 interface create ip to-r2  address-netmask  120.190.1.1/16 port et.1.2
 interface create ip to-r3  address-netmask  130.1.1.1/16 port et.1.3
 interface create ip to-r41 address-netmask  140.1.1.1/24 port et.1.4
 interface create ip to-r42 address-netmask  140.1.2.1/24 port et.1.5
 interface create ip to-r6  address-netmask  140.1.3.1/24 port et.1.6
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route 202.1.0.0/16 gateway 120.1.1.2
 ip add route 160.1.5.0/24 gateway 120.1.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! OSPF Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ospf start
 ospf create area 140.1.0.0
 ospf create area backbone
 ospf set ase-defaults cost 4
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! OSPF Interface Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ospf add interface 140.1.1.1 to-area 140.1.0.0
 ospf add interface 140.1.2.1 to-area 140.1.0.0
 ospf add interface 140.1.3.1 to-area 140.1.0.0
 ospf add interface 130.1.1.1 to-area backbone
```

Importing a Selected Subset of OSPF-ASE Routes

1. Create a OSPF import source so that only routes that have a tag of 100 are considered for importation.

   ```
   ip-router policy create ospf-import-source ospfImpSrct100 tag 100
   ```

2. Create the Import-Policy importing all OSPF ASE routes with a tag of 100 except the default ASE route.

   ```
   ip-router policy import source ospfImpSrct100 network all
   ip-router policy import source ospfImpSrct100 network default restrict
   ```

Examples of Export Policies

Example 1: Exporting to RIP

Exporting to RIP is controlled by any of protocol, interface or gateway. If more than one is specified, they are processed from most general (protocol) to most specific (gateway).

It is not possible to set metrics for exporting RIP routes into RIP. Attempts to do this are silently ignored.

If no export policy is specified, RIP and interface routes are exported into RIP. If any policy is specified, the defaults are overridden; it is necessary to explicitly specify everything that should be exported.

RIP version 1 assumes that all subnets of the shared network have the same subnet mask so it is only able to propagate subnets of that network. RIP version 2 removes that restriction and is capable of propagating all routes when not sending version 1 compatible updates.

To announce routes which specify a next hop of the loopback interface (i.e., static and internally generated default routes) via RIP, it is necessary to specify the metric at some level in the export policy. Just setting a default metric for RIP is not sufficient. This is a safeguard to verify that the announcement is intended.

For all examples in this section, refer to the configuration shown in Figure 33 on page 308.

The following configuration commands for router R1:

- Determine the IP address for each interface
- Specify the static routes configured on the router
- Determine its RIP configuration

```
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Create the various IP interfaces.
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 interface create ip to-r2  address-netmask  120.190.1.1/16 port et.1.2
 interface create ip to-r3  address-netmask  130.1.1.1/16 port et.1.3
 interface create ip to-r41 address-netmask  140.1.1.1/24 port et.1.4
 interface create ip to-r42 address-netmask  140.1.2.1/24 port et.1.5
 interface create ip to-r6  address-netmask  160.1.1.1/16 port et.1.6
 interface create ip to-r7  address-netmask  170.1.1.1/16 port et.1.7
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Configure a default route through 170.1.1.7
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route default gateway 170.1.1.7
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Configure default routes to the 135.3.0.0 subnets reachable through
 ! R3.
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
 ip add route 135.3.1.0/24 gateway 130.1.1.3
 ip add route 135.3.2.0/24 gateway 130.1.1.3
 ip add route 135.3.3.0/24 gateway 130.1.1.3
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ !
Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route 202.1.0.0/16 gateway 120.190.1.2
 ip add route 160.1.5.0/24 gateway 120.190.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! RIP Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 rip start
 rip set default-metric 2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! RIP Interface Configuration. Create a RIP interfaces, and set
 ! their type to (version II, multicast).
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 rip add interface to-r41
 rip add interface to-r42
 rip add interface to-r6
 rip set interface to-r41 version 2 type multicast
 rip set interface to-r42 version 2 type multicast
 rip set interface to-r6 version 2 type multicast
```

Exporting a Given Static Route to All RIP Interfaces

Router R1 has several static routes, of which one is the default route. We would export this default route over all RIP interfaces.

1. Create a RIP export destination since we would like to export routes into RIP.

   ```
   ip-router policy create rip-export-destination ripExpDst
   ```

Note: In IPv4 networks, when you use the **ip-router policy create rip-export-destination** command in conjunction with the **gateway** option in **ip-router policy export**, you must use **rip add source-gateways** for each address indicated in the gateway option.

2. Create a Static export source since we would like to export static routes.

   ```
   ip-router policy create static-export-source statExpSrc
   ```

   If no export policy is specified, RIP and interface routes are exported into RIP. If any policy is specified, the defaults are overridden; it is necessary to explicitly specify everything that should be exported.

   Since we would also like to export/redistribute RIP and direct routes into RIP, we would also create export-sources for those protocols.

3. Create a RIP export source since we would like to export RIP routes.

```
ip-router policy create rip-export-source ripExpSrc
```

4. Create a Direct export source since we would like to export direct/interface routes.

```
ip-router policy create direct-export-source directExpSrc
```

5. Create the export-policy redistributing the statically created default route, and all (RIP, Direct) routes into RIP.

```
ip-router policy export destination ripExpDst source statExpSrc network default
ip-router policy export destination ripExpDst source ripExpSrc network all
ip-router policy export destination ripExpDst source directExpSrc network all
```

Exporting a Given Static Route to a Specific RIP Interface

In this case, router R1 would export/redistribute the default route over its interface 140.1.1.1 only.

1. Create a RIP export destination for interface with address 140.1.1.1, since we intend to change the rip export policy only for interface 140.1.1.1.

```
ip-router policy create rip-export-destination ripExpDst141 interface 140.1.1.1
```

Note: In IPv4 networks, when you use the **ip-router policy create rip-export-destination** command in conjunction with the **gateway** option in **ip-router policy export**, you must use **rip add source-gateways** for each address indicated in the gateway option.

2. Create a static export source since we would like to export static routes.

```
ip-router policy create static-export-source statExpSrc
```

3. Create a RIP export source since we would like to export RIP routes.

```
ip-router policy create rip-export-source ripExpSrc
```

4. Create a Direct export source since we would like to export direct/interface routes.

```
ip-router policy create direct-export-source directExpSrc
```

5. Create the Export-Policy redistributing the statically created default route, and all (RIP, Direct) routes into RIP.

```
ip-router policy export destination ripExpDst141 source statExpSrc network default
ip-router policy export destination ripExpDst141 source ripExpSrc network all
ip-router policy export destination ripExpDst141 source directExpSrc network all
```

Exporting All Static Routes Reachable Over a Given Interface to a Specific RIP-Interface

In this case, router R1 would export/redistribute all static routes accessible through its interface 130.1.1.1 to its RIP-interface 140.1.1.1 only.

1. Create a RIP export destination for interface with address 140.1.1.1, since we intend to change the rip export policy for interface 140.1.1.1

```
ip-router policy create rip-export-destination ripExpDst141 interface 140.1.1.1
```

Note: In IPv4 networks, when you use the **ip-router policy create rip-export-destination** command in conjunction with the **gateway** option in **ip-router policy export**, you must use **rip add source-gateways** for each address indicated in the gateway option.

2. Create a Static export source since we would like to export static routes.

```
ip-router policy create static-export-source statExpSrc130 interface 130.1.1.1
```

3. Create a RIP export source since we would like to export RIP routes.

```
ip-router policy create rip-export-source ripExpSrc
```

4. Create a Direct export source.

```
ip-router policy create direct-export-source directExpSrc
```

5. Create the Export-Policy, redistributing all static routes reachable over interface 130.1.1.1 and all (RIP, Direct) routes into RIP.

```
ip-router policy export destination ripExpDst141 source statExpSrc130 network all
ip-router policy export destination ripExpDst141 source ripExpSrc network all
ip-router policy export destination ripExpDst141 source directExpSrc network all
```

Exporting Aggregate-Routes into RIP

In the configuration shown in Figure 33 on page 308, suppose you decide to run RIP Version 1 on network 130.1.0.0/16, connecting routers R1 and R3. Router R1 desires to announce the 140.1.1.0/24 and 140.1.2.0/24 networks to router R3. RIP Version 1 does not carry any information about subnet masks in its packets. Thus it would not be possible to announce the subnets (140.1.1.0/24 and 140.1.2.0/24) into RIP Version 1 without aggregating them.

1. Create an Aggregate-Destination which represents the aggregate/summarized route.

```
ip-router policy create aggr-gen-dest aggrDst140 network 140.1.0.0/16
```

2. Create an Aggregate-Source which qualifies the source of the routes contributing to the aggregate. Since in this case, we do not care about the source of the contributing routes, we would specify the protocol as **all**.

```
ip-router policy create aggr-gen-source allAggrSrc protocol all
```

3. Create the aggregate/summarized route. This command binds the aggregated route with the contributing routes.

```
ip-router policy aggr-gen destination aggrDst140 source allAggrSrc network 140.1.1.0/24
ip-router policy aggr-gen destination aggrDst140 source allAggrSrc network 140.1.2.0/24
```

4. Create a RIP export destination for interface with address 130.1.1.1, since we intend to change the rip export policy only for interface 130.1.1.1.

```
ip-router policy create rip-export-destination ripExpDst130 interface 130.1.1.1
```

Note: In IPv4 networks, when you use the **ip-router policy create rip-export-destination** command in conjunction with the **gateway** option in **ip-router policy export**, you must use **rip add source-gateways** for each address indicated in the gateway option.

5. Create a Aggregate export source since we would to export/redistribute an aggregate/summarized route.

```
ip-router policy create aggr-export-source aggrExpSrc
```

6. Create a RIP export source since we would like to export RIP routes.

```
ip-router policy create rip-export-source ripExpSrc
```

7. Create a Direct export source since we would like to export Direct routes.

```
ip-router policy create direct-export-source directExpSrc
```

8. Create the Export-Policy redistributing all (RIP, Direct) routes and the aggregate route 140.1.0.0/16 into RIP.

```
ip-router policy export destination ripExpDst130 source aggrExpSrc network 140.1.0.0/16
ip-router policy export destination ripExpDst130 source ripExpSrc network all
ip-router policy export destination ripExpDst130 source directExpSrc network all
```

Example 2: Exporting to OSPF

Note: The examples in this section apply to IPv4 networks only.

It is not possible to create OSPF intra- or inter-area routes by exporting routes from the X-Pedition routing table into OSPF. It is only possible to export from the X-Pedition routing table into OSPF ASE routes. It is also not possible to control the propagation of OSPF routes within the OSPF protocol.

There are two types of OSPF ASE routes: type 1 and type 2. The default type is specified by the **ospf set ase-defaults type 1|2** command. This may be overridden by a specification in the **ip-router policy create ospf-export-destination** command.

OSPF ASE routes also have the provision to carry a tag. This is an arbitrary 32-bit number that can be used on OSPF routers to filter routing information. The default tag is specified by the **ospf set ase-defaults tag** command. This may be overridden by a tag specified with the **ip-router policy create ospf-export-destination** command.

Interface routes are not automatically exported into OSPF. They have to be explicitly done.

For all examples in this section, refer to the configuration shown in Figure 34 on page 311.

The following configuration commands for router R1:

- Determine the IP address for each interface
- Specify the static routes configured on the router
- Determine its OSPF configuration

```
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Create the various IP interfaces.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 interface create ip to-r2  address-netmask  120.190.1.1/16 port et.1.2
 interface create ip to-r3  address-netmask  130.1.1.1/16 port et.1.3
 interface create ip to-r41 address-netmask  140.1.1.1/24 port et.1.4
 interface create ip to-r42 address-netmask  140.1.2.1/24 port et.1.5
 interface create ip to-r6  address-netmask  140.1.3.1/24 port et.1.6
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! Configure default routes to the other subnets reachable through R2.
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ip add route 202.1.0.0/16 gateway 120.1.1.2
 ip add route 160.1.5.0/24 gateway 120.1.1.2
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! OSPF Box Level Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ospf start
 ospf create area 140.1.0.0
 ospf create area backbone
 ospf set ase-defaults cost 4
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ! OSPF Interface Configuration
!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ospf add interface 140.1.1.1 to-area 140.1.0.0
 ospf add interface 140.1.2.1 to-area 140.1.0.0
 ospf add interface 140.1.3.1 to-area 140.1.0.0
 ospf add interface 130.1.1.1 to-area backbone
```

Exporting All Interface and Static Routes to OSPF

Router R1 has several static routes. We would export these static routes as type-2 OSPF routes. The interface routes would redistributed as type 1 OSPF routes.

1. Create a OSPF export destination for type-1 routes since we would like to redistribute certain routes into OSPF as type 1 OSPF-ASE routes.

   ```
   ip-router policy create ospf-export-destination ospfExpDstType1 type 1 metric 1
   ```

2. Create a OSPF export destination for type-2 routes since we would like to redistribute certain routes into OSPF as type 2 OSPF-ASE routes.

   ```
   ip-router policy create ospf-export-destination ospfExpDstType2 type 2 metric 4
   ```

3. Create a Static export source since we would like to export static routes.

   ```
   ip-router policy create static-export-source statExpSrc
   ```

4. Create a Direct export source since we would like to export interface/direct routes.

```
ip-router policy create direct-export-source directExpSrc
```

5. Create the Export-Policy for redistributing all interface routes and static routes into OSPF.

```
ip-router policy export destination ospfExpDstType1 source directExpSrc network all
ip-router policy export destination ospfExpDstType2 source statExpSrc network all
```

Exporting All RIP, Interface and Static Routes to OSPF

Note: Also export interface, static, RIP, OSPF, and OSPF-ASE routes into RIP.

In the configuration shown in Figure 34 on page 311, suppose we decide to run RIP Version 2 on network 120.190.0.0/16, connecting routers R1 and R2.

We would like to redistribute these RIP routes as OSPF type-2 routes, and associate the tag 100 with them. Router R1 would also like to redistribute its static routes as type 2 OSPF routes. The interface routes would redistributed as type 1 OSPF routes.

Router R1 would like to redistribute its OSPF, OSPF-ASE, RIP, Static and Interface/Direct routes into RIP.

1. Enable RIP on interface 120.190.1.1/16.

```
rip add interface 120.190.1.1
rip set interface 120.190.1.1 version 2 type multicast
```

2. Create a OSPF export destination for type-1 routes.

```
ip-router policy create ospf-export-destination ospfExpDstType1 type 1 metric 1
```

3. Create a OSPF export destination for type-2 routes.

```
ip-router policy create ospf-export-destination ospfExpDstType2 type 2 metric 4
```

4. Create a OSPF export destination for type-2 routes with a tag of 100.

```
ip-router policy create ospf-export-destination ospfExpDstType2t100 type 2 tag 100 metric 4
```

5. Create a RIP export source.

```
ip-router policy export destination ripExpDst source ripExpSrc network all
```

6. Create a Static export source.

```
ip-router policy create static-export-source statExpSrc
```

7. Create a Direct export source.

```
ip-router policy create direct-export-source directExpSrc
```

8. Create the Export-Policy for redistributing all interface, RIP and static routes into OSPF.

```
ip-router policy export destination ospfExpDstType1 source directExpSrc network all
ip-router policy export destination ospfExpDstType2 source statExpSrc network all
ip-router policy export destination ospfExpDstType2t100 source ripExpSrc network all
```

9. Create a RIP export destination.

```
ip-router policy create rip-export-destination ripExpDst
```

Note: In IPv4 networks, when you use the **ip-router policy create rip-export-destination** command in conjunction with the **gateway** option in **ip-router policy export**, you must use **rip add source-gateways** for each address indicated in the gateway option.

10. Create OSPF export source.

```
ip-router policy create ospf-export-source ospfExpSrc metric 1
```

11. Create OSPF-ASE export source.

```
ip-router policy create ospfase-export-source ospfAseExpSrc metric 1
```

12. Create the Export-Policy for redistributing all interface, RIP, static, OSPF and OSPF-ASE routes into RIP.

```
ip-router policy export destination ripExpDst source statExpSrc network all
ip-router policy export destination ripExpDst source ripExpSrc network all
ip-router policy export destination ripExpDst source directExpSrc network all
ip-router policy export destination ripExpDst source ospfExpSrc network all
ip-router policy export destination ripExpDst source ospfAseExpSrc network all
```

Chapter 20

Multicast Routing Configuration Guide

IP Multicast Overview

Multicast routing on the X-Pedition router is supported through DVMRP and IGMP. IGMP is used to determine host membership on directly attached subnets. DVMRP is used to determine forwarding of multicast traffic between X-Pedition routers.

This chapter:

- Provides an overview of the X-Pedition router's implementation of the Internet Group Management Protocol (IGMP)
- Provides an overview of the X-Pedition router's implementation of the Distance Vector Multicast Routing Protocol (DVMRP)
- Discusses configuring DVMRP routing on the X-Pedition router
- Discusses configuring IGMP on the X-Pedition router
- Reviews Protocol Independent Multicast (PIM) configuration

IGMP Overview

The X-Pedition router supports IGMP Version 2.0 as defined in RFC 2236. IGMP runs on a per-IP interface basis and an interface can be configured to run just IGMP. Since multiple physical ports (VLANs) can be configured with the same IP interface on the X-Pedition router, IGMP keeps track of multicast host members on a per-port basis. Ports belonging to an IP VLAN without any IGMP membership will not forward any multicast traffic.

The router allows per-interface control of the host query interval and response time. Query interval is the time between IGMP queries. Response time is the time the router will wait for host responses to IGMP queries. The router can be configured to deny or accept group membership filters.

Note: Before the E10 system firmware release, some IGMP and PIM commands were combined. If you revert the system firmware from E10 to an earlier release, be aware that the PIM configuration will be removed from the active configuration upon bootup. To prevent this, negate all **igmp** commands, reboot the router to the previous release, and replace **igmp** commands with **pim igmp** commands.

Configuring IGMP

To configure IGMP on the X-Pedition router, perform the following configuration tasks:

- Create IP interfaces.
- Set global parameters that will be used for all the interfaces on which multicast sources and receivers are attached.
- Configure IGMP on individual interfaces. You do so by enabling and disabling IGMP on interfaces and then setting IGMP parameters on the interfaces on which IGMP is enabled.
- Start the IGMP protocol.

IGMP on an IP Interface

By default IGMP is disabled on the X-Pedition router. To enable IGMP on an interface, enter the following command in Configure mode:

Enable IGMP on an interface.	**igmp enable interface** <*name/ipAddr*>

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

IGMP Query Interval

You can configure the X-Pedition router with a different IGMP Host Membership Query time interval. The interval you set applies to all ports on the router. The default query time interval is 125 seconds. To configure the IGMP host membership query time interval, enter the following command in Configure mode:

Configure the IGMP host membership query time interval.	**igmp set queryinterval** *<num>*

IGMP Response Wait Time

You can configure the X-Pedition router with a wait time for IGMP Host Membership responses which is different from the default. The wait time you set then applies to all ports on the X-Pedition router. The default response time is 10 seconds.

To configure the host response wait time, enter the following command in Configure mode:

Configure the IGMP host response wait time.	**igmp set responsetime** *<num>*

Static IGMP Groups

If IGMP is enabled on an interface, at least one group member needs to be present on the interface for the X-Pedition router to retain the group on its list of multicast group memberships for the interface. You can configure a static IGMP group for an interface; a static group can exist without any group members present on the interface. To configure a static IGMP group on an interface, enter the following command in Configure mode:

Configure a static IGMP group on an interface.	**igmp join group** *<ip-addr/subnet mask>* **interface** *<name/ip-addr>*

Note: The X-Pedition router displays interface names up to 32 characters in length.

L2 Snooping

IGMP L2 snooping allows an X-Pedition router functioning strictly as a layer-2 switch for a specific VLAN to actively participate in IGMP traffic forwarding. To enable IGMP snooping on a VLAN, use the following command:

igmp enable vlan <*vlan-name*>

To configure IGMP snooping parameters on a VLAN, use the following command:

igmp set vlan <*vlan-name*> [**filter-ports** <*port*>] [**host-timeout** <*number*>] [**leave-timeout** <*number*>] [**querier-timeout** <*number*>] [**router-timeout** <*number*>] [**permanent-ports** <*port*>]

Note: The **igmp start-snooping** command must be present for the **igmp enable vlan** command to function properly, and the **igmp start-snooping** command supports IGMP-enabled VLANs only—it is not intended for use with IGMP-enabled interfaces.

The X-Pedition router displays VLAN names up to 32 characters in length.

IGMP L2 snooping depends on the presence of an upstream IGMP querier. Whenever it receives an IGMP query, the X-Pedition router forwards the query out the appropriate VLAN ports. IGMP snooping allows per-port traffic patterns in VLANs with multiple ports.

Note: You may not use any L3 configuration on a VLAN that is snooping (e.g., no interfaces created).

DVMRP Overview

DVMRP is an IP multicast routing protocol. On the X-Pedition router, DVMRP routing is implemented as specified in the **draft-ietf-idmr-dvmrp-v3-09.txt** file, which is an Internet Engineering Task Force (IETF) document. The X-Pedition router's implementation of DVMRP supports the following:

- Generation identifiers, which are assigned to DVMRP whenever that protocol is started on a router.
- Pruning, which is an operation DVMRP routers perform to exclude interfaces not in the shortest path tree.

DVMRP uses the Reverse Path Multicasting (RPM) algorithm to perform pruning. In RPM, a source network rather than a host is paired with a multicast group. This is known as an (S,G) pair. RPM permits the X-Pedition router to maintain multiple (S,G) pairs.

On the X-Pedition router, DVMRP can be configured on a per-interface basis. An interface does not have to run both DVMRP and IGMP. You can start and stop DVMRP independently from other multicast routing protocols. The X-Pedition router supports up to 96 multicast interfaces.

Note: The X-Pedition router does not allow users to enable DVMRP and PIM simultaneously. The router is only allowed to run either PIM or DVMRP in an active configuration. If a user is running one of the multicast protocols and tries to activate the second protocol, an appropriate error message is displayed and the second protocol is not activated. However, the new configuration is entered into the configuration file without it being marked as being in error.

To support backward compatibility on DVMRP interfaces, you can configure the router expire time and prune time on each X-Pedition router DVMRP interface. This lets it work with older versions of DVMRP.

You can use threshold values and scopes to control internetwork traffic on each DVMRP interface. Threshold values determine whether traffic is either restricted or not restricted to a subnet, site, or region. Scopes define a set of multicast addresses of devices to which the X-Pedition router can send DVMRP data. Scopes can include only addresses of devices on a company's internal network and cannot include addresses that require the X-Pedition router to send DVMRP data on the Internet. The X-Pedition router also allows control of routing information exchange with peers through route filter rules.

You can also configure tunnels on X-Pedition DVMRP interfaces. A tunnel is used to send packets between routers separated by gateways that do not support multicast routing. A tunnel acts as a virtual network between two routers running DVMRP. A tunnel does not run IGMP. The X-Pedition router supports a maximum of eight tunnels.

Note: Tunnel traffic is not optimized on a per-port basis, and it goes to all ports on an interface, even though IGMP keeps per-port membership information. This is done to minimize CPU overload for tunneled traffic.

Configuring DVMRP

You configure DVMRP routing on the X-Pedition router by performing the following DVMRP-configuration tasks:

- Creating IP interfaces.
- Setting global parameters that will be used for all the interfaces on which DVMRP is enabled.
- Configuring DVMRP on individual interfaces. You do so by enabling and disabling DVMRP on interfaces and then setting DVMRP parameters on the interfaces on which DVMRP is disabled.
- Defining DVMRP tunnels, which IP uses to send multicast traffic between two end points.

Starting and Stopping DVMRP

DVMRP is disabled by default on the X-Pedition router.

To start or stop DVMRP, enter one of the following commands in Configure mode:

Start DVMRP.	**dvmrp start**
Stop DVMRP.	**no dvmrp start**

DVMRP on Individual Interfaces

DVMRP can controlled/configured on per interface basis. An interface does not have to run both DVMRP and IGMP together. Router to Router connections only require DVMRP to be enabled between the routers; IGMP does not have to be enabled on these links if there are no receivers. Receiver connection interfaces can either be IGMP only enabled or both IGMP and DVMRP enabled.

To enable DVMRP on an interface, enter the following commands in the configure mode:

Enable DVMRP on an interface.	**dvmrp enable interface** <*ipAddr*>|<*interface-name*>

Note: The X-Pedition router displays interface names up to 32 characters in length.

DVMRP Parameters

DVMRP neighbor timeout and prune time can be configured for all interfaces. The default neighbor timeout is 35 seconds. The default prune time is 7200 seconds (2 hours).

To configure neighbor timeout or prune time, enter one of the following commands in Configure mode:

Configure the DVMRP neighbor timeout.	**dvmrp set neighbor-timeout** <*seconds*>
Configure the DVMRP prune time.	**dvmrp set prunetime** <*seconds*>

DVMRP Routing Metric

You can configure the DVMRP routing metric associated with a set of destinations for DVMRP reports. The default metric is 1.

To configure the DVMRP routing metric, enter the following command in Configure mode:

Configure the DVMRP routing metric.	**dvmrp set interface** <*ipAddr/name*> **metric** <*num*>

DVMRP TTL & Scope

For control over internet traffic, per-interface control is allowed through Scopes and TTL thresholds.

The TTL value controls whether packets are forwarded from an interface. The following are conventional guidelines for assigning TTL values to a multicast application and their corresponding X-Pedition setting for DVMRP threshold:

TTL = 1	Threshold = 1	Application restricted to subnet
TTL < 16	Threshold = 16	Application restricted to a site
TTL < 64	Threshold = 64	Application restricted to a region
TTL < 128	Threshold = 128	Application restricted to a continent
TTL = 255		Application not restricted

To configure the TTL Threshold, enter the following command in Configure mode:

Configure the TTL Threshold.	**dvmrp set interface** <*ipAddr/name*> **threshold** <*number*>

TTL thresholding is not always considered useful. There is another approach of a range of multicast addresses for “administrative” scoping. In other words, such addresses would be usable within a certain administrative scope, a corporate network, for instance, but would not be forwarded across the internet. The range from 239.0.0.0 through 239.255.255.255 is being reserved for administratively scoped applications. Any organization can currently assign this range of addresses and the packets will not be sent out of the organization. In addition, multiple scopes can be defined on per-interface basis.

To prevent the X-Pedition router from forwarding any data destined to a scoped group on an interface, enter the following command in the Configure mode:

Configure the DVMRP scope.	**dvmrp set interface** <*ip-addr*> **scope** <*ip-addr/mask*>

DVMRP Tunnels

The **dvmrp create tunnel** command creates a tunnel used to pass multicast traffic through a unicast network that resides between DVMRP clouds. As multicast frames exit the DVMRP source cloud, they are encapsulated in a unicast packet. When frames enter the destination cloud, the unicast packets are un-encapsulated and returned to the native multicast format.

The X-Pedition control module encapsulates and un-encapsulates each packet, not in the hardware ASICs but via software (this can degrade overall performance and drop frames from the stream). Therefore, bandwidth is limited and CPU utilization can increase when sending multicast traffic across the tunnel. The amount of bandwidth and CPU utilization is dependant on many factors, such as: CPU-bound traffic such as learning new flows, ARPs, ACLs, RMON, packet size, traffic rate, and routing updates.

The tunnel is disabled by default. To enable it, uses the **dvmrp enable tunnel** command.

DVMRP tunnels need to be created before being enabled. Tunnels are recognized by the tunnel’s local IP address. Although any IP interface can be configured as DVMRP tunnel interface, the correct set of interfaces that should be enabled is the router to router endpoint connections, in which the traffic will be traversing through the tunnel. This will display the correct set of routes in the DVMRP route table. Also, the **dvmrp enable interface** command must be set to the DVMRP tunnel endpoint in order to establish a DVMRP neighbor relationship between endpoints.

Once a DVMRP tunnel is created, you can enable DVMRP on the interface. The X-Pedition router supports a maximum of eight tunnels.

To configure a DVMRP tunnel, enter the following command in Configure mode:

Configure a DVMRP tunnel.	**dvmrp create tunnel local** <*ip-addr*> **remote** <*ip-addr*>

Monitoring IGMP and DVMRP

You can monitor IGMP and DVMRP information on the X-Pedition router. To display IGMP and DVMRP information, enter the following commands from Enable mode.

Show grafts, interfaces, mfc, neighbors, prunes-rx, prunes-tx, and routes.	f. dvmrp show all
Show the DVMRP designated forwarder for the source address given.	**dvmrp show designated-forwarder source** <*ipaddr*>
Show DVMRP grafts that are pending.	**dvmrp show grafts**
Show interfaces running DVMRP. Also shows the neighbors on each interface.	**dvmrp show interface** <*ipaddr*> | **all**
Show DVMRP multicast forwarding cache.	**dvmrp show mfc**
Show DVMRP enabled neighbors.	**dvmrp show neighbors**
Show received prune requests.	**dvmrp show prunes-rx**
Show transmitted prune requests.	**dvmrp show prunes-tx**
Show routes learned via DVMRP.	**dvmrp show route** <*ipaddr*> | **all**
Show all IGMP group memberships.	**igmp show groups**
Show all the interfaces running IGMP.	**igmp show interfaces all**
Show IGMP VLANs. The **timers** keyword shows L2 snooping related timers.	**igmp show vlans** [**detail**] [**name** <*name*>] [**timers**]
Show information about multicasts registered by IGMP.	**l2-tables show igmp-mcast-registrations**
Show IGMP status on a VLAN.	**l2-tables show vlan-igmp-status vlan** <*VLAN-num*>
Show combined DVMRP and PIM forwarding cache.	**multicast show mfc**

Configuration Examples

The following is a sample X-Pedition router configuration for DVMRP and IGMP. Seven subnets are created. IGMP is enabled on four IP interfaces. The IGMP query interval is set to 30 seconds. DVMRP is enabled on five IP interfaces. IGMP is not running on “downstream” interfaces.

```
! Create VLANS.
!
vlan create upstream ip
vlan add ports et.5.3,et.5.4 to upstream
!
! Create IP intefaces
!
interface create ip mls15 address-netmask 172.1.1.10/24 port et.5.8
interface create ip company address-netmask 207.135.89.64/25 port et.5.1
interface create ip test address-netmask 10.135.89.10/25 port et.1.8
interface create ip rip address-netmask 190.1.0.1 port et.1.4
interface create ip mbone address-netmask 207.135.122.11/29 port et.1.1
interface create ip downstream address-netmask 10.40.1.10/24 vlan upstream
!
! Enable IGMP interfaces.
!
igmp enable interface 10.135.89.10
igmp enable interface 172.1.1.10
igmp enable interface 207.135.122.11
igmp enable interface 207.135.89.64
!
! Set IGMP Query Interval
!
igmp set queryinterval 30
!
! Enable DVMRP
!
dvmrp enable interface 10.135.89.10
dvmrp enable interface 172.1.1.10
dvmrp enable interface 207.135.122.11
dvmrp enable interface 207.135.89.64
dvmrp enable interface 10.40.1.10
!
! Set DVMRP parameters
!
dvmrp set interface 172.1.1.10
dvmrp set neighbor-timeout 200
!
! Start DVMRP
!
dvmrp start
!
! Start IGMP
!
igmp start
```

Protocol Independent Multicast (PIM-SM)

Introduction

Protocol Independent Multicast, *PIM*, is a multicast routing protocol used to dynamically build a distribution tree used to forward multicast data on a network. Although group members are at the "leaves" of the tree, the multicast data transmission source is generally at the root. PIM utilizes existing unicast routes to perform *reverse path forwarding* (RPF) checks—essentially a route lookup on the source—and the routing engine returns the best interface, regardless of how the routing table is constructed. In this sense, PIM is *independent* of any routing protocol—PIM can perform RPF checks using OSPF routes, RIP routes, static routes, or a combination of route types.

Note: See *Interoperability* on page 364 for restrictions associated with the PIM protocol.

The PIM specifications define several modes or methods by which a PIM router can build the distribution tree. The mode described in this document is *sparse mode* (also referred to as PIM-SM). Sparse Mode utilizes only those routers that need to be included in forwarding multicast data. Sparse mode routers use bandwidth more efficiently than other modes, but can require more processing time when working with large numbers of streams.

Features common to all modes include:

1. Using IGMP to propagate group membership information.
2. Sending "hello" messages to determine neighbor presence and configuration.
3. Sending "join/prune" messages to determine the need to retain multicast route information for a particular group on an interface.
4. Sending "assert" messages to resolve conflicts that occur regarding inbound interfaces.

PIM-SM message types

PIM-SM defines the following message types:

- **Hello messages** announce the sender's presence to other PIM-SM routers. The hello packet includes options such as hold time (the length of time to keep the sender reachable) and *designated router* (DR) priority (used to designate which PIM-SM router will act on behalf of sources and receivers in the PIM-SM domain).
- **Register messages (sparse only)** are used by a source's DR to encapsulate (i.e., register) multicast data and send it to the *rendezvous point* (RP)—a PIM sparse mode router designated as the root of a shared tree.
- **Register-Stop messages (sparse only)** are used by the RP to tell the source's DR to stop registering traffic for a particular source.

- **Join/Prune (J/P) messages** contain information on group membership received from downstream routers. J/P messages contain one or more group entries which, in turn, contain one or more *source entries* called *joins* and *prunes*. J/P messages allow a PIM-SM router to build or modify the distribution tree—join messages *maintain* existing links or *create* new ones as part of the tree; prune messages *remove* links from the tree. Routers always send J/P messages upstream, toward the root of the tree—the interface that receives the join is the *outbound interface* (OIF) and the interface that sends the join is the *inbound interface* (IIF). Because routers build the tree opposite the flow of traffic, this process is called *reverse-path* forwarding.

  Joins and prunes employ two important bits—the RPT (R) bit indicates that the join or prune applies to the shared tree and the WC (W) bit indicates that the join or prune applies to the group-only (*,G) forwarding state. If the router does not set the WC bit, the join or prune will apply to source—group pairs (S,G). If the router sets the W bit, it *must* also set the R bit.

  If a source entry contains the address for the group's RP instead of a *source*, the entry is a (*,G) join or prune and has its WC (W) and RPT (R) bits set. PIM-SM routers forward (*,G) joins and prunes toward the RP—not toward a source. Similarly, if the source entry contains the address of a multicast source that is sending packets to the group, the entry is an (S,G) join or prune. A PIM-SM router forwards (S,G) joins and prunes toward the source's DR. If a PIM-SM router sets the R bit in an (S,G) entry, the join or prune becomes an (S,G,RPT) join or prune and indicates an interest or a lack of interest in a particular source on the shared tree. Like (*,G) joins and prunes, a PIM-SM router forwards (S,G,RPT) joins and prunes toward the RP for the group.

- **Assert messages** indicate that the router received a data packet on its OIF for the group. Assert messages report the *metric* or distance to the source or RP to help the router identify the most direct path to the root of the tree. If multiple routers claim to have the most direct path to the source or RP, each router sends its own assert message—the router with the best metric wins. The other router removes that link from its OIF list for the group.

Sparse Mode

In contrast to dense multicast routing protocols like DVMRP and PIM Dense Mode which flood traffic across every link and prune back unnecessary paths (i.e., the *Flood and Prune* model), PIM Sparse Mode follows the *Explicit Join* model. Sparse-mode routers send multicast traffic to only those parts of the network that request it.

The Shared Tree

When a receiver subscribes to a group, the receiver's *Designated Router* (DR) sends a join toward the *Rendezvous Point* (RP)—a router designated as the root of a shared tree (the *RPT*) for the group. The join's *wildcard* (WC) bit is set, indicating that it contains no source-specific information, and its RP bit is set, indicating that its destination is the shared tree. A (*,G) entry (*star-jee* or *star-comma-jee*) in the multicast routing table indicates that no source information is available yet (*), but there is an interest in the group (G). The RPF check is performed on the RP instead of on a source. After receiving a (*,G) join, each router along the way to the RP retains a (*,G) entry. If a router's OIF list is empty (indicating that it has no interest in the group), the router sends a (*,G) prune (a prune with its WC bit set) toward the RP. If another router on a multi-access link sees the (*,G) prune and elects to continue receiving traffic, it may send a (*,G) join to override the prune.

When a source starts sending packets to the group, the source's DR encapsulates each packet in a *Register* message and sends them, unicast, directly to the RP. The RP de-encapsulates each Register message and sends the resulting packet, multicast, down the shared tree toward the listeners. This encapsulation mechanism allows the RP to act as the root of the tree for all sources sending packets to the group and conserves bandwidth by maintaining one *RP*-rooted tree instead of multiple *source*-rooted trees. After it sends the multicast data packet, the RP will send an (S,G) join (a join sent without the W and R bits set) toward the source.

Note: The RP may send the join immediately, or after the data rate exceeds a configured threshold. This allows the administrator to control how PIM-SM uses network resources.

Once the source's DR receives the (S,G) join, the DR becomes the root of a shortest-path tree (SPT) or a source-rooted tree and creates an OIF. With the OIF created, the DR begins sending native multicast packets (non-registered packets) down its SPT and registered packets to the RP.

Once the RP's IIF receives native multicast packets from the SPT, the RP sends a *Register Stop* message to the source's DR to indicate that the DR no longer needs to encapsulate messages. Although it joined an SPT, the RP continues to forward traffic down the shared tree—this allows the RP and DR to conserve processor time; however, this increases resource usage on the routers between them because these routers now maintain an (S,G) state that was unnecessary during encapsulation. The RP also sends a Register Stop message if no OIF list exists for the group—if there is no shared tree to distribute the traffic, there is no need to encapsulate the traffic.

The Shortest Path Tree

When a receiver's DR wants to receive traffic directly from the source, the DR sends (S,G) join messages upstream toward the source, not the *RP.* This will create an SPT. (S,G) prune messages sent in the same manner indicate that the leaf router has no more active listeners for the source. In PIM-SM's *source-specific mode (SSM)*, joins and prunes are always (S,G)—since there is no RP or shared tree, routers always send joins toward the source. Multicast IP addresses in the range *232.x.x.x* are considered SSM addresses, so any routers that receive memberships for a group address in this range will always generate an SPT for that group.

Transitioning from RPT to SPT

If the rate of multicast traffic from the source exceeds the bandwidth specified by a threshold on a receiver's DR, the DR will send an (S,G) join toward the source (S). The listener, now on the SPT for S, can receive traffic directly from S. Additionally, if the IIF for the (S,G) differs from the IIF for the (*,G), the router sends an *(S,G,RPT)* prune (along with any (*,G) joins it sends) toward the RP. An (S,G,RPT) prune indicates to the upstream RPT router that this router does not wish to receive traffic sent to the group on the RPT—this restriction applies only to traffic from this source. Similar to an (S,G) prune, an (S,G,RPT) prune with its R bit *set* indicates that the action applies to the shared tree.

RP Selection

To forward multicast traffic in Sparse Mode, a router must determine which router is the RP or *rendezvous point* (a PIM-SM router designated as the root of a shared tree) for a particular group. An administrator may configure a PIM-SM router as a *candidate RP (C-RP)*, set of groups, or all groups. The C-RP sends advertisements to the *Bootstrap Router (BSR)* which, in turn, advertises the *candidate RP-set* to every router in the domain. Once the PIM-SM routers receive the BSR's message, the routers use a common hashing algorithm to hash the C-RP address, group, and mask together to identify which router will be the RP for a given group.

A C-RP router must also learn which router is the BSR. A PIM-SM router can also be configured as a *candidate-BSR (C-BSR)*. Each C-BSR asserts itself as the BSR, then defers once it receives a preferable BSR message. Eventually, all C-RPs will send their messages to a single BSR which will communicate the Candidate RP-set to all PIM-SM routers in the domain.

System Overview

System Architecture

As a result of including PIM-SM support in the current routing base, multicast is now supported by two daemons—*gated* and *mrouted*. This requires routers to provide IGMP traffic to both daemons, directing IGMP traffic to socket listeners within each environment. Both DVMRP and PIM-SM require the underlying IGMP protocol as a base.

Configuration & Limitations

Routing Tables

By definition, PIM-SM does not operate a routing protocol. PIM-SM uses active routes in the unicast routing table to determine the IIF; however, PIM-SM may not use those routes until they are configured for use by multicast routing protocols. For this reason, *gated* now maintains two *routing information bases* or *RIBs*—one unicast and one multicast. The unicast RIB is the same information base maintained and updated by unicast routing protocols. The multicast RIB is a second information base used by multicast routing protocols to determine accessibility and a route back to the source (RPF).

The X-Pedition router distributes routes into the multicast RIB through configuration (except in the case of DVMRP, which enters routes directly into the multicast RIB). Routes connected directly, (or local routes) distribute into the multicast RIB automatically. Static routes, aggregate routes, and OSPF now include directives to install routes in the multicast RIB and the unicast RIB. Other protocols will use the **ip-router policy** commands to redistribute routes into the multicast RIB.

Interfaces

PIM-SM supports all WAN interface types. The maximum number of multicast-configured interfaces supported is 96. Size is limited because:

- Every *multicast-forwarding cache* (or *MFC*) control structure passed between the IP stack and various processes is limited in size to fit inside an *mbuf* structure, an essential component of inter-process communication.
- The MFC control structure contains other information related to multicast routing.
- Each control structure contains an array the size of the maximum number of *virtual interfaces* (or *vifs*).
- A vif is created for every multicast-enabled interface on the router.
- A vif is used for replicating a multicast packet to each outbound interface. Therefore, the maximum number of interfaces is the same as the maximum number of vifs.

PIM-SM domains must be *convex*. That is, along any shortest path between two PIM-SM routers, all links must be configured as PIM-SM interfaces-this ensures that all routers listening to Joins and Prunes sent along the path to the RP or source will understand these messages.

Flows

The limit to the number of flows on a router depends only on system memory. Since each (S,G) or (*,G) pair requires a great deal of memory, (S,G) forwarding flows will often exhaust physical memory before line-card table memory.

The mode of Layer-3 flow setup in hardware is restricted. Since the set of outbound ports depends upon the group *and* the source addresses, destination-based flow mode will not work properly in most configurations.

Reconfiguration

The X-Pedition router does not support the use of GenIDs, a mechanism used to let neighboring routers know that the router rebooted. Upon receiving a new GenID, a router should retransmit certain information related to its current state and RP-set.

Static RPs

Using static RPs can simplify the convergence of a PIM-SM domain by eliminating the need for BSR support; however, static RPs are difficult to maintain in a large domains and are similar to static routes in advantages and limitations.

Examples

Every PIM-SM enabled network needs to know which routers are selected as the RP and RP candidates for the network. This information is distributed through configuration (static RPs) or via a *Bootstrap Router* (BSR) mechanism. PIM-SM uses the underlying Unicast routing protocol to determine the best path back to the RP or to the source.

Configuration

Before configuring PIM-SM on your network, you will need to create a *sparse* component for each router on which PIM-SM will be enabled. The sparse component allows the router to combine all PIM-SM configuration commands and place them together at a single location. To create a sparse component, use the **pim sparse create component <*name*>** command. With the sparse component created, you may add interfaces to the component to indicate to the router the interfaces on which PIM-SM will be enabled. To add interfaces to the sparse component, use the **pim sparse add interface <*name*> to-component <*component*>** command where **<*component*>** is the name used in the **create component** command.

If the router on which you configure PIM-SM will act as a candidate RP or candidate BSR, you must enable this capability with the **pim sparse set component <*component*> crp-on** or **pim sparse set component <*component*> bsr-on** command. Once the configuration is complete, enter the **pim global start** command to begin running the protocol.

Note: Clients connected directly to the router must run IGMP in order to request multicast traffic.

To configure IGMP on a specific interface(s), use the **igmp enable interface <*name*>** command. After configuring the interface(s), enter the **igmp start** command from the CLI to launch IGMP.

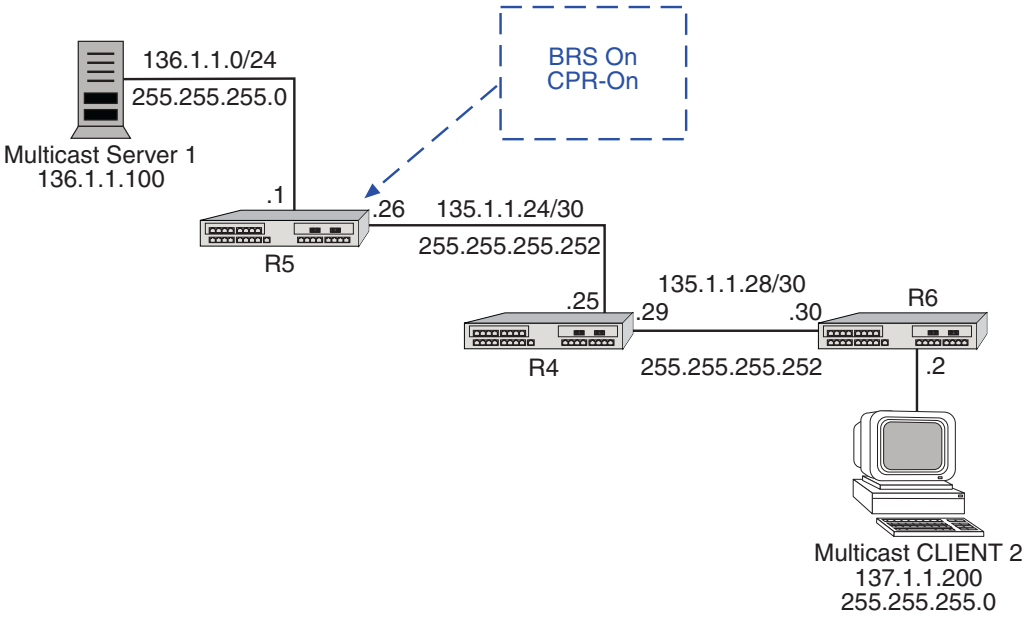
Note: When using the **igmp start** command, failing to specify individual interfaces with the **igmp enable interface** command will cause the router to enable IGMP on all IP interfaces.

Example 1

The following example demonstrates how to configure three routers that use different routing protocols (RIPv2, OSPF, and BGP) as the underlying method to provide route information for PIM-SM. One of the servers is attached directly to the RP (which is also configured as the BSR in the network).

Note: To perform RPF checks to the IP address of the server, you must enable PIM-SM on the server's network.

Figure 35. Using RIPv2, OSPF, and BGP to Provide Route Information for PIM-SM



Static Routes

Static route information is made available to PIM-SM through the "**unicast-rib**" and "**multicast-rib**" keywords. By default, a static route is available only to the unicast RIB; however, if you use the **multicast-rib** keyword, the static route is used only in the multicast RIB. To use the static route for both unicast and multicast route lookups, use both keywords as depicted in the following example:

```
xp(config)# ip add route default gateway 135.1.1.1 unicast-rib multicast-rib
```

The following is taken from the R6 configuration file and depicts multiple keyword use:

```
interface create ip toR4 address-netmask 135.1.1.30/30 port et.5.8
interface create ip i137 address-netmask 137.1.1.2/24 port et.5.7
interface add ip lo0 address-netmask 6.6.6.6/32
ip add route default gateway 135.1.1.29 unicast-rib multicast-rib
igmp enable interface i137
igmp start
pim sparse create component R6
pim sparse add interface toR4 to-component R6
pim global start
system set name "R6"
```

The output below depicts the RIB as seen from R6:

```
R6# ip-router show rib
Routing Tables:
Destinations: 6    Routes: 6
Holddown: 0    Delete: 6    Hidden: 0
Codes: Network - Destination Network Address
     U - Unicast Status   + = Best Route, - = Last Active, * = Both
     M - Multicast Status + = Best Route, - = Last Active, * = Both
     Src - Source of the route :
     Ag - Aggregate, B - BGP derived, C - Connected
     R - RIP derived, St - Static, O - OSPF derived
     OE - OSPF ASE derived, D - Default
     Next hop - Gateway for the route ; Next hops in use: 4
     Netif - Next hop interface
     Prf1 - Preference of the route, Prf2 - Second Preference of the route
     Mtrc1 - Metric1 of the route, Mtrc2 - Metric2 of the route
     Age - Age of the route since last refreshed
Network/Mask  U M   Src  Next hop       Netif      Prf1  Mtrc1  Mtrc2        Age
------------------  -  -   ---  ---------------  ----------  ----  -----  -----  --------
0.0.0.0/0     *  *   St   135.1.1.29    toR4       60    0      0           10
6.6.6.6/32    *  *   C    6.6.6.6       lo         0     1      0           14:37
127/8         *      St   127.0.0.1     lo         0     0      0           25:02:27
127.0.0.1/32  *  *   C    127.0.0.1     lo         0     1      0           25:02:27
135.1.1.28/30 *  *   C    135.1.1.30    toR4       0     1      0           14:14
137.1.1/24    *  *   C    137.1.1.2     i137       0     1      0           12:49
```

Note: The networks above that are denoted with an asterisk (*) in the **U** or **M** column are active in the unicast and multicast RIBs.

RIPv2

RIP routes must be imported into the multicast RIB using the **ip-router policy**. To create this policy, use the **ip-router policy create rip-import-source <*name*> unicast multicast** command. To apply the policy, use the **ip-router policy import source <*name*> network all** command. The following examples depict the steps taken to perform this action on each router:

Note: If you run the **ip-router policy import source <*name*> network all** command after RIP is already running, the router will remove the unicast RIP routes from the RIB. In order for the network to learn these changes, you must turn off RIP, create and apply the import source, then re-enable RIP on the network. This is not necessary for new interfaces added to RIP after the policy is created.

R4:

```
interface create ip toR5 address-netmask 135.1.1.25/30 port et.2.5
interface create ip toR6 address-netmask 135.1.1.29/30 port et.2.6
interface add ip lo0 address-netmask 4.4.4.4/32
ip-router policy create rip-import-source RIP_multicast_routes unicast multicast
ip-router policy import source RIP_multicast_routes network all
rip add interface toR5
rip add interface toR6
rip set interface toR5 version 2
rip set interface toR6 version 2
rip start
pim sparse create component R4
pim sparse add interface toR6 to-component R4
pim sparse add interface toR5 to-component R4
pim global start
system set name “R4”
```

R5:

```
interface create ip iserver address-netmask 136.1.1.1/24 port et.2.5
interface create ip toR4 address-netmask 135.1.1.26/30 port et.2.4
interface add ip lo0 address-netmask 5.5.5.5/32
ip-router policy create rip-import-source RIPv2_import_for_multicast unicast multicast
ip-router policy import source RIPv2_import_for_multicast network all
rip add interface toR4
rip set interface toR4 version 2
rip start
igmp enable interface iserver
igmp start
pim sparse create component R5
pim sparse set component R5 crp-on bsr-on
pim sparse add interface toR4 to-component R5
pim sparse add interface iserver to-component R5
pim global start
system set name “R5”
```

R6:

```
interface create ip toR4 address-netmask 135.1.1.30/30 port et.5.8
interface create ip i137 address-netmask 137.1.1.2/24 port et.5.7
interface add ip lo0 address-netmask 6.6.6.6/32
ip-router policy create rip-import-source RIP_v2_to_multi unicast multicast
ip-router policy import source RIP_v2_to_multi network all
rip add interface toR4
rip set interface toR4 version 2
rip start
igmp enable interface i137
igmp start
pim sparse create component R6
pim sparse add interface toR4 to-component R6
pim global start
system set name “R6”
```

The example below displays the RIB as seen from R4:

```
R4# ip-router show rib
Routing Tables:
Destinations: 9    Routes: 9
Holddown: 0    Delete: 13    Hidden: 0
Codes: Network - Destination Network Address
     U - Unicast Status   + = Best Route, - = Last Active, * = Both
     M - Multicast Status + = Best Route, - = Last Active, * = Both
     Src - Source of the route :
     Ag - Aggregate, B - BGP derived, C - Connected
     R - RIP derived, St - Static, O - OSPF derived
     OE - OSPF ASE derived, D - Default
     Next hop - Gateway for the route ; Next hops in use: 4
     Netif - Next hop interface
     Prf1 - Preference of the route, Prf2 - Second Preference of the route
     Mtrc1 - Metric1 of the route, Mtrc2 - Metric2 of the route
     Age - Age of the route since last refreshed

Network/Mask      U   M   Src  Next hop        Netif      Prf1  Mtrc1  Mtrc2       Age
------------------ -   -   ---  ---------------  ----------  ----  -----  -----  --------
4.4.4.4/32        *   *   C    4.4.4.4         lo         0     1      0      24:33:40
5.5.5.5/32        *   *   RIP  135.1.1.26       toR5       100   2      0      6
6.6.6.6/32        *   *   RIP  135.1.1.30       toR6       100   2      0      4
127/8             *       St   127.0.0.1       lo         0     0      0      25:40:13
127.0.0.1/32      *   *   C    127.0.0.1       lo         0     1      0      25:40:13
135.1.1.24/30     *   *   C    135.1.1.25      toR5       0     1      0      24:23:32
135.1.1.28/30     *   *   C    135.1.1.29      toR6       0     1      0      1:54:28
136.1.1/24        *   *   RIP  135.1.1.26       toR5       100   2      0      6
137.1.1/24        *   *   RIP  135.1.1.30       toR6       100   2      0      4
```

Note: The networks above that are denoted with an asterisk (*) in the **U** or **M** column are active in the unicast and multicast RIBs.

OSPF

In order to get OSPF to make its routes available for PIM-SM to use, you must enter the **ospf set ase-defaults multicast** command in the configuration. The following examples are taken from the configuration files for the routers used in Figure 35 on page 339.

R4:

```
interface create ip toR5 address-netmask 135.1.1.25/30 port et.2.5
interface create ip toR6 address-netmask 135.1.1.29/30 port et.2.6
interface add ip lo0 address-netmask 4.4.4.4/32
ospf create area 0.0.0.255
ospf add interface toR5 to-area 0.0.0.255
ospf add stub-host 4.4.4.4 to-area 0.0.0.255 cost 5
ospf add interface toR6 to-area 0.0.0.255
ospf set ase-defaults multicast
ospf start
pim sparse create component R4
pim sparse add interface toR6 to-component R4
pim sparse add interface toR5 to-component R4
pim global start
system set name “R4”
```

R5:

```
interface create ip iserver address-netmask 136.1.1.1/24 port et.2.5
interface create ip toR4 address-netmask 135.1.1.26/30 port et.2.4
interface add ip lo0 address-netmask 5.5.5.5/32
ospf create area 0.0.0.255
ospf add stub-host 5.5.5.5 to-area 0.0.0.255 cost 5
ospf add interface iserver to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf set ase-defaults multicast
ospf start
igmp enable interface iserver
igmp start
pim sparse create component R5
pim sparse set component R5 crp-on bsr-on
pim sparse add interface toR4 to-component R5
pim sparse add interface iserver to-component R5
pim global start
system set name “R5”
```

R6:

```
interface create ip toR4 address-netmask 135.1.1.30/30 port et.5.8
interface create ip i137 address-netmask 137.1.1.2/24 port et.5.7
interface add ip lo0 address-netmask 6.6.6.6/32
ip-router policy redistribute from-proto static to-proto ospf
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add stub-host 6.6.6.6 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i137
igmp start
pim sparse create component R6
pim sparse add interface toR4 to-component R6
pim global start
system set name “R6”
```

The RIB as seen from R4:

```
R4# ip-router show rib
Routing Tables:
Destinations: 9    Routes: 9
Holddown: 0    Delete: 5    Hidden: 3
Codes: Network - Destination Network Address
     U - Unicast Status   + = Best Route, - = Last Active, * = Both
     M - Multicast Status + = Best Route, - = Last Active, * = Both
     Src - Source of the route :
     Ag - Aggregate, B - BGP derived, C - Connected
     R - RIP derived, St - Static, O - OSPF derived
     OE - OSPF ASE derived, D - Default
     Next hop - Gateway for the route ; Next hops in use: 4
     Netif - Next hop interface
     Prf1 - Preference of the route, Prf2 - Second Preference of the route
     Mtrc1 - Metric1 of the route, Mtrc2 - Metric2 of the route
     Age - Age of the route since last refreshed
```

Network/Mask	U	M	Src	Next hop	Netif	Prf1	Mtrc1	Mtrc2	Age
4.4.4.4/32	*	*	C	4.4.4.4	lo	0	1	0	23:48:33
4.4.4.4/32			O	4.4.4.4	lo	-10	5		23:48:28
5.5.5.5/32	*	*	O	135.1.1.26	toR5	10	25		23:37:28
6.6.6.6/32	*	*	O	135.1.1.30	toR6	10	25		6:33
127/8	*		St	127.0.0.1	lo	0	0	0	24:55:06
127.0.0.1/32	*	*	C	127.0.0.1	lo	0	1	0	24:55:06
135.1.1.24/30	*	*	C	135.1.1.25	toR5	0	1	0	23:38:25
135.1.1.24/30			O	135.1.1.25	toR5	-10	20		23:38:18
135.1.1.28/30	*	*	C	135.1.1.29	toR6	0	1	0	1:09:21
135.1.1.28/30			O	135.1.1.29	toR6	-10	20		1:09:13
136.1.1/24	*	*	O	135.1.1.26	toR5	10	40		5:18
137.1.1/24	*	*	OE	135.1.1.30	toR6	150	1	20	5:18

Note: The networks above that are denoted with an asterisk (*) in the **U** or **M** column are active in the unicast and multicast RIBs.

BGP

To configure BGP to populate the multicast RIB, you must create an import policy for the peer AS. This is similar to RIP, but you must associate an AS number with the import source. To create the import source policy, use the **ip-router policy create bgp-import-source <*name*> unicast multicast autonomous-system <*num*>** command, then apply the policy with the **ip-router policy import source <*name*> network all** command.

The following examples show R4 in AS3, and R5 in AS2. R4 will redistribute the routes from AS2 into the RIP network for R6.

R4:

```
interface create ip toR5 address-netmask 135.1.1.25/30 port et.2.5
interface create ip toR6 address-netmask 135.1.1.29/30 port et.2.6
interface add ip lo0 address-netmask 4.4.4.4/32
ip-router global set router-id 4.4.4.4
ip-router global set autonomous-system 3
ip-router policy create rip-import-source RIP_multicast_routes unicast multicast
ip-router policy create bgp-import-source from_AS2_to_multi unicast multicast autonomous-system 2
ip-router policy redistribute from-proto static to-proto rip
ip-router policy redistribute from-proto direct to-proto rip
ip-router policy redistribute from-proto bgp to-proto rip source-as 2
ip-router policy redistribute from-proto direct to-proto bgp target-as 2
ip-router policy redistribute from-proto static to-proto bgp target-as 2
ip-router policy redistribute from-proto rip to-proto bgp target-as 2
ip-router policy import source RIP_multicast_routes network all
ip-router policy import source from_AS2_to_multi network all
rip add interface toR6
rip set interface toR6 version 2
rip start
pim sparse create component R4
pim sparse add interface toR6 to-component R4
pim sparse add interface toR5 to-component R4
pim global start
bgp create peer-group toAS2 autonomous-system 2 type external
bgp add peer-host 135.1.1.26 group toAS2
bgp start
system set name “R4”
```

R5:

```
interface create ip iserver address-netmask 136.1.1.1/24 port et.2.5
interface create ip toR4 address-netmask 135.1.1.26/30 port et.2.4
interface add ip lo0 address-netmask 5.5.5.5/32
ip-router global set autonomous-system 2
ip-router global set router-id 5.5.5.5
ip-router policy create bgp-import-source from_AS3_to_multi unicast multicast autonomous-system 3
ip-router policy import source from_AS3_to_multi network all
igmp enable interface iserver
igmp start
pim sparse create component R5
pim sparse set component R5 crp-on bsr-on
pim sparse add interface toR4 to-component R5
pim sparse add interface iserver to-component R5
pim global start
bgp create peer-group toAS3 autonomous-system 3 type external
bgp add peer-host 135.1.1.25 group toAS3
bgp start
system set name “R5”
```

R6:

```
interface create ip toR4 address-netmask 135.1.1.30/30 port et.5.8
interface create ip i137 address-netmask 137.1.1.2/24 port et.5.7
interface add ip lo0 address-netmask 6.6.6.6/32
ip-router policy create rip-import-source RIP_v2_to_multi unicast multicast
ip-router policy import source RIP_v2_to_multi network all
rip add interface toR4
rip add interface i137
rip set interface toR4 version 2
rip set interface i137 version 2
rip start
igmp enable interface i137
igmp start
pim sparse create component R6
pim sparse add interface toR4 to-component R6
pim global start
system set name “R6”
```

The RIB as seen from R4:

```
R4# ip-router show rib
Routing Tables:
Destinations: 9    Routes: 9
Holddown: 0    Delete: 15    Hidden: 0
Codes: Network - Destination Network Address
     U - Unicast Status   + = Best Route, - = Last Active, * = Both
     M - Multicast Status + = Best Route, - = Last Active, * = Both
     Src - Source of the route :
     Ag - Aggregate, B - BGP derived, C - Connected
     R - RIP derived, St - Static, O - OSPF derived
     OE - OSPF ASE derived, D - Default
     Next hop - Gateway for the route ; Next hops in use: 4
     Netif - Next hop interface
     Prf1 - Preference of the route, Prf2 - Second Preference of the route
     Mtrc1 - Metric1 of the route, Mtrc2 - Metric2 of the route
     Age - Age of the route since last refreshed
```

Network/Mask	U	M	Src	Next hop	Netif	Prf1	Mtrc1	Mtrc2	Age
4.4.4.4/32	*	*	C	4.4.4.4	lo	0	1	0	25:44:44
5.5.5.5/32	*	*	BGP	135.1.1.26	toR5	170			21:14
6.6.6.6/32	*	*	RIP	135.1.1.30	toR6	100	2	0	7
127/8	*		St	127.0.0.1	lo	0	0	0	26:51:17
127.0.0.1/32	*	*	C	127.0.0.1	lo	0	1	0	26:51:17
135.1.1.24/30	*	*	C	135.1.1.25	toR5	0	1	0	25:34:36
135.1.1.24/30			BGP	135.1.1.26	toR5	170			21:14
135.1.1.28/30	*	*	C	135.1.1.29	toR6	0	1	0	3:05:32
136.1.1/24	*	*	BGP	135.1.1.26	toR5	170			21:14
137.1.1/24	*	*	RIP	135.1.1.30	toR6	100	2	0	7

Note: The networks above that are denoted with an asterisk (*) in the **U** or **M** column are active in the unicast and multicast RIBs.

The RIB as seen from R5:

R5# **ip-router show rib**
Routing Tables:
Destinations: 9 Routes: 9
Holddown: 0 Delete: 19 Hidden: 0
Codes: Network - Destination Network Address
U - Unicast Status + = Best Route, - = Last Active, * = Both
M - Multicast Status + = Best Route, - = Last Active, * = Both
Src - Source of the route :
Ag - Aggregate, B - BGP derived, C - Connected
R - RIP derived, St - Static, O - OSPF derived
OE - OSPF ASE derived, D - Default
Next hop - Gateway for the route ; Next hops in use: 4
Netif - Next hop interface
Prf1 - Preference of the route, Prf2 - Second Preference of the route
Mtrc1 - Metric1 of the route, Mtrc2 - Metric2 of the route
Age - Age of the route since last refreshed

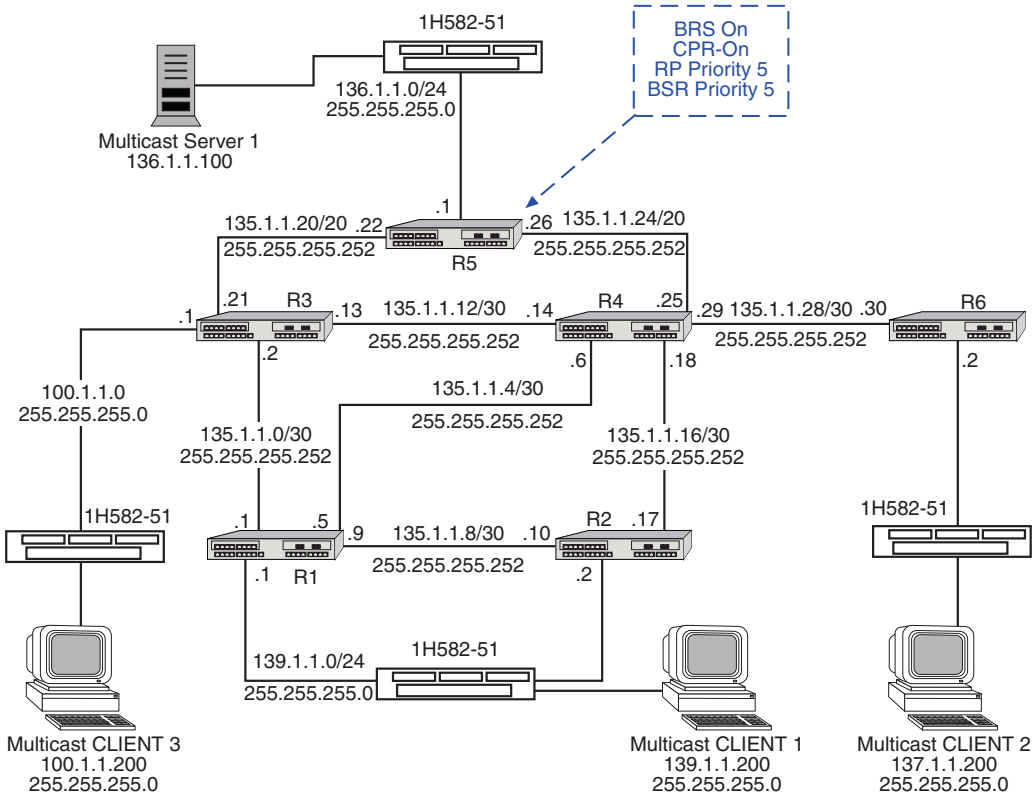
Network/Mask	U	M	Src	Next hop	Netif	Prf1	Mtrc1	Mtrc2	Age
4.4.4.4/32	*	*	BGP	135.1.1.25	toR4	170			21:38
5.5.5.5/32	*	*	C	5.5.5.5	lo	0	1	0	26:45:32
6.6.6.6/32	*	*	BGP	135.1.1.25	toR4	170			16:39
127/8	*		St	127.0.0.1	lo	0	0	0	26:51:55
127.0.0.1/32	*	*	C	127.0.0.1	lo	0	1	0	26:51:55
135.1.1.24/30	*	*	C	135.1.1.26	toR4	0	1	0	25:35:11
135.1.1.24/30			BGP	135.1.1.25	toR4	170			21:38
135.1.1.28/30	*	*	BGP	135.1.1.25	toR4	170			21:38
136.1.1/24	*	*	C	136.1.1.1	iserver	0	1	0	2:02:06
137.1.1/24	*	*	BGP	135.1.1.25	toR4	170			16:39

Note: The networks above that are denoted with an asterisk (*) in the **U** or **M** column are active in the unicast and multicast RIBs.

Example 2

This example demonstrates some options for configuring the PIM DR, BSR address, CRP address, and static RPs. The configuration for the routers in the following diagram are listed below.

Figure 36. Configuring the PIM DR, BSR Address, CRP Address, and Static RPs



R1 configuration:

```
interface create ip toR3 address-netmask 135.1.1.1/30 port et.1.3
interface create ip toR4 address-netmask 135.1.1.5/30 port et.1.4
interface create ip toR2 address-netmask 135.1.1.9/30 port et.1.2
interface create ip i139 address-netmask 139.1.1.1/24 port et.4.8
interface add ip lo0 address-netmask 1.1.1.1/32
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR3 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR2 to-area 0.0.0.255
ospf add stub-host 1.1.1.1 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i139
igmp start
pim sparse create component R1
pim sparse add interface toR3 to-component R1
pim sparse add interface toR4 to-component R1
pim sparse add interface toR2 to-component R1
pim sparse add interface i139 to-component R1
pim global start
system set name “R1”
```

R2 configuration:

```
interface create ip toR4 address-netmask 135.1.1.17/30 port et.4.4
interface create ip toR1 address-netmask 135.1.1.10/30 port et.4.1
interface create ip i139 address-netmask 139.1.1.2/24 port et.4.8
interface add ip lo0 address-netmask 2.2.2.2
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR1 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add stub-host 2.2.2.2 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i139
igmp start
pim sparse create component R2
pim sparse add interface toR1 to-component R2
pim sparse add interface toR4 to-component R2
pim sparse add interface i139 to-component R2
pim global start
system set name “R2”
```

R3 configuration:

```
interface create ip toR5 address-netmask 135.1.1.21/30 port et.2.5
interface create ip toR4 address-netmask 135.1.1.13/30 port et.2.4
interface create ip toR1 address-netmask 135.1.1.2/30 port et.2.1
interface create ip musers address-netmask 100.1.1.1/24 port et.2.8
interface add ip lo0 address-netmask 3.3.3.3/32
ip-router global set router-id 3.3.3.3
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR5 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR1 to-area 0.0.0.255
ospf add stub-host 3.3.3.3 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface musers
igmp start
pim sparse create component R3
pim sparse add interface toR5 to-component R3
pim sparse add interface toR4 to-component R3
pim sparse add interface toR1 to-component R3
pim global start
system set name “R3”
```

R4 configuration:

```
interface create ip toR5 address-netmask 135.1.1.25/30 port et.2.5
interface create ip toR6 address-netmask 135.1.1.29/30 port et.2.6
interface create ip toR2 address-netmask 135.1.1.18/30 port et.2.2
interface create ip toR3 address-netmask 135.1.1.14/30 port et.2.3
interface create ip toR1 address-netmask 135.1.1.6/30 port et.2.1
interface add ip lo0 address-netmask 4.4.4.4/32
ip-router global set router-id 4.4.4.4
ospf create area 0.0.0.255
ospf add interface toR5 to-area 0.0.0.255
ospf add stub-host 4.4.4.4 to-area 0.0.0.255 cost 5
ospf add interface toR6 to-area 0.0.0.255
ospf add interface toR1 to-area 0.0.0.255
ospf add interface toR2 to-area 0.0.0.255
ospf add interface toR3 to-area 0.0.0.255
ospf set ase-defaults multicast
ospf start
pim sparse create component R4
pim sparse add interface toR6 to-component R4
pim sparse add interface toR5 to-component R4
pim sparse add interface toR1 to-component R4
pim sparse add interface toR2 to-component R4
pim sparse add interface toR3 to-component R4
pim global start
system set name “R4”
```

R5 configuration:

```
interface create ip iserver address-netmask 136.1.1.1/24 port et.2.5
interface create ip toR4 address-netmask 135.1.1.26/30 port et.2.4
interface create ip toR3 address-netmask 135.1.1.22/30 port et.2.3
interface add ip lo0 address-netmask 5.5.5.5/32
ip-router global set router-id 5.5.5.5
ospf create area 0.0.0.255
ospf add stub-host 5.5.5.5 to-area 0.0.0.255 cost 5
ospf add interface iserver to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR3 to-area 0.0.0.255
ospf set ase-defaults multicast
ospf start
igmp enable interface iserver
igmp start
pim sparse create component R5
pim sparse set component R5 crp-on bsr-on
pim sparse set component R5 bsr-priority 5
pim sparse add interface toR4 to-component R5
pim sparse add interface iserver to-component R5
pim sparse add interface toR3 to-component R5
pim global start
system set name “R5”
```

R6 configuration:

```
interface create ip toR4 address-netmask 135.1.1.30/30 port et.5.8
interface create ip i137 address-netmask 137.1.1.2/24 port et.5.7
interface add ip lo0 address-netmask 6.6.6.6/32
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add stub-host 6.6.6.6 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i137
igmp start
pim sparse create component R6
pim sparse add interface toR4 to-component R6
pim global start
system set name “R6”
```

PIM DR

The X-Pedition router supports the PIM Hello packet priority option which, by default, carries an advertised priority of 1. If all PIM-SM routers sharing a subnet use the priority option in the Hello packet, the router with the highest priority will win the DR election. In the case of a tie, the router with the highest IP address becomes the DR. To set the hello priority for an interface, use the **pim sparse set interface <*name*> hello-priority <*num*>** command.

The configuration for R1 reads:

```
interface create ip toR3 address-netmask 135.1.1.1/30 port et.1.3
interface create ip toR4 address-netmask 135.1.1.5/30 port et.1.4
interface create ip toR2 address-netmask 135.1.1.9/30 port et.1.2
interface create ip i139 address-netmask 139.1.1.1/24 port et.4.8
interface add ip lo0 address-netmask 1.1.1.1/32
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR3 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR2 to-area 0.0.0.255
ospf add stub-host 1.1.1.1 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i139
igmp start
pim sparse create component R1
pim sparse add interface toR3 to-component R1
pim sparse add interface toR4 to-component R1
pim sparse add interface toR2 to-component R1
pim sparse add interface i139 to-component R1
pim global start
system set name “R1”
```

Using the configuration listed for R1 above, the neighbor table shows:

```
R1# pim show neighbor all

PIM Neighbors:

Neighbor Address : 135.1.1.2,  Interface : toR3
   Component : R1, Uptime : 52:26s, Expires : 1:43s soon
   Hello Priority : 1  (DR)

Neighbor Address : 135.1.1.1,  Interface : toR3
   Component : R1, Uptime : 52:27s, Expires : never
   Hello Priority : 1

Neighbor Address : 135.1.1.6,  Interface : toR4
   Component : R1, Uptime : 52:26s, Expires : 1:44s soon
   Hello Priority : 1  (DR)

Neighbor Address : 135.1.1.5,  Interface : toR4
   Component : R1, Uptime : 52:27s, Expires : never
   Hello Priority : 1

Neighbor Address : 135.1.1.10,  Interface : toR2
   Component : R1, Uptime : 51:45s, Expires : 1:30s soon
   Hello Priority : 1  (DR)

Neighbor Address : 135.1.1.9,  Interface : toR2
   Component : R1, Uptime : 52:27s, Expires : never
   Hello Priority : 1
```

The output displayed in the previous screen indicates that R1 is *not* the DR for any of the subnets listed. In the following configuration, the priority is increased for R1 to make it the DR on the 135.1.1.0/30 network (the network between R1 and R3).

```
R1# configure
R1(config)# pim sparse set interface toR3 hello-priority 10
R1(config)# save startup
Are you sure you want to overwrite the Startup configuration [no]? y

There are non-committed configuration changes. Do you want to make
these changes active and then save everything to Startup [yes]? y
2002-11-22 10:51:18 %CONFIG-I-SAVED, Console user (root), configuration saved to Startup configuration.
R1(config)# 2002-11-22 10:51:18 %GATED-I-RECONFIGDONE, Routing configuration changes completed (pid
0x80b5abe8).
Exit

R1# pim show neighbor all

PIM Neighbors:

Neighbor Address : 135.1.1.1,  Interface : toR3
   Component : R1, Uptime : 56:09s, Expires : never
   Hello Priority : 10  (DR)

Neighbor Address : 135.1.1.2,  Interface : toR3
   Component : R1, Uptime : 56:08s, Expires : 1:31s soon
   Hello Priority : 1

Neighbor Address : 135.1.1.6,  Interface : toR4
   Component : R1, Uptime : 56:08s, Expires : 1:32s soon
   Hello Priority : 1  (DR)

Neighbor Address : 135.1.1.5,  Interface : toR4
   Component : R1, Uptime : 56:09s, Expires : never
   Hello Priority : 1

Neighbor Address : 135.1.1.10,  Interface : toR2
   Component : R1, Uptime : 55:27s, Expires : 1:18s soon
   Hello Priority : 1  (DR)

Neighbor Address : 135.1.1.9,  Interface : toR2
   Component : R1, Uptime : 56:09s, Expires : never
   Hello Priority : 1
```

R1 configuration after setting the hello priority:

```
interface create ip toR3 address-netmask 135.1.1.1/30 port et.1.3
interface create ip toR4 address-netmask 135.1.1.5/30 port et.1.4
interface create ip toR2 address-netmask 135.1.1.9/30 port et.1.2
interface create ip i139 address-netmask 139.1.1.1/24 port et.4.8
interface add ip lo0 address-netmask 1.1.1.1/32
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR3 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR2 to-area 0.0.0.255
ospf add stub-host 1.1.1.1 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i139
igmp start
pim sparse create component R1
pim sparse add interface toR3 to-component R1
pim sparse add interface toR4 to-component R1
pim sparse add interface toR2 to-component R1
pim sparse add interface i139 to-component R1
pim sparse set interface toR3 hello-priority 10
pim global start
system set name “R1”
```

BSR Address

The BSR or *bootstrap router* is responsible for providing the RP set information to all of the routers in the PIM-SM domain. A router configured as a candidate bootstrap router is elected if it has the highest priority in the network. In the event of a tie, the candidate BSR with the highest IP address becomes the BSR.

Users may configure a router to be a candidate BSR in one of two ways. The first is to turn on the router’s BSR configuration with the **bsr-on** option. When employing this method, the router selects one of the current PIM-enabled interfaces that is up and advertises its address as the candidate BSR address. Alternately, users may configure the router to use a specific IP address with the **bsr-address <*ip address*>** option. As long as the interface configured as the BSR is *up*, it will advertise BSR messages until it hears from a better candidate BSR in the network. If a better candidate exists, the interface will stop generating BSR messages and set a timer to detect when the elected BSR expires. To configure the candidate BSR priority, use the **pim sparse set component <*name*> bsr-priority <*num*>** command.

Use the following commands to define a router as a possible BSR candidate, use either of the following commands:

```
pim sparse set component R5 bsr-on
```

or...

```
pim sparse set component R5 bsr-address 135.1.1.26
```

No matter which command is used, users must enter another line in the configuration to set the priority:

```
pim sparse set component R5 bsr-priority 5
```

The current BSR status will appear as follows:

```
R5# pim show bsr

PIM-SM Bootstrap Routers:
Component     State       CBSR-Pri  CBSR-Addr        Elected-Pri  Elected-Addr
----------    ----------  --------  ---------------  -----------  ---------------
R5            Elected     5         135.1.1.26       5            135.1.1.26
```

When viewed from a router that is not a candidate BSR (the "State" of the BSR is either *Ineligible*, *Candidate*, or *Elected*).

```
R1# pim show bsr

PIM-SM Bootstrap Routers:
Component     State       CBSR-Pri  CBSR-Addr        Elected-Pri  Elected-Addr
----------    ----------  --------  ---------------  -----------  ---------------
R1            Ineligible  N/A       N/A              5            135.1.1.26
```

CRP address

Candidate RPs are configured on any router selected to be part of the RP set. RP selection is based on a hash of all the candidate RPs at a given priority. Users may configure a router to be a candidate RP in one of two ways. The first is to turn on the router's CRP capabilities with the **pim sparse set component <*name*> crp-on** command. When employing this method, the router selects one of the current PIM-enabled interfaces that is up and sends CRP messages to the currently elected BSR for inclusion in the RP set. The second is to configure the router to use a specific IP address with the **pim sparse set component <*name*> crp-address <*ip address*>** command.

In the following example, R5 and R1 are configured to be candidate RPs. The **pim show crp** command displays the CRP status and **pim show rp-set** displays the RP set that the BSR is advertising.

View from R5

```
R5# pim show crp
PIM-SM Candidate-RPs:
Candidate RPs for R5 (135.1.1.26) pri 0 holdtime: 2:30
                224/4 pri: 0

R5# pim show rp-set
PIM-SM RP-Set:
comp R5:
Group: 224/4
  Dependencies:
  Candidate RPs:
          135.1.1.1: pri: 0 age: 16s expires-in: 2:15s
          135.1.1.26: pri: 0 age: 44:45:24s expires-in: 2:06s
```

View from R3

```
R3# pim show crp
PIM-SM Candidate-RPs:
R3 is not configured to be a C-RP

R3# pim show rp-set
PIM-SM RP-Set:
comp R3:
Group: 224/4
  Dependencies:
  Candidate RPs:
          135.1.1.1: pri: 0 age: 1:21s
          135.1.1.26: pri: 0 age: 1:57:40s
```

View from R1

```
R1# pim show crp
PIM-SM Candidate-RPs:
Candidate RPs for R1 (135.1.1.1) pri 0 holdtime: 2:30
                224/4 pri: 0

R1# pim show rp-set
PIM-SM RP-Set:
comp R1:
Group: 224/4
  Dependencies:
  Candidate RPs:
          135.1.1.1: pri: 0 age: 1:28s
          135.1.1.26: pri: 0 age: 58:18s
```

Static RPs

Configuring static RPs allows users to configure an RP set that cannot change dynamically for a particular domain. When using static RPs, the domain does not require users to configure a candidate RP or BSR. Every router retains a copy of the RP set as it appears in the configuration file—the RP set cannot change except through alterations to the configuration.

To set up a static RP, use the **pim sparse add static-rp <*ip-address*> group <*address/netmask*> to-component <*name*>** command where:

- **<*ip-address*>**
  Refers to the IP address of the RP in the RP set.
- **<*address/netmask*>**
  Is the range of multicast addresses for which this address acts as the RP. Specify **224/4** to use this RP for all multicast addresses.

The following log depicts R1 configured as a static RP:

```
R1(config)# pim sparse add static-rp 139.1.1.1 group 224/4 to-component R1

R1# pim show rp-set
PIM-SM RP-Set:
comp R1:
Group: 224/4
  Dependencies:
  Candidate RPs:
        139.1.1.1: pri: 1 age: 2s
```

R1:

```
interface create ip toR3 address-netmask 135.1.1.1/30 port et.1.3
interface create ip toR4 address-netmask 135.1.1.5/30 port et.1.4
interface create ip toR2 address-netmask 135.1.1.9/30 port et.1.2
interface create ip i139 address-netmask 139.1.1.1/24 port et.4.8
interface add ip lo0 address-netmask 1.1.1.1/32
ip-router policy redistribute from-proto direct to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR3 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR2 to-area 0.0.0.255
ospf add stub-host 1.1.1.1 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface i139
igmp start
pim sparse create component R1
pim sparse add interface toR3 to-component R1
pim sparse add interface toR4 to-component R1
pim sparse add interface toR2 to-component R1
pim sparse add interface i139 to-component R1
pim sparse add static-rp 139.1.1.1 group 224/4 to-component R1
pim sparse set interface toR3 hello-priority 10
pim global start
system set name “R1”
```

Example 3

(*,G) Joins

Typically, a (*,G) join is created by an IGMP client attached to a router. In the following example, CLIENT3 connects to R3 and will send an IGMP Join to the router. When CLIENT3 sends a membership report for a particular group, the router will see the IGMP Join and create (*,G) state. To improve the readability of these tables, the configuration of R3 will change slightly. To begin, add **pim sparse add interface musers to-component musers** to the configuration.

R3:

```
interface create ip toR5 address-netmask 135.1.1.21/30 port et.2.5
interface create ip toR4 address-netmask 135.1.1.13/30 port et.2.4
interface create ip toR1 address-netmask 135.1.1.2/30 port et.2.1
interface create ip musers address-netmask 100.1.1.1/24 port et.2.8
interface add ip lo0 address-netmask 3.3.3.3/32
ip-router global set router-id 3.3.3.3
ip-router policy redistribute from-proto direct to-proto ospf
ip-router policy redistribute from-proto static to-proto ospf
ospf create area 0.0.0.255
ospf add interface toR5 to-area 0.0.0.255
ospf add interface toR4 to-area 0.0.0.255
ospf add interface toR1 to-area 0.0.0.255
ospf add stub-host 3.3.3.3 to-area 0.0.0.255 cost 5
ospf set ase-defaults multicast
ospf start
igmp enable interface musers
igmp start
pim sparse create component R3
pim sparse add interface toR5 to-component R3
pim sparse add interface toR4 to-component R3
pim sparse add interface toR1 to-component R3
pim sparse add interface musers to-component R3
pim global start
system set name “R3”
```

R3 will initially start with an empty table:

```
R3# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L – Local
```

Now CLIENT 3 joins to the 239.1.1.1 group:

```
R3# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local

ROUTES for R3
Source/Mask       Group/Mask       Flags  IIF           OIFs
----------------- ---------------- -----  ------------- ----
0.0.0.0/0         239.1.1.1/32     W      toR5          musers
```

The **0.0.0.0/0** above signifies a (*,G) join and is also marked “W” to identify it as a wildcard entry. If a user enters the **pim show route** command on R5, a (*,G) state will exist on that router.

```
R5# pim show route

PIM Routes:

FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local

ROUTES for R5
Source/Mask       Group/Mask       Flags  IIF           OIFs
----------------- ---------------- -----  ------------- ----
0.0.0.0/0         239.1.1.1/32     W      register      toR3
```

Note: R5 shows the register interface as the incoming interface (IIF) and toR3 as the outgoing interface (OIF). The register interface is used by the RP to decapsulate register messages and forward them down the RPT tree.

(S,G) Joins:

The X-Pedition router creates an (S,G) join when it wants to join to a specific source sending to a multicast group. This scenario typically indicates that multicast data is flowing through the network. The **pim show routes** and **multicast show mfc** commands will reflect this state.

```
R5# pim show route
PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local
ROUTES for R5
Source/Mask       Group/Mask        Flags  IIF           OIFs
----------------  ----------------  -----  ------------  ----
0.0.0.0/0         239.1.1.1/32      W      register      toR3
136.1.1.100/32    239.1.1.1/32      RSL    iserver       toR3


R5# multicast show mfc
Source Address    Group Address     Incoming I/f   Outgoing I/f   Exit Ports
--------------    -------------     ------------   ------------   ----------
136.1.1.100       239.1.1.1         iserver        toR3           et.2.3
```

Note: The IP address of the server is now present, depicting a live source sending traffic—the Multicast Forwarding Cache (MFC) also shows data forwarding through the router. The "R" bit indicates that this flow is RPT Pruned from the RP, or that the data should not be received from the RPF interface of the RP.

Looking at R3:

```
R3# pim show route
PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local
ROUTES for R3
Source/Mask       Group/Mask        Flags  IIF           OIFs
----------------  ----------------  -----  ------------  ----
0.0.0.0/0         239.1.1.1/32      W      toR5          musers
136.1.1.100/32    239.1.1.1/32      S      toR5          musers


R3# multicast show mfc
Source Address    Group Address     Incoming I/f   Outgoing I/f   Exit Ports
--------------    -------------     ------------   ------------   ----------
136.1.1.100       239.1.1.1         toR5           musers         et.2.8
```

Note: The "S" bit denotes that this multicast flow is currently joined to the SPT or *shortest path tree* for the data.

Multiple Clients Joining

Now that one client can see the multicast data, CLIENT 2 should also be able to request the stream. R2 will request the traffic when it receives an IGMP Join request.

Before the join, the view from R5:

```
R5# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local
ROUTES for R5
Source/Mask       Group/Mask       Flags  IIF           OIFs
----------------- ---------------- -----  ------------- ----
0.0.0.0/0         239.1.1.1/32     W      register      toR3
136.1.1.100/32    239.1.1.1/32     RSL    iserver       toR3
```

Before the join, the view from R4 (empty table):

```
R4# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L – Local
```

Before the join, the view from R2 (empty table)

```
R2# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L – Local
```

After receiving the join, the view from R5. You will note that traffic is routing to R3 and R4.

```
R5# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local
ROUTES for R5
Source/Mask       Group/Mask       Flags  IIF           OIFs
----------------- ---------------- -----  ------------- ----
0.0.0.0/0         239.1.1.1/32     W      register      toR3
                                                        toR4
136.1.1.100/32    239.1.1.1/32     RSL    iserver       toR3
                                                        toR4
```

After receiving the join, the view from R4:

```
R4# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local
ROUTES for R4
Source/Mask       Group/Mask       Flags  IIF           OIFs
----------------  ----------------  -----  ------------  ----
0.0.0.0/0         239.1.1.1/32     W      toR5          toR2
136.1.1.100/32    239.1.1.1/32     S      toR5          toR2
```

After receiving the join, the view from R2:

```
R2# pim show route

PIM Routes:
FLAGS: R - RPT W - WC S - SPT N - Neg cache J - Reject
     E - Expanded X - External P - Pending C - Changing
     U - Null IIF T - switching D - Delete L - Local
ROUTES for R2
Source/Mask       Group/Mask       Flags  IIF           OIFs
----------------  ----------------  -----  ------------  ----
0.0.0.0/0         239.1.1.1/32     W      toR4          i139
136.1.1.100/32    239.1.1.1/32     S      toR4          i139
```

Interoperability

The X-Pedition router does not allow users to enable DVMRP and PIM simultaneously. If a user attempts to enable DVMRP and PIM at the same time, one of the following messages will appear:

- %CLI-E-NODVMRPFAC, This command cannot be used when PIM has been configured.
- %CLI-E-NOPIMFAC, This command cannot be used when IGMP or DVMRP has been configured.

To switch between PIM and DVMRP you must remove the protocol's start command from the startup configuration and restart the router.

Note: If a user is running one of the multicast protocols and tries to activate the second protocol, an appropriate error message is displayed and the second protocol is not activated. However, the new configuration is entered into the configuration file without it being marked as being in error.

References

The information in this section referenced the following materials:

Deering, S., D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, L. Wei. *Draft-ietf-idmr-pim-dm-06.* Network Working Group, August 6, 1997.

Fenner, B., M. Handley, H. Holbrook, I. Kouvelas. *Draft-ietf-pim-sm-new-03.* Internet Engineering Task Force, July 20, 2001.

Fenner, B., M. Handley, R. Kermode, D. Thaler. *Draft-ietf-pim-sm-bsr-01.* Internet Engineering Task Force, July 20, 2001

Williamson, B. *Developing IP Multicast Networks, Volume I.* Cisco Press, 2000.

Chapter 21

IP Policy-Based Forwarding Configuration Guide

Overview

You can configure the X-Pedition router to route IP packets according to policies that you define. IP policy-based routing allows network managers to engineer traffic to make the most efficient use of their network resources.

IP policies forward packets based on Layer-3 or Layer-4 IP header information. You can define IP policies to route packets to a set of next-hop IP addresses based on any combination of the following IP header fields:

- IP protocol
- Source IP address
- Destination IP address
- Source Socket
- Destination Socket
- Type of service

For example, you can set up an IP policy to send packets originating from a certain network through a firewall, while letting other packets bypass the firewall. Sites that have multiple Internet service providers can use IP policies to assign user groups to particular ISPs. You can also create IP policies to select service providers based on various traffic types.

Configuring IP Policies

To implement an IP policy, you first create a profile for the packets to be forwarded using an IP policy. For example, you can create a profile defined as “all telnet packets going from network 9.1.0.0/16 to network 15.1.0.0/16”. You then associate the profile with an IP policy. The IP policy specifies what to do with the packets that match the profile. For example, you can create an IP policy that sends packets matching a given profile to next-hop gateway 100.1.1.1.

Configuring an IP policy consists of the following tasks:

- Defining a profile
- Associating the profile with a policy
- Applying the IP policy to an interface

Defining an ACL Profile

An ACL profile specifies the criteria packets must meet to be eligible for IP policy routing. You define profiles with the **acl** command. For IP policy routing, the X-Pedition router uses the packet-related information from the **acl** command and ignores the other fields.

For example, the following **acl** command creates a profile called “prof1” for Telnet packets going from network 9.1.1.5 to network 15.1.1.2:

```
xp(config)# acl prof1 permit ip 9.1.0.0/16 15.1.0.0/16 any any telnet 0
```

See the *Enterasys X-Pedition Command Line Interface Reference Manual* for complete syntax information for the **acl** command.

Note: ACLs for non-IP protocols cannot be used for IP policy routing.

Associating the Profile with an IP Policy

Once you have defined a profile with the **acl** command, you associate the profile with an IP policy by entering one or more **ip-policy** statements. An **ip-policy** statement specifies the next-hop gateway (or gateways) where packets matching a profile are forwarded. (See the *Enterasys X-Pedition Command Line Interface Reference Manual* for complete syntax information for the **ip-policy** command.)

For example, the following command creates an IP policy called “p1” and specifies that packets matching profile “prof1” are forwarded to next-hop gateway 10.10.10.10:

```
xp(config)# ip-policy p1 permit acl prof1 next-hop-list 10.10.10.10
```

You can also set up a policy to prevent packets from being forwarded by an IP policy. For example, the following command creates an IP policy called “p2” that prevents packets matching prof1 from being forwarded using an IP policy:

```
xp(config)# ip-policy p2 deny acl prof1
```

Packets matching the specified profile are forwarded using dynamic routes instead.

Creating Multi-Statement IP Policies

An IP policy can contain more than one **ip-policy** statement. For example, an IP policy can contain one statement that sends all packets matching a profile to one next-hop gateway, and another statement that sends packets matching a different profile to a different next-hop gateway. If an IP policy has multiple **ip-policy** statements, you can assign each statement a sequence number that controls the order in which they are evaluated. Statements are evaluated from lowest sequence number to highest.

For example, the following commands create an IP policy called “p3”, which consists of two IP policy statements. The **ip policy permit** statement has a sequence number of 1, which means it is evaluated before the **ip policy deny** statement, which has a sequence number of 900.

```
xp(config)# ip-policy p3 permit acl prof1 next-hop-list 10.10.10.10 sequence 1
xp(config)# ip-policy p3 deny acl prof2 sequence 900
```

Setting the IP Policy Action

You can use the **action** parameter with the **ip-policy permit** command to specify when to apply the IP policy route with respect to dynamic or statically configured routes. The options of the **action** parameter can cause packets to use the IP policy route first, then the dynamic route if the next-hop gateway specified in the IP policy is unavailable; use the dynamic route first, then the IP policy route; or drop the packets if the next-hop gateway specified in the IP policy is unavailable.

Note: To ensure that a gateway for policy-based routing is available, use the **ip-policy set** command and enable the **pinger** task.

For example, the following command causes packets that match the profile to use dynamic routes first and use the IP policy gateway only if a dynamic route is not available:

```
xp(config)# ip-policy p2 permit acl prof1 action policy-last
```

Setting Load Distribution for Next-Hop Gateways

You can specify up to four next-hop gateways in an **ip-policy** statement. If you specify more than one next-hop gateway, you can use the **ip-policy set** command to control how the load is distributed among them and to check the availability of the next-hop gateways.

Note: To ensure that a gateway for policy-based routing is available, use the **ip-policy set** command and enable the **pinger** task.

By default, each new flow uses the first available next-hop gateway. You can use the **ip-policy set** command to cause flows to use all the next-hop gateways in the **ip-policy permit** statement sequentially. For example, the following command picks the next gateway in the list for each new flow for policy ‘p1’:

```
xp(config)# ip-policy p1 set load-policy round-robin
```

The **ip-policy set** command can also be used to check the availability of next-hop gateways by periodically querying them with ICMP_ECHO_REQUESTS. Only gateways that respond to these requests are used for forwarding packets. For example, the following command checks the availability of next-hop gateways specified in the policy ‘p1’:

```
xp(config)# ip-policy p1 set pinger on
```

Note: Some hosts may have disabled responding to ICMP_ECHO packets. Make sure each next-hop gateway can respond to ICMP_ECHO packets before using this option.

Applying an IP Policy to an Interface

After you define the IP policy, it must be applied to an inbound IP interface with the **ip-policy apply** command. Once the IP policy is applied to the interface, packets start being forwarded according to the IP policy. (See the *Enterasys X-Pedition Command Line Interface Reference Manual* for complete syntax information for the **ip-policy apply** command.)

For example, the following command applies the IP policy ‘p2’ to the interface ‘int2’:

```
xp(config)# ip-policy p2 apply interface int2
```

Applying an IP Policy to Locally Generated Packets

You can apply an IP policy to locally-generated packets (that is, packets generated by the X-Pedition router). For example, the following command applies the IP policy ‘p2’ to locally-generated packets:

```
xp(config)# ip-policy p2 apply local
```

IP Policy Configuration Examples

This section presents some examples of IP policy configurations. The following uses of IP policies are demonstrated:

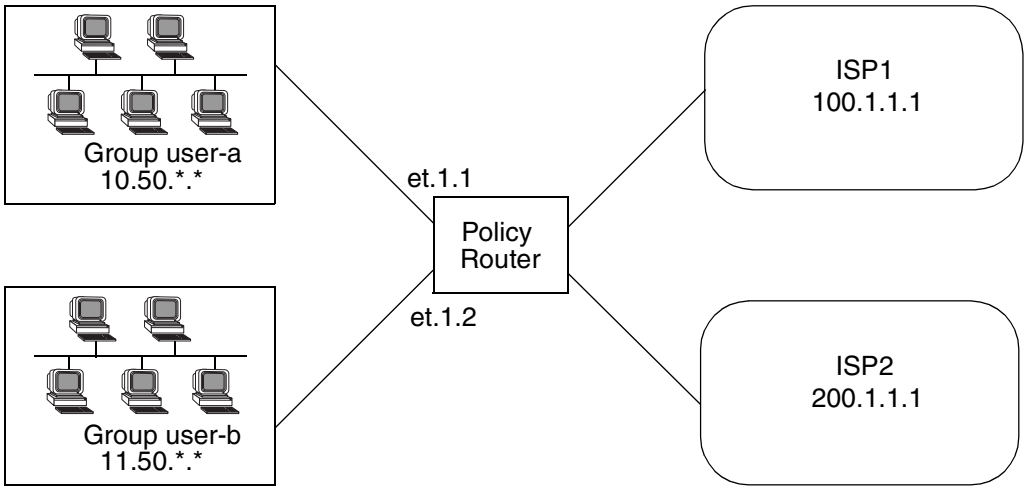
- Routing traffic to different ISPs
- Prioritizing service to customers
- Authenticating users through a firewall
- Firewall load balancing

Routing Traffic to Different ISPs

Sites that have multiple Internet service providers can create IP policies that cause different user groups to use different ISPs. You can also create IP policies to select service providers based on various traffic types.

In the sample configuration in Figure 37, the policy router is configured to divide traffic originating within the corporate network between different ISPs (100.1.1.1 and 200.1.1.1).

Figure 37. Using an IP Policy to Route Traffic to Two Different ISPs



HTTP traffic originating from network 10.50.0.0 for destination 207.31.0.0/16 is forwarded to 100.1.1.1. Non-HTTP traffic originating from network 10.50.0.0 for destination 207.31.0.0/16 is forwarded to 200.1.1.1. All other traffic is forwarded to 100.1.1.1.

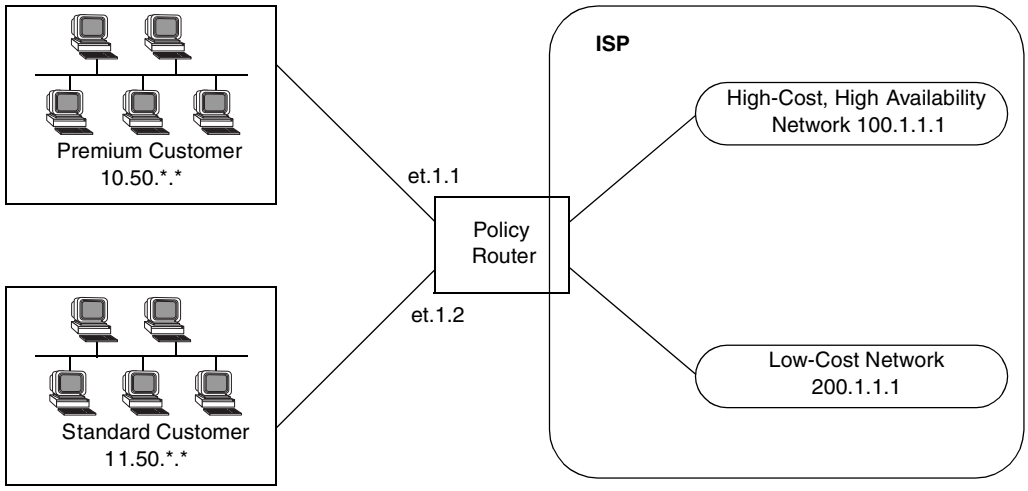
The following is the IP policy configuration for the Policy Router in Figure 37:

```
interface create ip user-a address-netmask 10.50.1.1/16 port et.1.1
interface create ip user-b address-netmask 11.50.1.1/16 port et.1.2

acl user-a-http permit ip 10.50.0.0/16 207.31.0.0/16 any http 0
acl user-a permit ip 10.50.0.0/16 207.31.0.0/16 any any 0
acl user-b permit ip 11.50.0.0/16 any any any 0

ip-policy net-a permit acl user-a-http next-hop-list 100.1.1.1 action policy-first sequence 20

ip-policy net-a permit acl user-a next-hop-list 200.1.1.1 action policy-only sequence 25

ip-policy net-a apply interface user-a

ip-policy net-b permit acl user-b next-hop-list 200.1.1.1 action policy-first

ip-policy net-b apply interface user-b
```

Prioritizing Service to Customers

An ISP can use policy-based routing on an access router to supply different customers with different levels of service. The sample configuration in Figure 38 shows an X-Pedition router using an IP policy to classify customers and route traffic to different networks based on customer type.

Figure 38. Using an IP Policy to Prioritize Service to Customers



Traffic from the premium customer is load balanced across two next-hop gateways in the high-cost, high-availability network. If neither of these gateways is available, then packets are forwarded based on dynamic routes learned via routing protocols.

Traffic from the standard customer always uses one gateway (200.1.1.1). If for some reason that gateway is not available, packets from the standard customer are dropped.

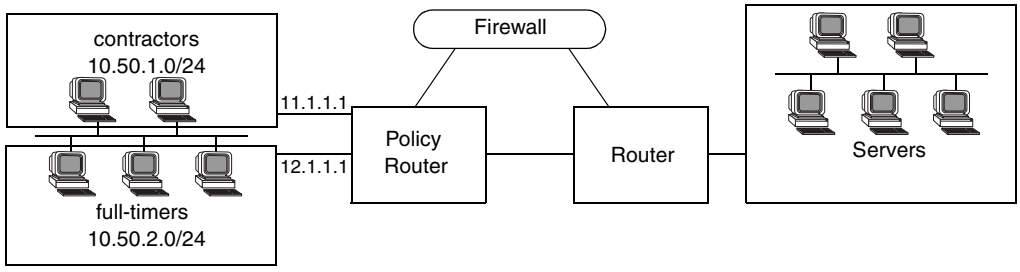
The following is the IP policy configuration for the Policy Router in Figure 38:

```
interface create ip premium-customer address-netmask 10.50.1.1/16 port et.1.1

interface create ip standard-customer address-netmask 11.50.1.1/16 port et.1.2

acl premium-customer permit ip 10.50.0.0/16 any any any 0
acl standard-customer   permit ip 11.50.0.0/16 any any any 0

ip-policy p1 permit acl premium-customer next-hop-list “100.1.1.1 100.1.1.2" action policy-first
      sequence 20

ip-policy apply interface premium-customer

ip-policy p2 permit acl standard-customer next-hop-list 200.1.1.1 action policy-only sequence 30

ip-policy apply interface standard-customer
```

Authenticating Users Through a Firewall

You can define an IP policy that authenticates packets from certain users via a firewall before accessing the network. If for some reason the firewall is not responding, the packets to be authenticated are dropped. Figure 39 illustrates this kind of configuration.

Figure 39. Using an IP Policy to Authenticate Users Through a Firewall



Packets from users defined in the “contractors” group are sent through a firewall. If the firewall cannot be reached packets from the contractors group are dropped. Packets from users defined in the “full-timers” group do not have to go through the firewall.

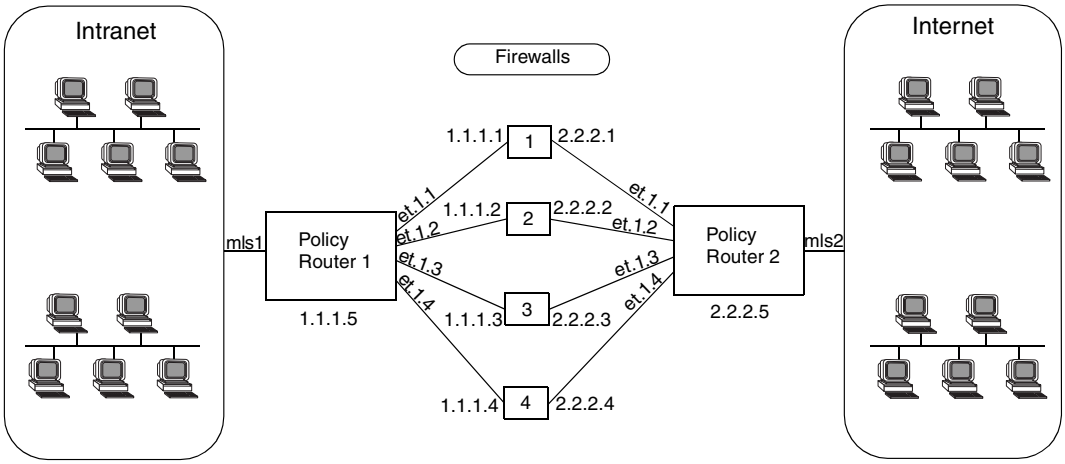
The following is the IP policy configuration for the Policy Router in Figure 39:

```
interface create ip mls0 address-netmask 10.50.1.1/16 port et.1.1

acl contractors permit ip 10.50.1.0/24 any any any 0
acl full-timers permit ip 10.50.2.0/24 any any any 0

ip-policy access permit acl contractors next-hop-list 11.1.1.1 action policy-only
ip-policy access permit acl full-timers next-hop-list 12.1.1.1 action policy-first
ip-policy access apply interface mls0
```

Firewall Load Balancing

The next hop gateway can be selected by the following information in the IP packet: source IP, destination IP, or both the source and destination IP. Figure 40 illustrates this configuration.

Figure 40. Selecting Next Hop Gateway from IP Packet Information



One session should always go to a particular firewall for persistence.

The following is the configuration for Policy Router 1 in Figure 40.

```
vlan create firewall
vlan add ports et.1.(1-5) to firewall

interface create ip firewall address-netmask 1.1.1.5/16 vlan firewall

acl firewall permit ip any any any 0

ip-policy p1 permit acl firewall next-hop-list “1.1.1.1 1.1.1.2 1.1.1.3 1.1.1.4” action policy-only

ip-policy p1 set load-policy ip-hash both

ip-policy p1 apply interface mls1
```

The following is the configuration for Policy Router 2 in Figure 40.

```
vlan create firewall
vlan add ports et.1.(1-5) to firewall

interface create ip firewall address-netmask 2.2.2.5/16 vlan firewall

acl firewall permit ip any any any 0

ip-policy p2 permit acl firewall next-hop-list “2.2.2.1 2.2.2.2 2.2.2.3 2.2.2.4” action policy-only

ip-policy p2 set load-policy ip-hash both

ip-policy p2 apply interface mls2
```

Monitoring IP Policies

The **ip-policy show** command reports information about active IP policies, including profile definitions, policy configuration settings, and next-hop gateways. The command also displays statistics about packets that have matched an IP policy statement as well as the number of packets that have been forwarded to each next-hop gateway.

For example, to display information about an active IP policy called “p1”, enter the following command in Enable mode:

```
xp# ip-policy show policy-name p1

-----------------------------------------------------------------------------
IP Policy name     : p1 ❶
Applied Interfaces : int1 ❷
Load Policy        : first available ❸

❹          ❺               ❻             ❼        ❽        ❾    ❿
ACL        Source IP/Mask  Dest. IP/Mask SrcPort  DstPort  TOS  Prot
---        --------------  ------------- -------- -------- ---  ----
prof1      9.1.1.5/32      15.1.1.2      any      any      0    IP
prof2      2.2.2.2/32      anywhere      any      any      0    IP
everything anywhere        anywhere      any      any      0    IP

                                   Next Hop Information
                                   --------------------
⓫      ⓬      ⓭          ⓮   ⓯           ⓰           ⓱    ⓲
Seq    Rule   ACL        Cnt Action      Next Hop    Cnt  Last
---    ----   --------   --- ----------- --------    ---  ----
10     permit prof1      0   Policy Only 11.1.1.2    0    Dwn
20     permit prof2      0   Policy Last 1.1.1.1     0    Dwn
                                         2.2.2.2     0    Dwn
                                         3.3.3.3     0    Dwn
999    permit everything 0   Policy Only drop        N/A  N/A
65536  deny   deny       0   N/A         normal fwd  N/A  N/A
       ⓳
```

Legend:

1. The name of the IP policy.
2. The interface where the IP policy was applied.
3. The load distribution setting for IP-policy statements that have more than one next-hop gateway; either first available (the default) or round-robin.
4. The names of the profiles (created with an **acl** statement) associated with this IP policy.
5. The source address and filtering mask of this flow.
6. The destination address and filtering mask of this flow.
7. For TCP or UDP, the number of the source TCP or UDP port.
8. For TCP or UDP, the number of the destination TCP or UDP port.
9. The TOS value in the packet.
10. IP protocol (ICMP, TCP UDP).
11. The sequence in which the statement is evaluated. IP policy statements are listed in the order they are evaluated (lowest sequence number to highest).

12. The rule to apply to the packets matching the profile: either permit or deny
13. The name of the profile (ACL) of the packets to be forwarded using an IP policy.
14. The number of packets that have matched the profile since the IP policy was applied (or since the **ip-policy clear** command was last used).
15. The method by which IP policies are applied with respect to dynamic or statically configured routes; possible values are Policy First, Policy Only, or Policy Last.
16. The list of next-hop gateways in effect for the policy statement.
17. The number of packets that have been forwarded to this next-hop gateway.
18. The state of the link the last time an attempt was made to forward a packet; possible values are up, dwn, or N/A.
19. Implicit deny rule that is always evaluated last, causing all packets that do not match one of the profiles to be forwarded normally (with dynamic routes).

See the *Enterasys X-Pedition Command Line Interface Reference Manual* for complete syntax information for the **ip-policy show** command.

Chapter 22

Network Address Translation Configuration Guide

Overview

Network Address Translation (NAT) allows an IP address used within one network to be translated into a different IP address used within another network. NAT is often used to map addresses used in a private, local intranet to one or more addresses used in the public, global Internet. NAT provides the following benefits:

- Limits the number of IP addresses used for private intranets that are required to be registered with the Internet Assigned Numbers Authority (IANA).
- Conserves the number of global IP addresses needed by a private intranet (for example, an entity can use a single IP address to communicate on the Internet).
- Maintains privacy of local networks, as internal IP addresses are hidden from public view.

With NAT, the local network is designated the *inside* network and the global Internet is designated the *outside* network. In addition, the X-Pedition router supports Port Address Translation (PAT) for either static or dynamic address bindings.

The X-Pedition router's current ACL/NAT implementation does not make provisions for running standard or PASV FTP sessions across a translated interface when only ports 20 (FTP data port) and 21 (FTP control port) are open for communication. Because FTP will use other higher-numbered ports to establish TCP sessions, FTP sessions established across a NAT-translated interface may hang if these other TCP ports are not open for communication. In order to allow FTP to establish a TCP session on higher-numbered ports, the NAT-associated ACL must be set up to allow incoming traffic from any port. When running this configuration, it is suggested that NAT secure-plus is enabled (**nat set secure-plus on**) in order to increase security and prevent private address leaks. For more information, reference RFC 1579 ("Firewall-Friendly FTP").

The X-Pedition router allows you to create the following NAT address bindings:

- Static, one-to-one binding of inside, local address or address pool to outside, global address or address pool. A static address binding does not expire until the command that defines the binding is negated. IP addresses defined for static bindings cannot be reassigned. For static address bindings, PAT allows TCP or UDP port numbers to be translated along with the IP addresses.
- Dynamic binding between an address from a pool of local addresses to an address from a pool of outside addresses. With dynamic address binding, you define local and global address pools from which the addresses bindings can be made. IP addresses defined for dynamic binding are reassigned whenever they become free. For dynamic address bindings, PAT allows port address translation if no addresses are available from the global address pool. PAT allows port address translation for each address in the global pool. The ports are dynamically assigned between the range of 1024 to 4999. Hence, you have about 4,000 ports per global IP address.

  Dynamic bindings are removed automatically when the flow count goes to zero. At this point, the corresponding port (if PAT enabled) or the global IP address is freed and can be reused the next time. Although there are special cases like FTP where the flows are not installed for the control path, the binding will be removed only by the dynamic binding timeout interval.

Note: The X-Pedition router's version of PAT does not support an Application Level Gateway (ALG) for Generic Routing Encapsulation (GRE).

Configuring NAT

The following are the steps in configuring NAT on the X-Pedition router:

1. Setting the NAT interfaces to be “inside” or “outside.”
2. Setting the NAT rules (static or dynamic).

Note: NAT interfaces do not support VRRP.

Setting Inside and Outside Interfaces

When NAT is enabled, address translation is only applied to those interfaces which are defined to NAT as “inside” or “outside” interfaces. NAT only translates packets that arrive on a defined inside or outside interface. To specify an interface as inside (local) or outside (global), enter the following command in Configure mode.

Define an interface as inside or outside for NAT.	**nat set interface** <*InterfaceName*> **inside**|**outside**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Setting NAT Rules

Static

You create NAT static bindings by entering the following command in Configure mode.

Enable NAT with static address binding.	**nat create static protocol ip**|**tcp**|**udp** **local-ip** <*local-ip-add/address range*> **global-ip** <*global-ip-add/address range*> [**local-port** <*tcp/udp local-port*>|**any**] [**global-port** <*tcp/udp global-port*>|**any**]

Dynamic

You create NAT dynamic bindings by entering the following command in Configure mode.

Enable NAT with dynamic address binding.	**nat create dynamic local-acl-pool** <*local-acl*> **global-pool** <*ip-addr/ip-addr-range/ip-addr-list/ip-addr-mask*> [**matches-interface** <*interface*>] [**enable-ip-overload**]

For dynamic address bindings, you define the address pools with previously-created ACLs. You can also specify the **enable-port-overload** parameter to allow PAT.

Forcing Flows through NAT

If a host on the outside global network knows an inside local address, it can send a message directly to the inside local address. By default, the X-Pedition router will route the message to the destination. You can force *all* flows between the inside local pool and the outside global network to be translated. This prevents a host on the outside global network from being allowed to send messages directly to any address in the local address pool. You may force address translation of all flows to and from the inside local pool by entering the following command in Configure mode.

Force all flows to and from local address pool to be translated.	**nat set secure-plus on|off**

The X-Pedition router's current ACL/NAT implementation does not make provisions for running standard or PASV FTP sessions across a translated interface when only ports 20 (FTP data port) and 21 (FTP control port) are open for communication. Because FTP will use other higher-numbered ports to establish TCP sessions, FTP sessions established across a NAT-translated interface may hang if these other TCP ports are not open for communication. In order to allow FTP to establish a TCP session on higher-numbered ports, the NAT-associated ACL must be set up to allow incoming traffic from any port. When running this configuration, it is suggested that NAT secure-plus is enabled (**nat set secure-plus on**) in order to increase security and prevent private address leaks. For more information, please reference RFC 1579 (“Firewall-Friendly FTP”).

Managing Dynamic Bindings

As mentioned previously, dynamic address bindings expire only after a period of non-use or when they are manually deleted. The default timeout for dynamic address bindings is 1440 minutes (24 hours). You can manually delete dynamic address bindings for a specific address pool or delete all dynamic address bindings. To set the timeout for dynamic address bindings, enter the following command in Configure mode.

Set timeout for dynamic address bindings.	**nat set dynamic-binding-timeout** <*minutes*> |**disable**

To flush dynamic address bindings, enter the following command in Enable mode.

Flush all dynamic address bindings.	**nat flush-dynamic-binding all**
Flush dynamic address bindings based on local and global ACL pools.	**nat flush-dynamic-binding pool-specified local-acl-pool** <*local-acl*> **global-pool** <*ip-addr/ip-addr-range/ip-addr-list/ip-addr-mask*>
Flush dynamic address bindings based on binding type.	**nat flush-dynamic-binding type-specified dynamic** | **overloaded-dynamic**
Flush dynamic address bindings based on application.	**nat flush-dynamic-binding owner-specified dns**|**ftp-control**|**ftp-data**

NAT and DNS

NAT can translate an address that appears in a Domain Name System (DNS) response to a name or inverse lookup. For example, if an outside host sends a name lookup to an inside DNS server, the inside DNS server can respond with a local IP address, which NAT translates to a global address.

You create NAT dynamic bindings for DNS by entering the following command in Configure mode.

Enable NAT with dynamic address binding for DNS query/reply.	**nat create dynamic local-acl-pool** <*outside-local-acl*> **global-pool** <*ip-addr/ip-addr-range/ip-addr-list/ip-addr-mask*>

DNS packets that contain addresses that match the ACL specified by **outside-local-acl-pool** are translated using local addresses allocated from **inside-global-pool**.

The default timeout for DNS dynamic address bindings is 30 minutes. You can change this timeout by entering the following command in Configure mode:

Specify the timeout for DNS bindings.	**nat set dns-session-timeout** <*minutes*>

NAT and ICMP Packets

NAT translates addresses embedded in the data portion of the following types of ICMP error messages:

- Destination unreachable (type 3)
- Source quench (type 4)
- Redirect (type 5)
- Time exceeded (type 11)
- Parameter problem (type 12)

NAT and FTP

File Transfer Protocol (FTP) packets require special handling with NAT, because the FTP PORT command packets contain IP address information within the data portion of the packet. It is therefore important for NAT to know which control port is used for FTP (the default is port 21) and the timeout for the FTP session (the default is 30 minutes). If FTP packets will arrive on a different port number, you need to specify that port to NAT. To define FTP parameters to NAT, enter the following commands in Configure mode.

Specify the FTP control port.	**nat set ftp-control-port** <*port number*>
Specify the FTP session timeout.	**nat set ftp-session-timeout** <*minutes*>

If PAT is enabled, NAT checks packets for the FTP PORT command. If a packet is to be translated (as determined by the ACL specified for the dynamic address binding), NAT creates a dynamic binding for the PORT command. An outside host will only see a global IP address in an FTP response and not the local IP address.

The X-Pedition router's current ACL/NAT implementation does not make provisions for running standard or PASV FTP sessions across a translated interface when only ports 20 (FTP data port) and 21 (FTP control port) are open for communication. Because FTP will use other higher-numbered ports to establish TCP sessions, FTP sessions established across a NAT-translated interface may hang if these other TCP ports are not open for communication. In order to allow FTP to establish a TCP session on higher-numbered ports, the NAT-associated ACL must be set up to allow incoming traffic from any port. When running this configuration, it is suggested that NAT secure-plus is enabled (**nat set secure-plus on**) in order to increase security and prevent private address leaks. For more information, reference RFC 1579 ("Firewall-Friendly FTP").

NAT and VRRP

NAT interfaces do not currently support VRRP.

Monitoring NAT

To display NAT information, enter the following command in Enable mode.

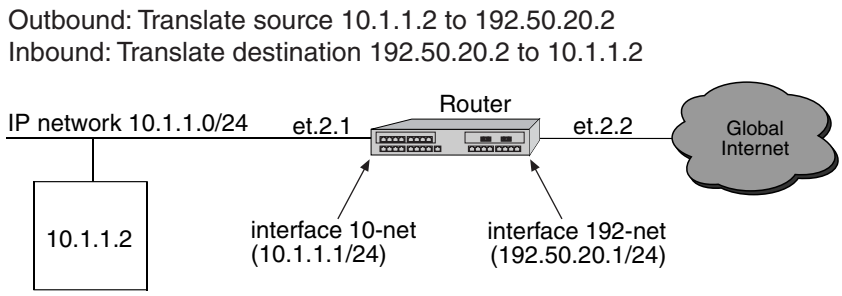
Display NAT information.	**nat show [translations all**|<*type*>**] [timeouts] [statistics]**

Configuration Examples

This section shows examples of NAT configurations.

Static Configuration

The following example configures a static address binding for inside address 10.1.1.2 to outside address 192.50.20.2:



The first step is to create the interfaces:

```
interface create ip 10-net address-netmask 10.1.1.1/24 port et.2.1
interface create ip 192-net address-netmask 192.50.20.1/24 port et.2.2
```

Next, define the interfaces to be NAT “inside” or “outside”:

```
nat set interface 10-net inside
nat set interface 192-net outside
```

Then, define the NAT static rules:

```
nat create static protocol ip local-ip 10.1.1.2 global-ip 192.50.20.2
```

Using Static NAT

Static NAT can be used when the local and global IP addresses are to be bound in a fixed manner. These bindings never get removed nor time out until the static NAT command itself is negated. Static binding is recommended when you have a need for a permanent type of binding.

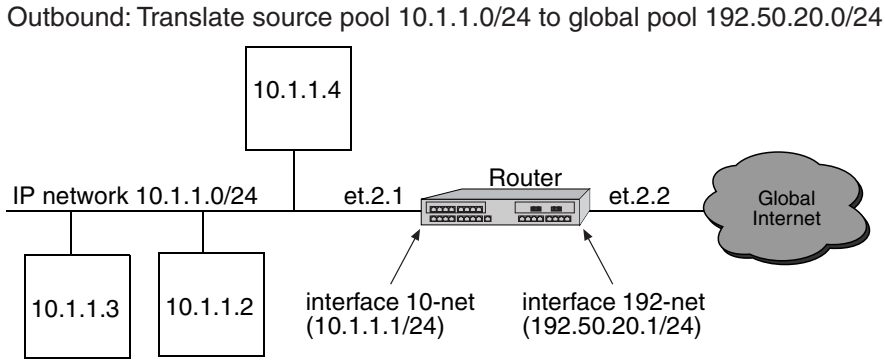
The other use of static NAT is when the out to in traffic is the first to initialize a connection, i.e., the first packet is coming from outside to inside. This could be the case when you have a server in the local network and clients located remotely. Dynamic NAT would not work for this case as bindings are always created when an in to out Internet connection occurs. A typical example is a web server inside the local network, which could be configured as follows:

```
nat create static protocol tcp local-ip 10.1.1.2 global-ip 192.50.20.2 local-port 80 global-port 80
```

This server, 10.1.1.2, is advertised as 192.50.20.2 to the external network.

Dynamic Configuration

The following example configures a dynamic address binding for inside addresses 10.1.1.0/24 to outside address 192.50.20.0/24:



The first step is to create the interfaces:

```
interface create ip 10-net address-netmask 10.1.1.1/24 port et.2.1
interface create ip 192-net address-netmask 192.50.20.1/24 port et.2.2
```

Next, define the interfaces to be NAT “inside” or “outside”:

```
nat set interface 10-net inside
nat set interface 192-net outside
```

Then, define the NAT dynamic rules by first creating the source ACL pool and then configuring the dynamic bindings:

```
acl lcl permit ip 10.1.1.0/24
nat create dynamic local-acl-pool lcl global-pool 192.50.20.0/24
```

Using Dynamic NAT

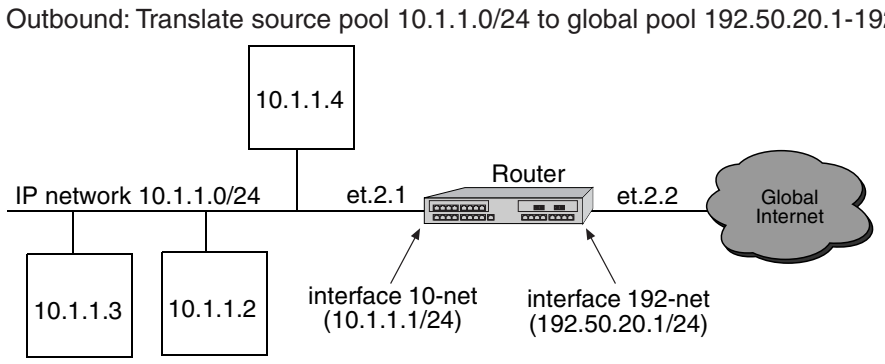
Dynamic NAT can be used when the local network (inside network) is going to initialize the connections. It creates a binding at run time when a packet is sent from a local network, as defined by the NAT dynamic local ACl pool. The network administrator does not have to worry about the way in which the bindings are created; the network administrator just sets the pools and the X-Pedition router automatically chooses a free global IP from the global pool for the local IP.

Dynamic bindings are removed when the flow count for that binding goes to zero or the timeout has been reached. The free globals are used again for the next packet.

A typical problem is that if there are more local IP addresses as compared to global IP addresses in the pools, then packets will be dropped if all the globals are used. A solution to this problem is to use PAT with NAT dynamic. This is only possible with TCP or UDP protocols.

Dynamic NAT with IP Overload (PAT) Configuration

The following example configures a dynamic address binding for inside addresses 10.1.1.0/24 to outside address 192.50.20.0/24:



The first step is to create the interfaces:

```
interface create ip 10-net address-netmask 10.1.1.1/24 port et.2.1
interface create ip 192-net address-netmask 192.50.20.1/24 port et.2.2
```

Next, define the interfaces to be NAT “inside” or “outside”:

```
nat set interface 10-net inside
nat set interface 192-net outside
```

Then, define the NAT dynamic rules by first creating the source ACL pool and then configuring the dynamic bindings:

```
acl lcl permit ip 10.1.1.0/24
nat create dynamic local-acl-pool lcl global-pool 192.50.20.1-192.50.20.3
```

Using Dynamic NAT with IP Overload

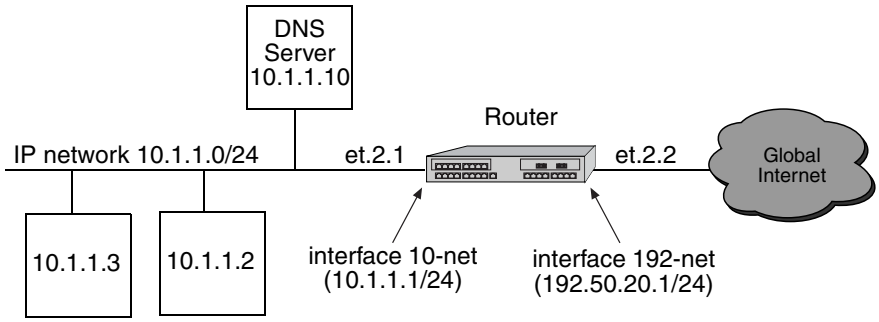
Dynamic NAT with IP overload can be used when the local network (inside network) will be initializing the connections using TCP or UDP protocols. It creates a binding at run time when the packet comes from a local network defined in the NAT dynamic local ACL pool. The difference between the dynamic NAT and dynamic NAT with PAT is that PAT uses port (layer 4) information to do the translation. Hence, each global IP has about 4000 ports that can be translated. NAT on the X-Pedition router uses the standard BSD range of ports from 1024-4999 which is fixed and cannot be configured by the user. The network administrator does not have to worry about the way in which the bindings are created; he/she just sets the pools and the X-Pedition router automatically chooses a free global IP from the global pool for the local IP.

Dynamic bindings are removed when the flow count goes to zero or the timeout has been reached. The removal of bindings frees the port for that global and the port is available for reuse. When all the ports for that global are used, then ports are assigned from the next free global. If no more ports and globals are available, the packets will be dropped.

Dynamic NAT with DNS

The following example configures a DNS dynamic address binding for outside address 192.50.20.2-192.50.20.9 to inside addresses 10.1.1.0/24:

DNS server static binding of 10.1.1.10 to 192.50.20.10



The first step is to create the interfaces:

```
interface create ip 10-net address-netmask 10.1.1.1/24 port et.2.1
interface create ip 192-net address-netmask 192.50.20.1/24 port et.2.2
```

Next, define the interfaces to be NAT “inside” or “outside”:

```
nat set interface 10-net inside
nat set interface 192-net outside
```

Then, define the NAT dynamic rules by first creating the source ACL pool and then configuring the dynamic bindings:

```
acl lcl permit ip 10.1.1.0/24
nat create dynamic local-acl-pool lcl global-pool 192.50.20.2-192.50.20.9
nat create static local-ip 10.1.1.10 global-ip 192.50.20.10 protocol ip
```

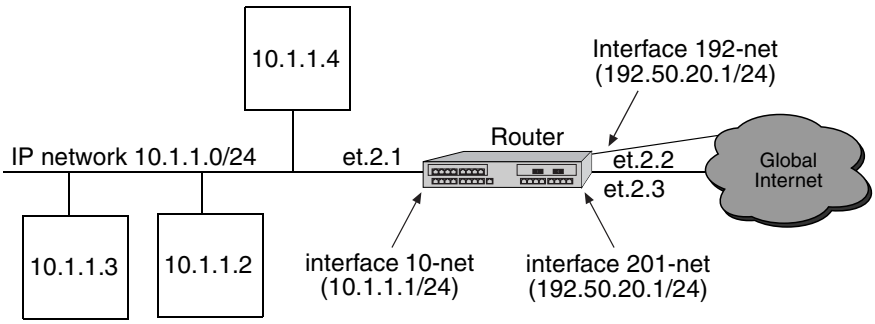
Using Dynamic NAT with DNS

When a client from outside sends a query to the static global IP address of the DNS server, NAT will translate the global IP address to the local IP address of the DNS server. The DNS server will resolve the query and respond with a reply. The reply can include the local IP address of a host inside the local network (for example, 10.1.1.2); this local IP address will be translated by NAT into a global IP address (for example, 192.50.20.2) in a dynamic binding for the response.

Dynamic NAT with Outside Interface Redundancy

The following example configures a dynamic address binding for inside addresses 10.1.1.1/24 to outside addresses 192.50.20.1/24 on interface 192-net and to outside addresses 201.50.20.1/24 on interface 201-net:

Outbound: Translate source pool 10.1.1.0/24 to global pool 192.50.20.0/24
Translate source pool 10.1.1.0/24 to global pool 201.50.20.0/24



The first step is to create the interfaces:

```
interface create ip 10-net address-netmask 10.1.1.1/24 port et.2.1
interface create ip 192-net address-netmask 192.50.20.1/24 port et.2.2
interface create ip 201-net address-netmask 201.50.20.1/24 port et.2.3
```

Next, define the interfaces to be NAT “inside” or “outside”:

```
nat set interface 10-net inside
nat set interface 192-net outside
nat set interface 201-net outside
```

Then, define the NAT dynamic rules by first creating the source ACL pool and then configuring the dynamic bindings:

```
acl lcl permit ip 10.1.1.1/24
nat create dynamic local-acl-pool lcl global-pool 192.50.20.1/24 matching-if 192-net
nat create dynamic local-acl-pool lcl global-pool 210.50.20.1/24 matching-if 201-net
```

Using Dynamic NAT with Matching Interface Redundancy

If you have redundant connections to the remote network via two different interfaces, you can use NAT for translating the local address to the different global pool specified for the two connections. This case is possible when you have two ISPs connected on two different interfaces to the Internet. Through a routing protocol, some routes will result in traffic going out of one interface and for others going out on the other interface. NAT will check which interface the packet is going out from before selecting a global pool. Hence, you can specify two different global pools with the same local ACL pool on two different interfaces.

Chapter 23

Web Hosting Configuration Guide

Overview

Accessing information on websites for both work or personal purposes is becoming a normal practice for an increasing number of people. For many companies, fast and efficient web access is important for both external customers who need to access the company websites, as well as for users on the corporate intranet who need to access Internet websites.

The following features on the X-Pedition router provide ways to improve web access for external and internal users:

- Load balancing allows incoming HTTP requests to a company's website to be distributed across several physical servers. If one server should fail, other servers can pick up the workload.
- Web caching allows HTTP requests from internal users to Internet sites to be redirected to cached Web objects on local servers. Not only is response time faster since requests can be handled locally, but overall WAN bandwidth usage is reduced.

Note: Load balancing and web caching can be performed using application software, however, the X-Pedition router can perform these functions much faster as the redirection is handled by specialized hardware.

Load Balancing

You can use the load balancing feature on the X-Pedition router to distribute session load across a group of servers. If you configure the X-Pedition router to provide load balancing, client requests that go through the X-Pedition router can be redirected to any one of several predefined hosts. With load balancing, clients access servers through a virtual IP. The X-Pedition router transparently redirects the requests with no change required on the clients or servers; all configuration and redirection is done on the X-Pedition router.

Configuring Load Balancing

The following are the steps in configuring load balancing on the X-Pedition router:

1. Create a logical group of load balancing servers and define a virtual IP for the group.
2. Define the servers in the group.
3. Specify optional operating parameters for the group of load balancing servers or for individual servers in the group.

Creating the Server Group

To use load balancing, users create a logical group of load balancing servers and define a virtual IP for the server that the clients will use to access the server pool. To create the server group and define the virtual IP for the server, enter the following command in Configure mode:

Create group of load balancing servers.	**load-balance create group-name** <*group name*> **virtual-ip** <*ipaddr*> [**virtual-port** <*port number*>] **protocol tcp|udp** [**persistence-level tcp|ssl|vpn|sticky**]
Create a range of load balancing server groups.	**load-balance create vip-range-name** <*range name*> **vip-range** <*range*> [**virtual-port** <*port number*>] **protocol tcp|udp** [**persistence-level tcp|ssl|vpn|sticky**]

Note: Do not use an IP address for load-balancing that is already configured for VRRP.

Adding Servers to the Load Balancing Group

Once a logical server group is created, you specify the servers that can handle client requests. When the X-Pedition router receives a client request directed to the virtual server address, it redirects the request to the actual server address and port. Server selection is done according to the specified policy.

To add servers to the server group, enter the following command in Configure mode:

Add load balancing servers to a specific server group.	**load-balance add host-to-group** <*ipaddr/range*> **group-name** <*group name*> **port** <*port number*> [**weight** <*weight*>]
Add range of load balancing servers to a range of server groups.	**load-balance add host-to-vip-range** <*range*> **vip-range-name** <*range name*> **port** <*port number*> [**weight** <*weight*>]

Note: Do not use an IP address for load-balancing that is already configured for VRRP.

Session Persistence

Load balancing clients connect to a *virtual* IP address which, in reality, is redirected to one of several physical servers in a load balancing group. In many web page display applications, a client may have its requests redirected to and serviced by different servers in the group. In certain situations, however, it may be critical that all traffic for the client be directed to the same physical server for the duration of the session; this is the concept of *session persistence*.

When the X-Pedition router receives a new session request from a client for a specific virtual address, the X-Pedition router creates a *binding* between the client (source) IP address/port socket and the (destination) IP address/port socket of the load balancing server selected for this client. Subsequent packets from clients are compared to the list of bindings: if there is a match, the packet is sent to the same server previously selected for this client; if there is not a match, a new binding is created. How the X-Pedition router determines the binding match for session persistence is configured with the **persistence-level** option when the load balancing group is created (see *Creating the Server Group* on page 392).

There are four configurable levels of session persistence:

- TCP persistence: a binding is determined by the matching the source IP/port address as well as the virtual destination IP/port address. For example, requests from the client address of 134.141.176.10:1024 to the virtual destination address 207.135.89.16:80 is considered one session and would be directed to the same load balancing server (for example, the server with IP address 10.1.1.1). A request from a different source socket from the same client address to the same virtual destination address would be considered another session and may be directed to a different load balancing server (for example, the server with IP address 10.1.1.2). This is the default level of session persistence.

- SSL persistence: a binding is determined by matching the source IP address and the virtual destination IP/port address. Note that requests from *any* source socket with the client IP address are considered part of the same session. For example, requests from the client IP address of 134.141.176.10:1024 or 134.141.176.10:1025 to the virtual destination address 207.135.89.16:80 would be considered one session and would be directed to the same load balancing server (for example, the server with IP address 10.1.1.1).
- Sticky persistence: a binding is determined by matching the source and destination IP addresses only. This allows all requests from a client to the same virtual address to be directed to the same load balancing server. For example, both HTTP and HTTPS requests from the client address 134.141.176.10 to the virtual destination address 207.135.89.16 would be directed to the same load balancing server (for example, the server with IP address 10.1.1.1).
- Virtual private network (VPN) persistence: for VPN traffic using Encapsulated Security Payload (ESP) mode of IPSec, a binding is determined by matching the source and destination IP addresses in the secure key transfer request to subsequent client requests. This allows both the secure key transfer and subsequent data traffic from a particular client to be directed to the same load balancing server. The default port number recognized by the X-Pedition router for secure key transfer in VPN is 500; you can use the **load-balance set vpn-dest-port** command to specify a different port number.

You can use the **load-balance show source-mappings** command to display information about the current list of bindings.

The binding between a client (source) and a load balancing server times out after a certain period of non-activity. The default timeout depends upon the session persistence level configured, as follows:

Persistence Level	Default Binding Timeout
TCP	3 minutes
SSL	120 minutes
Sticky	120 minutes
VPN	3 minutes

You can change the timeout for a load balancing group with the **load-balance set aging-for-src-maps** command.

The X-Pedition router also supports *netmask persistence*, which can be used with any of the four levels of session persistence. A netmask (configured with the **load-balance set client-proxy-subnet** command) is applied to the source IP address and this address is compared to the list of bindings: if a binding exists, the packet is sent to the same load balancing server previously selected for this client; if there is not a match, a new binding is created.

This feature allows a range of source IP addresses (with different port numbers) to be sent to the same load balancing server. This is useful where client requests may go through a proxy that uses Network Address Translation or Port Address Translation or multiple proxy servers. During a session, the source IP address can change to one of several sequential addresses in the translation pool; the netmask allows client requests to be sent to the same server.

Optional Group or Server Operating Parameters

There are several commands you can specify that affect the operating parameters of individual servers or the entire group of load balancing servers. In many cases, there are default parameter values and you only need to specify a command if you wish to change the default operation. For example, you can specify the policy to be used for distributing the workload for a group of load balancing servers. By default, the X-Pedition router assigns sessions to servers in a round-robin (sequential) manner.

Specifying Load Balancing Policy

The default policy for distributing workload among the load balancing servers is “round-robin,” where the X-Pedition router selects the server on a rotating basis without regard to the load on individual servers. Other policies can be chosen for the group, including least loaded, where the server with the fewest number of sessions bound to it is selected to service a new session. The weighted round robin policy is a variation of the round-robin policy, where each server takes on new sessions according to its assigned weight. If you choose the weighted round robin policy, you must assign a weight to each server that you add to the load balancing group.

To specify the load balancing policy, enter the following command in Configure mode:

Specify load balancing policy.	**load-balance set policy-for-group** <*group name*> **policy** <*policy*>

Note: These policies only affect groups created with the parameter **protocol tcp**; groups created with the parameter **protocol udp** are load-balanced using a fixed IP-hash policy.

Specifying a Connection Threshold

By default, there is no limit on the number of sessions that a load balancing server can service. You can configure a maximum number of connections that each server in a group can service. To specify the connection threshold for servers in the group, enter the following command in Configure mode:

Specify maximum number of connections for all servers in the group.	**load-balance set group-conn-threshold** <*group name*> **limit** <*maximum connections*>

Note: This limits the number of connections for *each* server in the group, not the total number of connections for the group.

Verifying Servers and Applications

The X-Pedition router *automatically* performs the following types of verification for the attached load balancing servers/applications:

- Verifies the state of the server by sending a ping to the server at 5-second intervals. If the X-Pedition router does not receive a reply from a server after four ping requests, the server is considered to be “down.”
- Checks that an application session on the server can be established by doing a simple TCP handshake with the application on the configured physical port of the server at 15-second intervals. If the X-Pedition router does not receive a reply from the application after four tries, the application is considered to be “down.”

You can change the intervals at which pings or handshakes are attempted and the number of times that the X-Pedition router retries the ping or handshake before considering the server or application to be “down.” You can change these parameters for all servers in a load balancing group or for specific servers.

You can change the ping intervals and the number of retries by entering the following Configure mode commands:.

Set ping interval for all servers in specified group.	**load-balance set group-options** <*group name*> **ping-int** <*seconds*>
Set ping interval for specific server.	**load-balance set server-options** <*ipaddr*> **ping-int** <*seconds*> **port** <*port number*>
Set number of ping retries for all servers in specified group.	**load-balance set group-options** <*group name*> **ping-tries** <*number*>
Set number of ping retries for specific server.	**load-balance set server-options** <*ipaddr*> **ping-tries** <*number*> **port** <*port number*>

You can change the handshake intervals and the number of retries by entering the following Configure mode commands:

Set handshake interval for all servers in specified group.	**load-balance set group-options** <*group name*> **app-int** <*seconds*>
Set handshake interval for specified server.	**load-balance set server-options** <*group name*> **app-int** <*seconds*> **port** <*port number*>
Set number of handshake retries for all servers in specified group.	**load-balance set group-options** <*group name*> **app-tries** <*number*>
Set number of verification retries for specified server.	**load-balance set server-options** <*group name*> **app-tries** <*number*> **port** <*port number*>

Verifying Extended Content

You can also have the X-Pedition router verify the *content* of an application on one or more load balancing servers. For this type of verification, you specify the following:

- A string that the X-Pedition router sends to a single server or to the group of load balancing servers. The string can be a simple HTTP command to get a specific HTML page. Or, it can be a command to execute a user-defined CGI script that tests the operation of the application.
- The reply that the application on each server sends back that the X-Pedition router will use to validate the content. In the case where a specific HTML page is retrieved, the reply can be a string that appears on the page, such as "OK." If a CGI script is executed on the server, it should return a specific response (for example, "OK") that the X-Pedition router can verify.

Application Content Verification (ACV) is a method of ensuring that data coming from your servers remains intact and does not change without your knowledge. ACV can simultaneously protect against server outages, accidental file modification or deletion, and servers whose security has been compromised. By nature, ACV is protocol independent and is designed to work with any type of server that communicates via formatted ASCII text messages, including HTTP, FTP, and SMTP. ACV works by sending a command to your server and searching the response for a certain string. If it finds the string, the server is marked as Up—if not, the server is marked as Down. For example, if you sent the following string to your HTTP server, "**GET /index.html HTTP/1.1\r\nHost: <server IP>\r\n\r\n**", you could search for a response similar to the following (this response would be an ETag in the packet's HTTP header):

```
12345-123a-12345678
```

Another approach is to use the "Last Modified" field of the HTTP header. Because ACV can search for a string in only the first 255 bytes of the response, in most HTTP cases the response will have to be in the packet's HTTP header (i.e., you will not be able to search for a string contained in the web page itself). Some protocols such as FTP or SMTP require users to issue a command to close the session after making the request. This command should appear in the acv-quit field.

Note: A Line Feed character "\n" is appended to these commands when they are sent to the server; therefore, it is not necessary to put a CR or LF in your acv-quit string. For example, when working with FTP, use "BYE" rather than "BYE\n."

You can verify application content by entering the following Configure mode commands:

Specify application verification for all servers in specified group.	**load-balance set group-options** <*group name*> **acv-command** <*command string*> **acv-reply** <*reply string*> **read-till-index** <*number*> [**check-port** <*port-number*>] [**acv-quit** <*quit string*>]
Specify application verification for specified server.	**load-balance set server-options** <*ipaddr*> **port** <*port number*> **acv-command** <*command string*> **acv-reply** <*reply string*> **read-till-index** <*number*> [**check-port** <*port-number*>][**acv-quit** <*quit string*>]

Setting Server Status

It may become necessary at times to prevent new sessions from being directed to one or more load balancing servers. For example, if you need to perform maintenance tasks on a server system, you might want new sessions to temporarily *not* be directed to that server. Setting the status of a server to “down” prevents new sessions from being directed to that server. The “down” status does not affect any current sessions on the server. When the server is again ready to accept new sessions, you can set the server status to “up.”

To set the status of a load balancing server, enter the following command in Enable mode:

Set status of load balancing server.	**load-balance set server-status server-ip** <*ipaddr/range*> **server-port** <*port number*> **group-name** <*group name*> **status up|down**

Load Balancing and FTP

File Transfer Protocol (FTP) packets require special handling with load balancing, because the FTP PORT command packets contain IP address information within the data portion of the packet. If the FTP control port used is not port 21, it is important for the X-Pedition router to know the port number that is used for FTP.

To define an FTP control port (other than port 21) to the load balancing function, enter the following command in Configure mode:

Specify the FTP control port.	**load-balance set ftp-control-port** <*port number*>

Allowing Access to Load Balancing Servers

Load balancing causes both source and destination addresses to be translated on the X-Pedition router. It may be undesirable in some cases for a source address to be translated; for example, when data is to be updated on an individual server. Specified hosts can be allowed to directly access servers in the load balancing group without address translation. Note, however, that such hosts cannot use the virtual IP address and port number to access the load balancing group of servers.

To allow specified hosts to access the load balancing servers without address translation, enter the following command in Configure mode:

Specify the hosts that can access servers without address translation.	**load-balance allow access-to-servers client-ip** <*ipaddr/range*> **group-name** <*group name*>

Setting Timeouts for Load Balancing Mappings

A mapping between a host (source) and a load-balancing server (destination) times out after a certain period. After the mapping times out, any server in the load balancing group can be selected. The default timeouts depend upon the session persistence level configured when the load balance group is created. You can specify the timeout for source-destination load balancing mappings.

To specify the timeout for load balancing mappings, enter the following command in Configure mode:

Specify the timeout for source-destination mappings.	**load-balance set aging-for-src-maps** <*string*> **aging-time** <*timer*>

Displaying Load Balancing Information

To display load balancing information, enter the following commands in Enable mode:

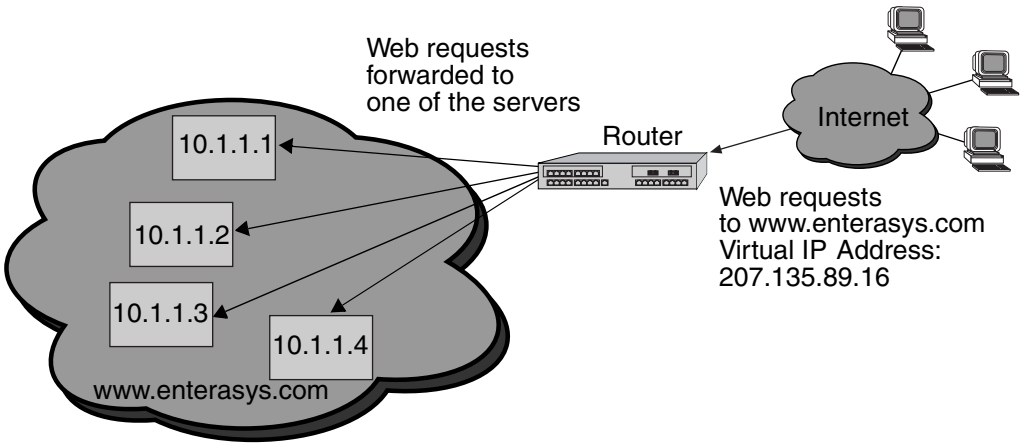
Show the groups of load balancing servers.	**load-balance show virtual-hosts** [**group-name** <*group name*>] [**virtual-ip** <*ipaddr*>] [**virtual-port** <*port number*>]
Show source-destination bindings.	**load-balance show source-mappings** [**client-ip** <*ipaddr/range*>][**virtual-ip** <*ipaddr*>] [**virtual-port** <*port number*>] [**destination-host-ip** <*ipaddr*>]
Show load balancing statistics.	**load-balance show statistics** [**group-name** <*group name*>] [**virtual-ip** <*ipaddr*>] [**virtual-port** <*port number*>]
Show load balance hash table statistics.	l**oad-balance show hash-stats**
Show load balance options for verifying the application.	**load-balance show acv-options** [**group-name** <*group name*>] [**destination-host-ip** <*virtual-ipaddr*>] [**destination-host-port** <*virtual-port-number*>]

Configuration Examples

This section shows examples of load balancing configurations.

Web Hosting with One Virtual Group and Multiple Destination Servers

In the following example, a company web site is established with a URL of www.enterasys.com. The system administrator configures the networks so that the X-Pedition router forwards web requests among four separate servers, as shown below.



Domain Name	Virtual IP	TCP Port	Real Server IP	TCP Port
www.enterasys.com	207.135.89.16	80	10.1.1.1	80
			10.1.1.2	80
			10.1.1.3	80
			10.1.1.4	80

The network shown above can be created with the following load-balance commands:

```
load-balance create group-name enterasys-www virtual-ip 207.135.89.16 virtual-port 80 protocol tcp
load-balance add host-to-group 10.1.1.1-10.1.1.4 group-name enterasys-www port 80
```

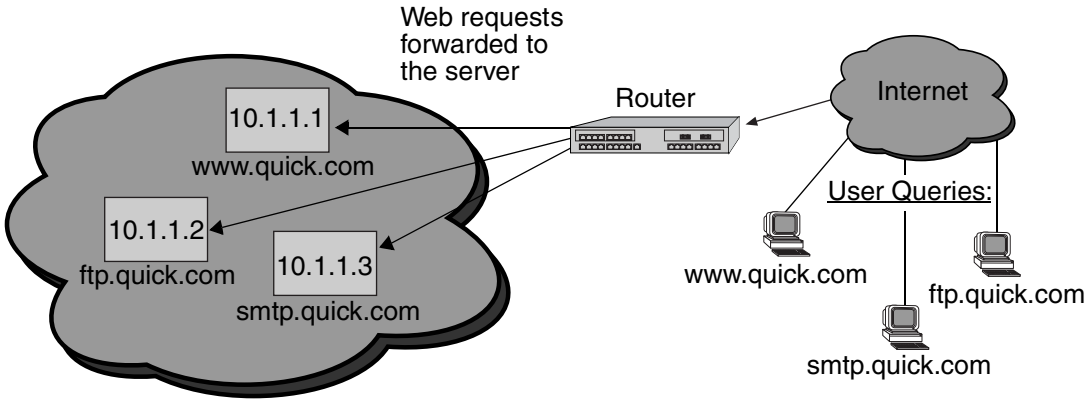
The following is an example of how to configure a simple verification check where the X-Pedition router will issue an HTTP command to retrieve an HTML page and check for the string “OK”:

```
load-balance set group-options enterasys-www acv-command “GET /test.html” acv-reply “OK”
read-till-index 25
```

The **read-till-index** option is not necessary if the file test.html contains “OK” as the first two characters. The **read-till-index** option is helpful if the exact index of the **acv-reply** string in the file is not known to the user. In the above example, the X-Pedition router will search from the beginning of the file up to the 25th character for the start of the string “OK.”

Web Hosting with Multiple Virtual Groups and Multiple Destination Servers

In the following example, three different servers are used to provide different services for a site.



Domain Name	Virtual IP	TCP Port	Real Server IP	TCP Port
www.quick.com	207.135.89.16	80	10.1.1.1	80
ftp.quick.com	207.135.89.16	21	10.1.1.2	21
smtp.quick.com	207.135.89.16	25	10.1.1.3	25

The network shown above can be created with the following load-balance commands:

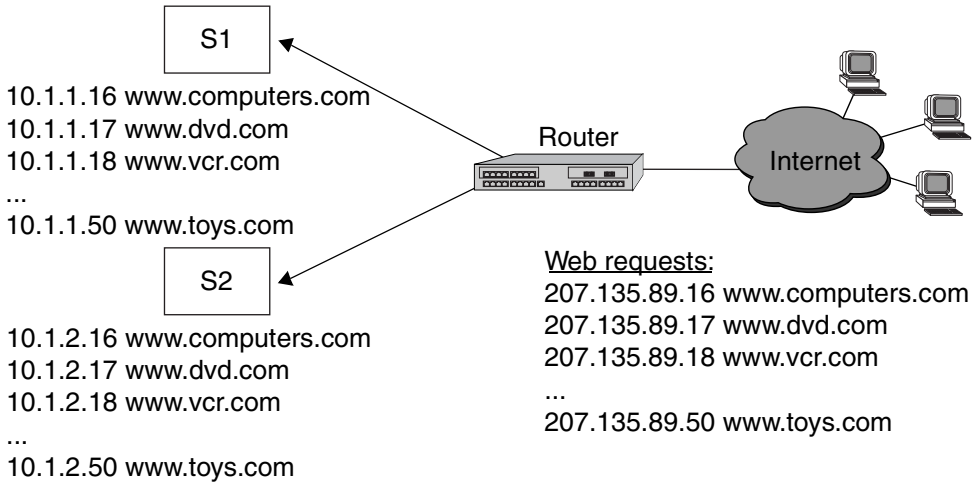
```
load-balance create group-name quick-www virtual-ip 207.135.89.16 virtual-port 80 protocol tcp
load-balance create group-name quick-ftp virtual-ip 207.135.89.16 virtual-port 21 protocol tcp
load-balance create group-name quick-smtp virtual-ip 207.135.89.16 virtual-port 25 protocol tcp
load-balance add host-to-group 10.1.1.1 group-name quick-www port 80
load-balance add host-to-group 10.1.1.2 group-name quick-ftp port 21
load-balance add host-to-group 10.1.1.3 group-name quick-smtp port 25
```

If no application verification options are specified, the X-Pedition router will do a simple TCP handshake to check that the application is “up.” Some applications require specific commands for proper closure of the connection. The following command shows an example of how to send a specific string to close a connection on a server:

```
load-balance set group-options quick-smtp acv-quit “quit”
```

Virtual IP Address Ranges

ISPs who provide web hosting services for their clients require a large number of virtual IP addresses (VIPs). The **load-balance create vip-range-name** and **load-balance add host-to-vip-range** commands were created specifically for this. An ISP can create a range of VIPs for up to an entire class C network with the **load-balance create vip-range-name** command. Once the vip-range is in place, the ISP can then create the corresponding secondary addresses on their destination servers. Once these addresses have been created, the ISP can add these servers to the vip-range with the **load-balance add host-to-vip-range** command. These two commands combined help ISPs take advantage of web servers like Apache which serve different web pages based on the destination address in the http request. The following example illustrates this:



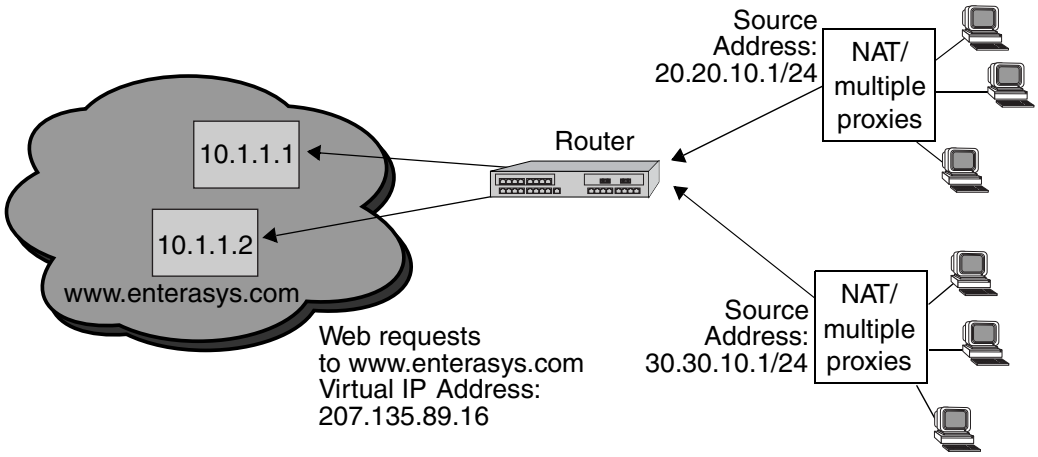
Group Name	Virtual IP	TCP Port	Destination Server IP	TCP Port
www.computers.com	207.135.89.16	80	S1: 10.1.1.16 S2: 10.1.2.16	80
www.dvd.com	207.135.89.17	80	S1: 10.1.1.17 S2: 10.1.2.17	80
www.vcr.com	207.135.89.18	80	S1: 10.1.1.18 S2: 10.1.2.18	80
www.toys.com	207.135.89.50	80	S1: 10.1.1.50 S2: 10.1.2.50	80

The network shown in the previous example can be created with the following load-balance commands:

```
load-balance create vip-range-name mywwwrange 207.135.89.16-207.135.89.50 virtual-port 80 protocol
      tcp
load-balance add host-to-vip-range 10.1.1.16-10.1.1.50 vip-range-name mywwwrange port 80
load-balance add host-to-vip-range 10.1.2.16-10.1.2.50 vip-range-name mywwwrange port 80
```

Session and Netmask Persistence

In the following example, traffic to a company web site (www.enterasys.com) is distributed between two separate servers. In addition, client traffic will have two separate ranges of source IP addresses. The same load balancing server will handle requests from clients of the same source IP subnet address.



Client IP Address	Domain Name	Virtual IP	Real Server IP	TCP Port
20.20.10.1 - 20.20.10.254	www.enterasys.com	207.135.89.16	10.1.1.1	80
30.30.10.1 - 30.30.10.254			10.1.1.2	80

The network shown above can be created with the following load-balance commands:

```
load-balance create group-name enterasys-sec virtual-ip 207.135.89.16 protocol tcp persistence-level ssl virtual-port 443
load-balance add host-to-group 10.1.1.1-10.1.1.2 group-name enterasys-sec port 443
load-balance set client-proxy-subnet enterasys-sec subnet 24
```

Web Caching

Web caching provides a way to store frequently accessed Web objects on a cache of local servers. Each HTTP request is transparently redirected by the X-Pedition router to a configured cache server. When a user first accesses a Web object, that object is stored on a cache server. Each subsequent request for the object uses this cached object. Web caching allows multiple users to access Web objects stored on local servers with a much faster response time than accessing the same objects over a WAN connection. This can also result in substantial cost savings by reducing the WAN bandwidth usage.

Note: The X-Pedition router does not act as cache for web objects; rather, it redirects HTTP requests to local servers on which web objects are cached. Implementing a Web Cache configuration requires users to configure a "routed" network with IP interfaces that allow the router to send requests for the Internet to the correct Web Caching Device.

Configuring Web Caching

The following are the steps in configuring Web caching on the X-Pedition router:

1. Create the cache group (a list of cache servers) to cache Web objects.
2. Specify the hosts whose HTTP requests will be redirected to the cache servers. This step is optional; if you do not explicitly define these hosts, then *all* HTTP requests are redirected.
3. Apply the caching policy to an outbound interface to redirect HTTP traffic on that interface to the cache servers.

Creating the Cache Group

You can specify either a range of contiguous IP addresses or a list of up to four IP addresses to define the servers when the cache group is created. If you specify multiple servers, load balancing is based on the destination address of the request. If any cache server fails, traffic is redirected to the other active servers.

To create the cache group, enter the following command in Configure mode:

Create the cache group.	**web-cache** <*cache-name*> **create server-list** <*server-list-name*> **range** <*ipaddr-range*> | **list** <*ipaddr-list*>

Note: If a range of IP addresses is specified, the range must be contiguous and contain no more than 256 IP addresses.

Specifying the Client(s) for the Cache Group (Optional)

You can explicitly specify the hosts whose HTTP requests are or are not redirected to the cache servers. If you do not explicitly specify these hosts, then *all* HTTP requests are redirected to the cache servers. To specify the clients or non-clients for the cache group, enter the following commands in Configure mode:

Define hosts whose requests are redirected to cache servers.	**web-cache** <*cache-name*> **permit hosts range** <*ipaddr-range*>**|list** <*ipaddr-list*>| **acl** <*acl-name*>
Define hosts whose requests are *not* redirected to cache servers.	**web-cache** <*cache-name*> **deny hosts range** <*ipaddr-range*> **|** **list** <*ipaddr-list*>**|** **acl** <*acl-name*>

Redirecting HTTP Traffic on an Interface

To start the redirection of HTTP requests to the cache servers, you need to apply a caching policy to a specific outbound interface. This interface is typically an interface that connects to the Internet.

Note: By default, the X-Pedition router redirects HTTP requests on port 80. Secure HTTP (https) requests do not run on port 80, therefore these types of requests are not redirected by the router.

To redirect outbound HTTP traffic to the cache servers, enter the following command in Configure mode:

Apply caching policy to outbound interface.	**web-cache** <*cache-name*> **apply interface** <*interface-name*>

Note: The X-Pedition router displays interface names up to 32 characters in length.

Web Cache Application Level “Health Check”

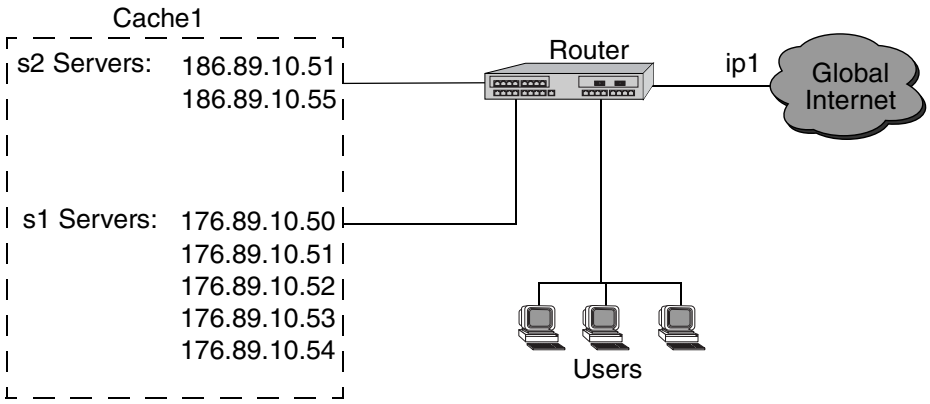
Web Cache server health checking is used to verify the status of one or more servers in a server-list. During a basic health check, the X-Pedition router periodically sends ICMP ping packets to verify the responsiveness of the IP stack operating on the server. An optional secondary health check verifies the server’s TCP responsiveness by opening and closing the TCP port monitored by the server’s web cache software. By default this feature is enabled. To disable this check, use the **web cache** commands described in the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Configuration Options

To configure the timing of ICMP and TCP checks, as well as how many failed attempts represent a connectivity or server failure, use optional TCP checking when the highest level of availability is required.

Configuration Example

In the following example, a cache group of seven local servers is configured to store Web objects for users in the local network:



The following commands configure the cache group ‘cache1’ that contains the servers shown in the figure above and applies the caching policy to the interface ‘ip1’:

```
xp(config)# web-cache cache1 create server-list s1 range “176.89.10.50 176.89.10.54”
xp(config)# web-cache cache1 create server-list s2 list “186.89.10.51 186.89.10.55”
xp(config)# web-cache cache1 apply interface ip1
```

Note that in this example, HTTP requests from *all* hosts in the network are redirected as there are no **web-cache permit** or **web-cache deny** commands.

Other Configurations

This section discusses other commands that may be useful in configuring Web caching in your network.

Note: In the event that you need to implement a Layer-2 configuration, disabling ICMP redirects may work—however, Enterasys does not recommend this design.

Bypassing Cache Servers

Some Web sites require source IP address authentication for user access, therefore HTTP requests for these sites *cannot* be redirected to the cache servers. To specify the sites for which HTTP requests are not redirected to the cache servers, enter the following command in Configure mode:

Define destination sites to which HTTP requests are sent directly.	**web-cache** <*cache-name*> **create bypass-list range** <*ipaddr-range*> | **list** <*ipaddr-list*>| **acl** <*acl-name*>

Proxy Server Redundancy

Some networks use proxy servers that receive HTTP requests on a non-standard port number (i.e., not port 80). When the proxy server is available, all HTTP requests are handled by the proxy server. The X-Pedition router can provide proxy server redundancy by transparently redirecting HTTP connections to the cache servers should the proxy server fail. To achieve this, the router must be configured to redirect HTTP requests on the (non-standard) HTTP port used by the proxy server.

To redirect HTTP requests to a non-standard HTTP port number, enter the following command in Configure mode:

Specify non-standard HTTP port.	**web-cache** <*cache-name*> **set http-port** <*port number*>

Distributing Frequently-Accessed Sites Across Cache Servers

The X-Pedition router uses the destination IP address of the HTTP request to determine which cache server to send the request. However, if there is a Web site that is being accessed very frequently, the cache server serving requests for this destination address may become overloaded with user requests. You can specify that certain destination addresses be distributed across the cache servers in a round-robin manner.

To distribute a specified destination address across cache servers, enter the following command in Configure mode:

Distribute destination address across cache servers.	**web-cache** <*cache-name*> **set round-robin range** <*ipaddr-range*>**|list** <*ipaddr-list*>

Monitoring Web-Caching

To display Web-caching information, enter the following commands in Enable mode.

Show information for all caching policies and all server lists.	**web-cache show all**
Show caching policy information.	**web-cache show cache-name** <*cache-name*> **|all**
Show cache server information.	**web-cache show servers cache** <*cache-name*> **|all**

Chapter 24

IPX Routing Configuration Guide

IPX Routing Overview

The Internetwork Packet Exchange (IPX) is a datagram connectionless protocol for the Novell NetWare environment. You can configure the X-Pedition router for IPX routing and SAP. Routers interconnect different network segments and by definition are network layer devices. Thus routers receive their instructions for forwarding a packet from one segment to another from a network layer protocol. IPX, with the help of RIP and SAP, perform these Network Layer Task. These tasks include addressing, routing, and switching information packets from one location to another on the internetwork.

IPX defines internetwork and intranode addressing schemes. IPX internetwork addressing is based on network numbers assigned to each network segment on a Novell NetWare internetwork. The IPX intranode address comes in the form of socket numbers. Because several processes are normally operating within a node, socket numbers provide a way for each process to distinguish itself.

The IPX packet consists of two parts: a 30-byte header and a data portion. The network node and socket addresses for both the destination and source are held within the IPX header.

RIP (Routing Information Protocol)

IPX routers use RIP to create and dynamically maintain a database of internetwork routing information. RIP allows a router to exchange routing information with a neighboring router. As a router becomes aware of any change in the internetwork layout, this information is immediately broadcast to any neighboring routers. Routers also send periodic RIP broadcast packets containing all routing information known to the router.

Note: The X-Pedition router supports a maximum of 120 RIP interfaces.

The X-Pedition router uses IPX RIP to create and maintain a database of internetwork routing information. The router's implementation of RIP allows the following exchanges of information:

- Workstations locate the fastest route to a network number by broadcasting a route request.
- Routers request routing information from other routers to update their own internal tables by broadcasting a route request.
- Routers respond to route requests from workstations and other routers.
- Routers perform periodic broadcasts to make sure that all other routers are aware of the internetwork configuration.
- Routers perform broadcasting whenever they detect a change in the internetwork configurations.

The X-Pedition router's RIP implementation follows the guidelines given in Novell's *IPX RIP and SAP Router Specification Version 1.30* document.

SAP (Service Advertising Protocol)

SAP provides routers with a means of exchanging internetwork service information. Through SAP, servers advertise their services and addresses. Routers gather this information and share it with other routers. This allows routers to create and dynamically maintain a database of internetwork service information. SAP allows a router to exchange information with a neighboring SAP agent. As a router becomes aware of any change in the internetwork server layout, this information is immediately broadcast to any neighboring SAP agents. SAP broadcast packets containing all server information known to the router are also sent periodically.

The X-Pedition router uses IPX SAP to create and maintain a database of internetwork service information. The router's implementation of SAP allows the following exchanges of information:

- Workstations locate the name and address of the nearest server of certain type
- Routers request the names and addresses of either all or certain type of servers
- Servers respond to the workstation's or router's request
- Routers make periodic broadcasts to make sure all other routers are aware of the internetwork configuration
- Routers perform broadcasting whenever they detect a change in the internetwork configurations

Configuring IPX RIP & SAP

This section provides an overview of configuring various IPX parameters and setting up IPX interfaces.

IPX RIP

On the X-Pedition router, RIP automatically runs on all IPX interfaces. The router will keep multiple routes to the same network having the lowest ticks and hop count. Static routes can be configured on the router using the **ipx add route** command. Through the use of RIP filters, the router can control the acceptance and advertisement of networks per-interface.

Note: The X-Pedition router supports a maximum of 120 RIP interfaces.

IPX SAP

On the X-Pedition router, SAP automatically runs on all the IPX interfaces. The X-Pedition router will keep multiple SAPs having the lowest hop count. Static SAPs can be configured on the router using the CLI's **ipx add sap** command. Through the use of SAP filters, the router can control the acceptance and advertisements of services per-interface.

Creating IPX Interfaces

When you create IPX interfaces on the X-Pedition router, you provide information about the interface (such as its name, output MAC encapsulation, and IPX address). You also enable or disable the interface and bind the interface to a single port or VLAN.

Note: The X-Pedition router supports a maximum of 64 IPX interfaces. The router displays interface names up to 32 characters in length.

Note: Interfaces bound to a single port go down when the port goes down but interfaces bound to a VLAN remain up as long as at least one port in that VLAN remains active.

The procedure for creating an IPX interface depends on whether you are binding that interface to a single port or a VLAN. Separate discussions on the different procedures follow.

Note: You cannot assign IPX interfaces for LAN and WAN to the same VLAN. In order for these two types of IPX interfaces to coexist on the X-Pedition router, each type must be assigned to a different VLAN.

IPX Addresses

The IPX address is a 12-byte number divided into three parts. The first part is the 4-byte (8-character) IPX external network number. The second part is the 6-byte (12-character) node number. The third part is the 2-byte (4-character) socket number.

Configuring IPX Interfaces and Parameters

This section provides an overview of configuring various IPX parameters and setting up IPX interfaces.

Configuring IPX Addresses to Ports

You can configure one IPX interface directly to a physical port. To configure an IPX interface to a port, enter the following command in Configure mode:

Configure an IPX interface to a physical port.	**interface create ipx** *<InterfaceName>* **address-mask** *<ipxAddr-mask>* **port** *<port>*

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring Secondary Addresses on an IPX Interface

You can configure secondary addresses on an IPX interface.

To configure a secondary address on an IPX interface, enter the following command in Configure mode:

Add a secondary address and encapsulation type to an existing IPX interface.	**interface add ipx** *<Interface Name>* **address** *<IPX-network-address>* **output-mac-encapsulation** *<encapsulation-type>*

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring IPX Interfaces for a VLAN

You can configure one IPX interface per VLAN. To configure a VLAN with an IPX interface, enter the following command in Configure mode:

Create an IPX interface for a VLAN.	**interface create ipx** *<InterfaceName>* **address-mask** *<ipxAddr-mask>* **vlan** *<name>*

Note: A VLAN containing both LAN and WAN ports will not support interfaces configured with IPX. The X-Pedition router displays interface names up to 32 characters in length.

Specifying IPX Encapsulation Method

The X-Pedition router supports four encapsulation types for IPX. You can configure encapsulation type on a per-interface basis.

- Ethernet II: The standard ARPA Ethernet Version 2.0 encapsulation, which uses a 16-bit protocol type code (the default encapsulation method)
- 802.3 SNAP: SNAP IEEE 802.3 encapsulation, in which the type code becomes the frame length for the IEEE 802.2 LLC encapsulation (destination and source Service Access Points, and a control byte)

Note: When using line cards introduced prior to the "AA" series, SNA/DLC/NetBIOS traffic may not bridge properly. The issue in bridging DLC packets occurs where the length field within an IEEE 802.3 frame indicates less than 46 bytes of data.

The X-Pedition router removes the length field information of incoming IEEE 802.3, 802.2, and Ethernet SNAP packets, then recalculates the field prior to re-transmission. Consequently, the calculation is based on the length of the entire data field. A packet entering the router whose length field indicates a data field of less than 46 bytes will exit with the length field recalculated incorrectly. This can be a problem with LLC2 and legacy IPX applications. Typically, such packets exist only in SNA and NetBIOS/NetBEUI environments.

- 802.3: 802.3 encapsulation method used within Novell IPX environments
- 802.2: 802.2 encapsulation method used within Novell IPX environments

To configure IPX encapsulation, enter the following commands in Configure mode:

Configure Ethernet II encapsulation.	**interface create ipx** <*Interface Name*> **output-mac-encapsulation ethernet_II**
Configure 802.3 SNAP encapsulation.	**interface create ipx** <*Interface Name*> **output-mac-encapsulation ethernet_snap**
Configure 802.3 IPX encapsulation.	**interface create ipx** <*Interface Name*> **output-mac-encapsulation ethernet_802.3**
Configure 802.2 IPX encapsulation.	**interface create ipx** <*Interface Name*> **output-mac-encapsulation ethernet_802.2_ipx**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuring IPX Routing

By default, IPX routing is enabled on the X-Pedition router.

Enabling IPX RIP

IPX RIP is enabled by default on the X-Pedition router. You must first create an IPX interface or assign an IPX interface to a VLAN before RIP will start learning routes.

Note: The X-Pedition router supports a maximum of 120 RIP interfaces.

Enabling SAP

IPX SAP is enabled by default on the X-Pedition router. You must first create an IPX interface or assign an IPX interface to a VLAN before SAP will start learning services.

Configuring Static Routes

In a Novell NetWare network, the X-Pedition router uses RIP to determine the best paths for routing IPX. However, you can add static RIP routes to RIP routing table to explicitly specify a route.

To add a static RIP route, enter the following command in Configure mode:

Add a static RIP route.	**ipx add route** *<networkaddr> <nextrouter or network node> <metric> <ticks>*

Configuring Static SAP Table Entries

Servers in an IPX network use SAP to advertise services via broadcast packets. Services from servers are stored in the Server Information Table. If you want to have a service explicitly advertised with different hops, you will need to configure a static entry.

To add an entry into the Server Information Table, enter the following command in Configure mode:

Add a SAP table entry.	**ipx add sap** *<service type> <SrvcName> <node> <socket> <metric> <interface-network>*

Controlling Access to IPX Networks

To control access to IPX networks, you create access control lists and then apply them with filters to individual interfaces. The X-Pedition router supports the following IPX access lists that you can use to filter various kinds of traffic:

Note: You may not apply ACLs to interface EN0 of the control module.

- IPX access control list: Restrict traffic based on the source address, destination address, source socket, destination socket, source network mask or destination network mask.
- SAP access control list: Restricts advertisements or learning of SAP services. These lists are used for SAP filters. They can also be used for Get Nearest Server (GNS) replies.
- RIP access control list: Restricts advertisements or learning of networks.

Creating an IPX Access Control List

IPX access control lists control which IPX traffic is received from or sent to an interface based on source address, destination address, source socket, destination socket, source network mask or destination network mask. This is used to permit or deny traffic from one IPX end node to another.

To create an IPX access control list, perform the following task in the Configure mode:

Create an IPX access control list.	**acl** <*name*> **permit**|**deny ipx** <*SrcAddr*> <*SrcSocket*> <*DstAddr*> <*DstSocket*> <*SrcNetMask*> <*DstNetMask*>

Once an IPX access control list has been created, you must apply the access control list to an IPX interface. To apply an IPX access control list, enter the following command in Configure mode:

Apply an IPX access control list.	**acl** <*name*> **apply interface** <*Interface Name*> **input**|**output** [**logging** [**on**|**off**]]

Note: The X-Pedition router displays interface names up to 32 characters in length.

Creating an IPX Type 20 Access Control List

IPX type 20 access control lists control the forwarding of IPX type 20 packets. To create an IPX type 20 access control list, enter the following command in Configure mode:

Create an IPX type 20 access control list.	**acl** <*name*> **permit**|**deny ipxtype20**

Creating an IPX SAP Access Control List

IPX SAP access control lists control which SAP services are available on a server. To create an IPX SAP access control list, enter the following command in Configure mode:

Create an IPX SAP access control list.	**acl** <*name*> **permit**|**deny ipxsap** <*ServerNetworkNode*> <*ServiceType*> <*ServiceName*>

Once an IPX SAP access control list has been created, you must apply the access control list to an IPX interface. To apply an IPX SAP access control list, enter the following command in Configure mode:

Apply an IPX SAP access control list.	**acl** <*name*> **apply interface** <*InterfaceName*> **input**|**output** [**logging** [**on**|**off**]]

Note: The X-Pedition router displays interface names up to 32 characters in length.

Creating an IPX GNS Access Control List

IPX GNS access control lists control which SAP services the X-Pedition router can reply with to a get nearest server (GNS) request. To create an IPX GNS access control list, enter the following command in Configure mode:

Create an IPX GNS access control list.	**acl** <*name*> **permit**|**deny ipxgns** <*ServerNetworkNode*> <*ServiceType*> <*ServiceName*>

Once an IPX GNS access control list has been created, you must apply the access control list to an IPX interface. To apply an IPX GNS access control list, enter the following command in Configure mode:

Apply an IPX GNS access control list.	**acl** <*name*> **apply interface** <*InterfaceName*> **output [logging [on**|**off]]**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Creating an IPX RIP Access Control List

IPX RIP access control lists control which RIP updates are allowed. To create an IPX RIP access control list, perform the following task in the Configure mode:

Create an IPX RIP access control list.	**acl** *<name>* **permit|deny ipxrip** *<FromNetwork> <ToNetwork>*

Once an IPX RIP access control list has been created, you must apply the access control list to an IPX interface. To apply an IPX RIP access control list, enter the following command in Configure mode:

Apply an IPX RIP access control list.	**acl** *<name>* **apply interface** *<Interface Name>* **input|output** [**logging [on|off**]]

Note: The X-Pedition router displays interface names up to 32 characters in length.

Monitoring an IPX Network

The X-Pedition router reports IPX interface information and RIP or SAP routing information.

To display IPX information, enter the following command in Enable mode:

Show a RIP entry in the IPX RIP table.	**ipx find rip** *<DstNetwork>*
Show a SAP entry in the IPX SAP table.	**ipx find sap** *<type> <ServiceType> <ServiceName> <ServerNetwork>*
Show IPX interface information.	**ipx show interfaces** *<interface-name>*
Show IPX RIP table.	**ipx show tables rip**
Show IPX routing table.	**ipx show tables routing**
Show IPX SAP table.	**ipx show tables sap**
Show IPX RIP/SAP table summary.	**ipx show tables summary**

Note: The X-Pedition router displays interface names up to 32 characters in length.

Configuration Example

This example performs the following configuration:

- Creates IPX interfaces

Note: The X-Pedition router supports a maximum of 64 IPX interfaces.

- Adds static RIP routes
- Adds static SAP entries
- Adds a RIP access list
- Adds a SAP access list
- Adds a GNS access list

```
! Create interface ipx1 with ipx address AAAAAAAA
interface create ipx ipx1 address AAAAAAAA port et.1.1 output-mac-encapsulation ethernet_802.2_IPX
!
! Create interface ipx2 with ipx address BBBBBBBB
interface create ipx ipx2 address BBBBBBBB port et.1.2 output-mac-encapsulation ethernet_802.3
!
! Add secondary address on interface ipx2 with ipx address CCCCCCCC
interface add ipx ipx2 address CCCCCCCC output-mac-encapsulation ethernet_II
!
!Add static route to network 9
ipx add route 9 BBBBBBBB.01:02:03:04:05:06 1 1
!
!Add static sap
ipx add sap 0004 FILESERVER1 9.03:04:05:06:07:08 452 1 AAAAAAAA
!
!RIP Access List
acl 100 deny ipxrip 1 2
!
!RIP inbound filter
acl 100 apply interface ipx1 input
!
!SAP Access List
acl 200 deny ipxsap A.01:03:05:07:02:03 0004 FILESERVER2
!
!SAP outbound filter to interface ipx2
acl 200 apply interface ipx2 output
!
!IPX type 20 access list
acl 300 deny ipxtype20
!
!IPX type 20 inbound filter to interface ipx2
acl 300 apply interface ipx2 input
!
!GNS Access List
acl 300 deny ipxgns A.01:03:05:07:02:03 0004 FILESERVER2
acl 200 apply interface ipx2 output
```

Chapter 25

Access Control List Configuration Guide

This chapter explains how to configure and use Access Control Lists (ACLs) on the X-Pedition router. ACLs are lists of selection criteria for specific types of packets. When used in conjunction with certain X-Pedition functions, ACLs allow you to restrict Layer-3/4 traffic going through the router.

This chapter contains the following sections:

- *ACL Basics* on page 420 explains how ACLs are defined and how the X-Pedition router evaluates them.
- *Creating ACLs* on page 425 describes how to create new ACLs and ACL rules.
- *Applying ACLs* on page 426 describes the different kinds of ACLs: Interface ACLs, Service ACLs, Layer-4 Bridging ACLs, and Profile ACLs, and gives examples of their usage.

Note: You may not apply ACLs to interface EN0 of the control module.

- *Modifying ACLs* on page 434 explains how to edit ACLs remotely or by using the X-Pedition router's built-in ACL Editor function.
- *Enabling ACL Logging* on page 436 explains how to log information about packets that are permitted or denied because of an ACL.
- *Monitoring ACLs* on page 437 lists the commands you can use to display information about ACLs active on the X-Pedition router.

ACL Basics

An ACL consists of one or more *rules* describing a particular type of IP or IPX traffic. ACLs can be simple, consisting of only one rule, or complicated with many rules. Each rule tells the X-Pedition router to either permit or deny packets that match selection criteria specified in the rule.

Each ACL is identified by a name. The name can be a meaningful string, such as *denyftp* or *noweb* or it can be a number such as *100* or *101*. For example, the following ACL has a rule that permits all IP packets from subnet 10.2.0.0/16 to go through the X-Pedition router:

```
acl 101 permit ip 10.2.0.0/16
```

Defining Selection Criteria in ACL Rules

Selection criteria in the rule describe characteristics about a packet. In the example above, the selection criteria are IP packets from 10.2.0.0/16. The selection criteria you can specify in an ACL rule depends on the type of ACL you are creating. For IPv4, TCP, and UDP ACLs, you may specify the following selection criteria.

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Type of Service (TOS)

For IPv6 ACLs, you may specify the following selection criteria:

- Source IPv6 address/prefix length
- Destination IPv6 address/prefix length
- Traffic Class field value
- Next Header field value

For IPX ACLs, you may specify the following selection criteria:

- Source network address
- Destination network address
- Source IPX socket
- Destination IPX socket

These selection criteria are specified as *fields* of an ACL rule. The following syntax description shows the fields of an IPv4 ACL rule:

acl <*name*> **permit|deny ip** <*SrcAddr/Mask*> <*DstAddr/Mask*> <*SrcPort*> <*DstPort*> <*tos*> <*tos-mask*> [**accounting**]

Note: The **acl permit|deny ip** command restricts traffic for all IP-based protocols, such as TCP, UDP, ICMP, and IGMP. Variants of the a**cl permit|deny ip** command exist that allow you to restrict traffic for a specific IP-based protocol; for example, the **acl permit|deny tcp command** lets you restrict only TCP traffic. These variants have the same syntax and fields as the **acl permit|deny ip** command.

The following syntax description shows the fields of an IPv6 ACL rule:

acl <*name*> [**permit|deny**] **ipv6** <*SrcIPv6Addr/PrefixLen*> <*DstIPv6Addr/PrefixLen*> <*tclass*> <*nheader*>

Note: Only ACLs created with the **acl permit|deny ipv6** command can be applied to IPv6 interfaces. Attempting to apply ACLs created with any other commands will generate an error.

The following syntax description shows the fields of an IPX ACL rule:

acl <*name*> **permit|deny ipx** <*SrcAddr*> <*SrcSocket*> <*DstAddr*> <*DstSocket*> <*SrcNetMask*> <*DstNetMask*>

Each field in an ACL rule is position sensitive. For example, for a TCP traffic rule, the source address must be followed by the destination address, the source socket, the destination socket, and so on.

In most cases, not all fields of an ACL rule need to be specified. If you do not specify a particular field, the X-Pedition router treats the field as a wildcard or “don't care” condition. However, if a field is specified, that particular field will be matched against the packet. Each protocol can have a number of different fields to match. For example, a rule for TCP can use socket port numbers, while a rule for IPX can use a network node address.

Note: IPv6 and IPX ACLs require that *all* fields be specified, either with a specific value or with the **any** parameter.

Since each field is position sensitive, it may be necessary to “skip” some fields in order to specify a value for another field. To skip a field, use the keyword **any**. For example, the following ACL rule denies SMTP traffic between any two hosts:

acl nosmtp deny tcp any any smtp smtp

Note that in the above example, the <*tos*> (Type of Service) field is not specified and is treated as a wildcard. The **any** keyword is needed only to skip a wildcard field in order to explicitly specify another field that is further down in the rule. If there are no other fields to specify, the **any** keyword is not necessary. For example, the following ACL permits all IP traffic to go through:

```
acl yesip permit ip
```

How ACL Rules are Evaluated

For an ACL with multiple rules, the ordering of the rules is important. When the X-Pedition router checks a packet against an ACL, it goes through each rule in the ACL sequentially. If a packet matches a rule, it is forwarded or dropped based on the **permit** or **deny** keyword in the rule. All subsequent rules are ignored. That is, a first-match algorithm is used. There is no hidden or implied ordering of ACL rules, nor is there precedence attached to each field. The X-Pedition router simply goes down the list, one rule at a time, until there is a match. Consequently, rules that are more specific (that is, with more selection criteria) should always be listed ahead of rules that are less specific. For example, the following ACL permits all TCP traffic except those from subnet 10.2.0.0/16:

```
acl 101 deny tcp 10.2.0.0/16 any any any
acl 101 permit tcp any any any any
```

When a TCP packet comes from subnet 10.2.0.0/16, it finds a match with the first rule. This causes the packet to be dropped. A TCP packet coming from other subnets does not match the first rule. Instead, it matches the second rule, which allows the packet to go through.

If you were to reverse the order of the two rules:

```
acl 101 permit tcp any any any any
acl 101 deny tcp 10.2.0.0/16 any any any
```

all TCP packets would be allowed to go through, including traffic from subnet 10.2.0.0/16. This is because TCP traffic coming from 10.2.0.0/16 would match the first rule and be allowed to go through. The second rule would not be looked at since the first match determines the action taken on the packet.

Implicit Deny Rule

At the end of each ACL, the system automatically appends an *implicit deny rule*. This implicit deny rule denies all traffic. For a packet that doesn't match any of the user-specified rules, the implicit deny rule acts as a catch-all rule. All packets match this rule.

This is done for security reasons. If an ACL is misconfigured, and a packet that should be allowed to go through is blocked because of the implicit deny rule, the worst that could happen is inconvenience. On the other hand, if a packet that should not be allowed to go through is instead sent through, there is now a security breach. Thus, the implicit deny rule serves as a line of defense against accidental misconfiguration of ACLs.

To illustrate how the implicit deny rule is used, consider the following ACL:

```
acl 101 permit ip 1.2.3.4/24
acl 101 permit ip 4.3.2.1/24 any nntp
```

With the implicit deny rule, this ACL actually has three rules:

```
acl 101 permit ip 1.2.3.4/24 any any any
acl 101 permit ip 4.3.2.1/24 any nntp any
acl 101 deny any any any any any
```

If a packet comes in and doesn't match the first two rules, the packet is dropped. This is because the third rule (the implicit deny rule) matches all packets.

Although the implicit deny rule may seem obvious in the above example, this is not always the case. For example, consider the following ACL rule:

```
acl 102 deny ip 10.1.20.0/24 any any any
```

If a packet comes in from a network other than 10.1.20.0/24, you might expect the packet to go through because it doesn't match the first rule. However, that is not the case because of the implicit deny rule. With the implicit deny rule attached, the rule looks like this:

```
acl 102 deny ip 10.1.20.0/24 any any any
acl 102 deny any any any any any
```

A packet coming from 10.1.20.0/24 would not match the first rule, but would match the implicit deny rule. As a result, no packets would be allowed to go through. The first rule is simply a subset of the second rule. To allow packets from subnets other than 10.1.20.0/24 to go through, you would have to explicitly define a rule to permit other packets to go through. To correct the previous example and let packets from other subnets enter the X-Pedition router, you must add a new rule to permit packets to go through:

```
acl 101 deny ip 10.1.20.0/24 any any any
acl 101 permit ip
acl 101 deny any any any any any
```

The second rule forwards all packets that are not denied by the first rule.

Because of the implicit deny rule, an ACL works similarly to a firewall that is elected to deny all traffic. You create ACL rules that punch “holes” into the firewall to permit specific types of traffic; for example, traffic from a specific subnet or traffic from a specific application.

Allowing External Responses to Established TCP Connections

Typically organizations that are connected to the outside world implement ACLs to deny access to the internal network. If an internal user wishes to connect to the outside world, the request is sent; however any incoming replies may be denied because ACLs prevent them from going through. To allow external responses to internally generated requests, you would have to create an ACL to allow responses from each specific outside host. If the number of outside hosts that internal users need to access is large or changes frequently, this can be difficult to maintain.

To address this problem, the X-Pedition router can be configured to accept outside TCP responses into the internal network, provided that the TCP connection was initiated internally. Otherwise, it will be rejected. To do this, enter the following command in Configure Mode:

Allow TCP responses from external hosts, provided the connection was established internally.	**acl** <*name*> **permit tcp established**

The following ACL illustrates this feature:

```
acl 101 permit tcp established
acl 101 apply interface int1 input
```

Any incoming TCP packet on interface int1 is examined, and if the packet is in response to an internal request, it is permitted; otherwise, it is rejected. Note that the ACL contains no restriction for outgoing packets on interface int1, since internal hosts are allowed to access the outside world.

Note: This feature does not apply to IPv6 interfaces.

Creating ACLs

To create a new ACL and apply its first rule, you must enter the corresponding ACL command from the CLI. This creates a new ACL and allows you to begin applying ACL rules. To create an ACL to permit IP traffic from the subnet 10.1.0.0 (with a 16 bit netmask) to any destination, enter the following from the CLI:

```
xp(config)# acl 101 permit ip 10.1.0.0/16 any
```

Note: You may not apply ACLs to interface EN0 of the control module.

ACL rules are checks that the router uses to verify that each packet it receives is permitted on the system. When you apply an ACL to an interface, the X-Pedition router appends an *implicit deny rule* to that ACL. The implicit deny rule denies ***all*** traffic. If you intend to allow all traffic that doesn't match your specified ACL rules to go through, you must *explicitly* define a rule to permit all traffic. Packets that do not match any of the criteria you specify are rejected from the system.

The following command creates an ACL rule to deny any incoming TCP or UDP traffic coming from a privileged port (less than 1024). If the incoming traffic is not TCP or UDP, then the X-Pedition router check only the source and destination addresses, not the port number. Therefore, this ACL will deny all non-TCP and non-UDP traffic.

```
xp(config)# acl 101 deny ip any any 1-1024 any
```

To create an additional ACL rule to permit Telnet traffic (port 23) from the host 10.23.4.8 to the subnet 10.2.3.0:

```
xp(config)# acl 101 permit ip 10.23.4.8 10.2.3.0/24
```

Note: The rules within an ACL must belong to the same protocol family.

To activate the ACL, you must apply it using the apply option of the ACL command used. This enables the ACL and causes the router to check all received packets to see if they are valid packets that may travel through the network. The following example demonstrates how to apply the ACL defined above:

```
xp(config)# acl 101 apply interface all-ip
```

Note: You must reapply or comment in the apply line of the ACL before changes will take affect.

In-line Editing

The X-Pedition router allows you to manage ACLs from the configuration by negating command lines. By negating the ACL **apply** line from the configuration, you may completely turn off an ACL. Negating a specific rule within the ACL will remove it from the ACL's rule list.

Wildcards

The following command creates an ACL to permit all IP traffic. Since none of the ACL fields are specified, they are all assumed to be wildcards.

```
xp(config)# acl allip permit ip
```

The above command is equivalent to the following:

```
xp(config)# acl allip permit ip any any any any
```

Applying ACLs

It is important to understand that an ACL is simply a definition of packet characteristics specified in a set of rules. An ACL must be *enabled* in one of the following ways:

- Applying an ACL to an interface, which permits or denies traffic to or from the X-Pedition router. ACLs used in this way are known as *Interface ACLs*.

Note: Only Interface ACLs can be applied to IPv6 interfaces.

- Applying an ACL to a service, which permits or denies access to system services provided by the X-Pedition router. ACLs used in this way are known as *Service ACLs*.
- Applying an ACL to ports operating in Layer-4 bridging mode, which permits or denies bridged traffic. ACLs used in this way are known as *Layer-4 Bridging ACLs*.
- Associating an ACL with **ip-policy**, **nat**, **port mirroring**, **rate-limit**, or **web-cache** commands, which specifies the criteria that packets, addresses, or flows must meet in order to be relevant to these X-Pedition features. ACLs used in this way are known as *Profile ACLs*.

These uses of ACLs are described in the following sections.

Applying ACLs to Interfaces

An ACL can be applied to an interface to examine either inbound or outbound traffic. Inbound traffic is traffic coming into the X-Pedition router. Outbound traffic is traffic going out of the router. For each interface, only one ACL can be applied for the same protocol in the same direction. For example, you cannot apply two or more IP ACLs to the same interface in the inbound direction. You can apply two ACLs to the same interface if one is for inbound traffic and one is for outbound traffic, but not in the same direction. However, this restriction does not prevent you from specifying many rules in an ACL. You have to put all of these rules into one ACL and apply it to an interface.

Note: You may not apply ACLs to interface EN0 of the control module.

When a packet comes into the X-Pedition router at an interface where an inbound ACL is applied, the router compares the packet to the rules specified by that ACL. If it is permitted, the packet is allowed into the router. If not, the packet is dropped. If that packet is to be forwarded to go out of another interface (that is, the packet is to be routed) then a second ACL check is possible. At the output interface, if an outbound ACL is applied, the packet will be compared to the rules specified in this outbound ACL. Consequently, it is possible for a packet to go through two separate checks, once at the inbound interface and once more at the outbound interface.

When you apply an ACL to an interface, you can also specify whether the ACL can be modified or removed from the interface by an external agent (such as the Policy Manager application). Note that for an external agent to modify or remove an applied ACL from an interface, the **acl-policy enable external** command must be in the configuration.

In general, you should try to apply ACLs at the inbound interfaces instead of the outbound interfaces. If a packet is to be denied, you want to drop the packet as early as possible, at the inbound interface. Otherwise, the X-Pedition router will have to process the packet, determine where the packet should go only to find out that the packet should be dropped at the outbound interface. In some cases, however, it may not be simple or possible for the administrator to know ahead of time that a packet should be dropped at the inbound interface. Nonetheless, for performance reasons, whenever possible, you should create and apply an ACL to the inbound interface.

You can use the **all-ip**, **all-ipv6**, or **all-ipv6in4-tunnels** options to apply an ACL to all the router's IPv4, IPv6 port-based, or IPv6-in-IPv4 tunnel interfaces at one time. However, if you then apply a different ACL to a specific interface, the ACL applied directly to the specific interface will replace any ACL applied to all IP interfaces, for that interface. If you subsequently remove or un-apply the specifically-applied ACL, the **all-ip/all-ipv6/all-ipv6in4-tunnels** ACL will be automatically applied to that interface again.

To apply an ACL to an interface, enter the following command in Configure mode:

Apply ACL to an interface.	**acl** <*name*> **apply interface** <*interface name*> **input|output [logging on|off|deny-only|permit-only] [report-denied periodic| all] [policy local|external] [deny-trap]**

Note: When applying IPv6 ACLs to IPv6 interfaces, you should be aware of the following limitations:

- IPv6 ACLs do not support the logging, reporting, or policy options of the **apply** command.
- IPv6 ACLs can be applied only to inbound traffic, with the **input** direction option.

ACLs and IPv6-in-IPv4 Tunnel Interfaces

IPv6-in-IPv4 tunnels are used for tunneling IPv6 datagrams over an IPv4 routing infrastructure, by encapsulating IPv6 packets within IPv4 packets. An IPv6-in-IPv4 tunnel interface is a virtual IPv6 interface that is configured with two IPv4 interfaces as tunnel endpoints, one local and one on the remote system at the other end of the tunnel. IPv4 ACLs can be applied to the IPv4 interfaces that serve as tunnel endpoints, and IPv6 ACLs can be applied to the tunnel IPv6 interfaces.

When an IPv4 / IPv6-enabled X-Pedition router receives an IPv4 packet with an IPv4 destination address belonging to itself and with a protocol type of 41 in the IPv4 header, any existing IPv4 ACLs will be applied to the packet. If the packet is allowed, it will be decapsulated and the IPv4 header discarded. Any existing IPv6 ACLs will then be applied to the packet, and if the packet is allowed, it will be forwarded using the IPv6 Forwarding Information Base.

Applying ACLs to Services

ACLs can also be created to permit or deny access to system services provided by the X-Pedition router; for example, HTTP or Telnet servers. This type of ACL is known as a *Service* ACL. By definition, a Service ACL is for controlling inbound packets to a service on specific interfaces on the router. For example, on a particular interface, you can grant Telnet server access from a few specific hosts or deny Web server access from a particular subnet. It is true that you can do the same thing with ordinary ACLs and apply them to specific interfaces. However, the Service ACL is created specifically to control access to some of the services on specified interfaces of the router. As a result, only inbound traffic to the router is checked.

Note: If a service does not have an ACL applied, that service is accessible to everyone. To control access to a service, an ACL must be used.

To apply an ACL to a service, enter the following command in Configure mode:

Apply ACL to a service.	**acl** <*name*> **apply service** <*service name*> [**logging** [**on**|**off**]]

Note: IPv6 ACLs cannot be applied to services.

Applying ACLs to Layer-4 Bridging Ports

ACLs can also be created to permit or deny access to one or more ports operating in Layer-4 bridging mode. Traffic that is switched at Layer 2 through the X-Pedition router can have ACLs applied on the Layer 3/4 information contained in the packet. The ACLs that are applied to Layer-4 Bridging ports are only used with bridged traffic. The ACLs that are applied to the interface are still used for routed traffic.

Like ACLs that are applied to interfaces, ACLs that are applied to Layer 4 bridging ports can be applied to either inbound or outbound traffic. For each port, only one ACL can be applied for the inbound direction and one for the outbound direction. You can apply two ACLs to the same port if one is for inbound traffic and one is for outbound traffic.

To apply an ACL to a port, enter the following command in Configure Mode:

Apply a Layer-4 bridging ACL to a port	**acl** <*name*> **apply port** <*port-list*>

See *Layer-4 Bridging and Filtering* on page 479 for information on configuring Layer-4 Bridging on the X-Pedition router.

Note: IPv6 ACLs cannot be applied to ports.

Using ACLs as Profiles

You can use the **acl** command to define a *profile*. A profile specifies the criteria that addresses, flows, hosts, or packets must meet to be relevant to certain X-Pedition features. Once you have defined an ACL profile, you can use the profile with the configuration command for that feature. For example, the Network Address Translation (NAT) feature on the router allows you to create address pools for dynamic bindings. You use ACL profiles to represent the appropriate pools of IP addresses. The following X-Pedition router features use ACL profiles:

Feature	ACL Profile Usage
IP policy	Specifies the packets that are subject to the IP routing policy.
Dynamic NAT	Defines local address pools for dynamic bindings.
Port mirroring	Defines traffic to be mirrored. **Note:** The X-Pedition router supports port mirroring, ACL, and Layer-2 filtering on a per-WAN-card basis, not a per-port basis.
Rate limiting	Specifies the incoming traffic flow to which rate limiting is applied.
Web caching	Specifies which HTTP traffic should always (or never) be redirected to the cache servers. Specifies characteristics of Web objects that should not be cached.

Note the following about using Profile ACLs:

- Only IPv4, ACLs can be used as Profile ACLs. ACLs for IPv6 and non-IP protocols *cannot* be used as Profile ACLs.
- The **permit**/**deny** keywords, while required in the ACL rule definition, are *disregarded* in the configuration commands for the above-mentioned features. In other words, the configuration commands will act upon a specified Profile ACL whether or not the Profile ACL rule contains the **permit** or **deny** keyword.
- Unlike with other kinds of ACLs, there is no implicit deny rule for Profile ACLs.
- Only certain ACL rule parameters are relevant for each configuration command. For example, the configuration command to create NAT address pools for dynamic bindings (the **nat create dynamic** command) only looks at the source IP address in the specified ACL rule. The destination IP address, ports, and TOS parameters, if specified, are ignored.

Specific usage of Profile ACLs is described in more detail in the following sections.

Using Profile ACLs with the IP Policy Facility

The IP policy facility uses a Profile ACL to define criteria that determines which packets should be forwarded according to an IP policy. Packets that meet the criteria defined in the Profile ACL are forwarded according to the **ip-policy** command that references the Profile ACL.

For example, you can define an IP policy that causes all telnet packets travelling from source network 9.1.1.0/24 to destination network 15.1.1.0/24 to be forwarded to destination address 10.10.10.10. You use a Profile ACL to define the selection criteria (in this case, Telnet packets travelling from source network 9.1.1.0/24 to destination network 15.1.1.0/24). Then you use an **ip-policy** command to specify what happens to packets that match the selection criteria (in this example, forward them to address 10.10.10.10). The following commands illustrate this example.

This command creates a Profile ACL called *prof1* that uses as its selection criteria all Telnet packets travelling from source network 9.1.1.0/24 to destination network 15.1.1.0/24:

```
xp(config)# acl prof1 permit ip 9.1.1.0/24 15.1.1.0/24 any any telnet 0
```

This Profile ACL is then used in conjunction with the **ip-policy** command to cause packets matching prof1's selection criteria (that is, telnet packets travelling from 9.1.1.0/24 to 15.1.1.0/24) to be forwarded to 10.10.10.10:

```
xp(config)# ip-policy p5 permit profile prof1 next-hop-list 10.10.10.10
```

See *IP Policy-Based Forwarding Configuration Guide* on page 367 for more information on using the **ip-policy** command.

Using Profile ACLs with the Traffic Rate Limiting Facility

Traffic rate limiting is a mechanism that allows you to control bandwidth usage of incoming traffic on a per-flow basis. A flow meeting certain criteria can have its packets re-prioritized or dropped if its bandwidth usage exceeds a specified limit.

For example, you can cause packets in flows from source address 1.2.2.2 to be dropped if their bandwidth usage exceeds 10 Mbps. You use a Profile ACL to define the selection criteria (in this case, flows from source address 1.2.2.2). Then you use a **rate-limit** command to specify what happens to packets that match the selection criteria (in this example, drop them if their bandwidth usage exceeds 10 Mbps). The following commands illustrate this example.

This command creates a Profile ACL called *prof2* that uses as its selection criteria all packets originating from source address 1.2.2.2:

```
xp(config)# acl prof2 permit ip 1.2.2.2
```

The following command creates a *rate limit definition* that causes flows matching Profile ACL prof2's selection criteria (that is, traffic from 1.2.2.2) to be restricted to 10 Mbps for each flow. If this rate limit is exceeded, the packets are dropped.

```
xp(config)# rate-limit client1 input acl prof2 rate-limit 10000000 exceed-action drop-packets
```

When the rate limit definition is applied to an interface (with the **rate-limit apply** command), packets in flows originating from source address 1.2.2.2 are dropped if their bandwidth usage exceeds 10 Mbps.

See *Limiting Traffic Rate* on page 496 for more information on using the **rate-limit** command.

Using Profile ACLs with Dynamic NAT

Network Address Translation (NAT) allows you to map an IP address used within one network to a different IP address used within another network. NAT is often used to map addresses used in a private, local intranet to one or more addresses used in the public, global Internet.

The X-Pedition router supports two kinds of NAT: *static* NAT and *dynamic* NAT. With dynamic NAT, an IP address within a range of local IP addresses is mapped to an IP address within a range of global IP addresses. For example, you can configure IP addresses on network 10.1.1.0/24 to use an IP address in the range of IP addresses in network 192.50.20.0/24. You can use a Profile ACL to define the ranges of local IP addresses.

The following command creates a Profile ACL called *local*. The local profile specifies as its selection criteria the range of IP addresses in network 10.1.1.0/24.

```
xp(config)# acl local permit ip 10.1.1.0/24
```

Note: When a Profile ACL is defined for dynamic NAT, only the source IP address field in the **acl** statement is evaluated. All other fields in the **acl** statement are ignored.

Once you have defined a Profile ACL, you can then use the **nat create dynamic** command to bind the range of IP addresses defined in the local profile to a range in network 192.50.20.0/24.

```
xp(config)# nat create dynamic local-acl-pool local global-pool 192.50.20.10/24
```

See *Network Address Translation Configuration Guide* *on page 379* for more information on using dynamic NAT.

Using Profile ACLs with the Port Mirroring Facility

Port mirroring refers to the X-Pedition router's ability to copy traffic on one or more ports to a "mirror" port, where an external analyzer or probe can be attached. In addition to mirroring traffic on one or more ports, the X-Pedition router can mirror traffic that matches selection criteria defined in a Profile ACL.

For example, you can mirror all IGMP traffic on the X-Pedition router. You use a Profile ACL to define the selection criteria (in this example, all IGMP traffic). Then you use a **port mirroring** command to copy packets that match the selection criteria to a specified mirror port. The following commands illustrate this example.

This command creates a Profile ACL called *prof3* that uses as its selection criteria all IGMP traffic on the X-Pedition router:

```
xp(config)# acl prof3 permit igmp
```

The following command causes packets matching Profile ACL prof3's selection criteria (that is, all IGMP traffic) to be copied to mirror port et.1.2.

```
xp(config)# port mirroring dst-ports et.1.2 src-acl prof3
```

See *Configuring for Port Mirroring* on page 507 for more information on using the **port mirroring** command.

Using Profile ACLs with the Web Caching Facility

Web caching is the X-Pedition router's ability to direct HTTP requests for frequently accessed Web objects to local cache servers, rather than to the Internet. Since the HTTP requests are handled locally, response time is faster than if the Web objects were retrieved from the Internet.

You can use Profile ACLs with Web caching in two ways:

- Specifying which HTTP traffic should always (or never) be redirected to the cache servers
- Specifying characteristics of Web objects that should not be cached

Redirecting HTTP Traffic to Cache Servers

You can use a Profile ACL to specify which HTTP traffic should always (or never) be redirected to the cache servers. (By default, when Web caching is enabled, all HTTP traffic from all hosts is redirected to the cache servers unless you specify otherwise.)

For example, you can specify that packets with a source address of 10.10.10.10 and a destination address of 1.2.3.4 always are sent to the Internet and never to the cache servers. The following commands illustrate this example.

This command creates a Profile ACL called *prof4* that uses as its selection criteria all packets with a source address of 10.10.10.10 and a destination address of 1.2.3.4:

```
xp(config)# acl prof4 permit ip 10.10.10.10 1.2.3.4
```

The following command creates a *Web caching policy* that prevents packets matching Profile ACL prof4's selection criteria (that is, packets with a source address of 10.10.10.10 and a destination address of 1.2.3.4) from being redirected to a cache server. Packets that match the profile's selection criteria are sent to the Internet instead.

```
xp(config)# web-cache policy1 deny hosts profile prof4
```

When the Web caching policy is applied to an interface (with the **web-cache apply interface** command), HTTP traffic with a source address of 10.10.10.10 and a destination address of 1.2.3.4 goes to the Internet instead of to the cache servers.

Preventing Web Objects from Being Cached

You can also use a Profile ACL to prevent certain Web objects from being cached. For example, you can specify that information in packets originating from Internet site 1.2.3.4 and destined for local host 10.10.10.10 not be sent to the cache servers. The following commands illustrate this example.

This command creates a Profile ACL called *prof5* that uses as its selection criteria all packets with a source address of 1.2.3.4 and a destination address of 10.10.10.10:

```
xp(config)# acl prof5 permit ip 1.2.3.4 10.10.10.10
```

To have packets matching Profile ACL prof5's selection criteria bypass the cache servers, use the following command:

```
xp(config)# web-cache policy1 create bypass-list profile prof5
```

When the Web caching policy is applied to an interface, information in packets originating from source address 1.2.3.4 and destined for address 10.10.10.10 is not sent to the cache servers.
See *Web Caching* on page 404 for more information on using the **web-cache** command.

Modifying ACLs

ACL entries are locked into the configuration database when the user applies the ACL to an interface. Once you apply an ACL, you may modify it using any of the following mechanisms:

- Using the X-Pedition router's ACL Editor
- Editing ACLs on a remote host and uploading them to the X-Pedition router using TFTP or RCP

The following sections describe these methods.

Maintaining ACLs Using the ACL Editor

In addition to the traditional method of maintaining ACLs using TFTP or RCP, the X-Pedition router provides a simpler and more user-friendly mechanism to maintain ACLs: the ACL Editor.

The ACL Editor can only be accessed within Configure mode using the **acl-edit** command. You edit an ACL by specifying its name together with the **acl-edit** command. For example, to edit ACL 101, you issue the command **acl-edit 101**. The only restriction is that when you edit a particular ACL, you cannot add rules for a different ACL. You can only add new rules for the ACL that you are currently editing. When the editing session is over, that is, when you are done making changes to the ACL, you can save the changes and make them take effect immediately. Within the ACL editor, you can add new rules (**add** command), delete existing rules (**delete** command) and re-order the rules (**move** command). To save the changes, use the **save** command or simply exit the ACL Editor.

If you edit and save changes to an ACL that is currently being used or applied to an interface, the changes will take effect immediately. There is no need to remove the ACL from the interface before making changes and reapply it after changes are made. The process is automatic.

Editing ACLs Offline

You can create and edit ACLs on a remote host and then upload them to the X-Pedition router with TFTP or RCP. With this method, you use a text editor on a remote host to edit, delete, replace, or reorder ACL rules in a file. Once the changes are made, you can then upload the ACLs to the router using TFTP or RCP and make them take effect on the running system. The following example describes how you can use TFTP to help maintain ACLs on the router.

Suppose the following ACL commands are stored in a file on some hosts:

```
no acl *
acl 101 deny tcp 10.11.0.0/16 10.12.0.0/16
acl 101 permit tcp 10.11.0.0 any
acl 101 apply interface int12 input
```

The first command, **no acl ***, negates all commands that start with the keyword, “acl”. This tells the X-Pedition router to remove the application and the definition of any ACL. You can be more selective if you want to remove only ACL commands related to, for instance, ACL 101 by entering, **no acl 101 ***. The negation of all related ACL commands is important because it removes any potential confusion caused by the addition of new ACL rules to existing rules. Basically, the **no acl** command cleans up the system for the new ACL rules.

Once the negation command is executed, the second and the third commands proceed to redefine ACL 101. The final command applies the ACL to interface int12.

If the changes are accessible from a TFTP server, you can upload and make the changes take effect by issuing commands like the following:

```
xp# copy tftp://10.1.1.12/config/acl.changes to scratchpad
xp# copy scratchpad to active
```

The first **copy** command uploads the file **acl.changes** from a TFTP server and puts the commands into the temporary configuration area, the scratchpad. The administrator can re-examine the changes if necessary before committing the changes to the running system. The second **copy** command makes the changes take effect by copying from the scratchpad to the active running system.

If you need to re-order or modify the ACL rules, you must make the changes in the **acl.changes** file on the remote host, upload the changes, and make them effective again.

Enabling ACL Logging

To see whether incoming packets are permitted or denied because of an ACL, enable ACL logging. You can enable logging when applying the ACL or you can enable logging for a specific ACL rule.

Note: ACL logging is not supported for IPv6 ACLs.

The following commands define an ACL and apply the ACL to an interface, with logging enabled for the ACL:

```
acl 101 deny ip 10.2.0.0/16 any any any
acl 101 permit ip any any any any
acl 101 apply interface int1 input logging on
```

With ACL logging turned on, the router prints a message on the console to indicate whether a packet was dropped *or* forwarded. If you have a Syslog server configured for the X-Pedition router, the same information will also be sent to the Syslog server. The following commands define an ACL and apply the ACL to an interface. In this case, logging is enabled for a specific ACL rule:

```
acl 101 deny ip 10.2.0.0/16 any any any log
acl 101 permit ip any any any any
acl 101 apply interface int1 input
```

For the above commands, the router prints out messages on the console only when packets that come from subnet 10.2.0.0/16 on interface ‘int1’ are dropped.

Note: When logging is enabled on a per-rule basis, you do not need to specify the **logging on** option when applying the ACL to an interface. With per-rule logging enabled, only the **logging off** option has an effect when the ACL is applied; this option turns off *all* ACL logging.

When applying an ACL to an interface, you may also specify the following logging options:

deny-only	Only packets denied by this ACL will be logged.
deny-syslog	Only packets denied by this ACL will be logged, and messages will be sent only to a syslog server (if configured), and not to the console.
permit-only	Only packets permitted by this ACL will be logged.
permit-syslog	Only packets permitted by this ACL will be logged, and messages will be sent only to a syslog server (if configured), and not to the console.
on-syslog	All ACL events (permitted and denied packets) will be logged, with messages only being sent to a syslog server (if configured), and not to the console.

Under normal circumstances, when an ACL denies a particular traffic pattern, only the first packet denied will be reported. Subsequent packets that are similar to the first packet will be dropped by the hardware with no reporting. In situations where it is necessary to see exactly how much traffic has been denied, you may use the **report-denied** configuration option. The report-denied option can be specified when applying an ACL to either an interface or a port. This option has two sub-options:

All This will report all denied packets. Reporting options will still be the same as those mentioned in the preceding paragraphs. This option effectively removes the X-Pedition router's hardware protection against DOS attacks, and is not recommended for normal usage.

Periodic This will periodically report statistics on all denied traffic streams. The reporting frequency defaults to 15 seconds, and is configurable with the **acl logging set deny-report-frequency** <*number*> command.

Note: A deny message can occur when an ACL denies a particular traffic pattern under these circumstances: 1) there are packets queued up in the interface to the CPU before the first packet is completely processed, and 2) a hardware flow has not yet been created. The packets will go to the CPU and be processed through the ACL causing the deny message.

Note: The **report-denied** option is available for IPv4 ACLs only.

Before enabling ACL logging, you should consider its impact on performance. With ACL logging enabled, the router prints out a message at the console before the packet is actually forwarded or dropped. Even if the console is connected to the router at a high baud rate, the delay caused by the console message is still significant. This can get worse if the console is connected at a low baud rate, for example, 1200 baud. Furthermore, if a Syslog server is configured, then a Syslog packet must also be sent to the Syslog server, creating additional delay. Therefore, you should consider the potential performance impact before turning on ACL logging.

Monitoring ACLs

The X-Pedition router provides a display of ACL configurations active in the system.

To display ACL information, enter the following commands in Enable mode.

Show all ACLs.	**acl show all**
Show a specific ACL.	**acl show aclname** <*name*> | **all**
Show an ACL on a specific interface.	**acl show interface** <*name*>
Show ACLs applied to all IPv4 interfaces.	**acl show interface all-ip**
Show ACLs applied to all IPv6 interfaces.	**acl show interface all-ipv6** **acl show interface all-ipv6in4-tunnels**
Show static entry filters.	**acl show service**

Chapter 26

Security Configuration Guide

Security Overview

The X-Pedition router provides security features that help control access to the router and filter traffic going through the router. Access to the router can be controlled by:

- Enabling RADIUS
- Enabling TACACS Plus

  Note: The X-Pedition router no longer supports TACACS and will ignore any commands used for it in the configuration—without generating an error.

- Security Attack Monitor (SAM)
- Password Management
- SNMP
- *Audit Trail* on page 66
- *Secure Shell (SSH) Server* on page 41

Traffic filtering on the X-Pedition router enables:

- Layer-2 security filters - Perform filtering on source or destination MAC addresses.
- Layer-3/4 Access Control Lists - Perform filtering on source or destination IP address, source or destination TCP/UDP port, TOS or protocol type for IPv4 traffic. Perform filtering on source or destination IP address, traffic class, or next header field for IPv6 traffic. Perform filtering on source or destination IPX address, or source or destination IPX socket. Perform access control to services provided on the X-Pedition router, for example, Telnet server and HTTP server.

Note: Currently, Source Filtering is available on X-Pedition router WAN cards; however, you must apply the filter to the entire WAN card.

Configuring Access Security

This section describes the following methods of controlling access to the X-Pedition router:

- RADIUS
- TACACS Plus
- Security Attack Monitor (SAM)
- Passwords
- SNMP

RADIUS

Configuring RADIUS

You can secure login or Enable mode access to the X-Pedition router by enabling a Remote Authentication Dial-In Service (RADIUS) client. A RADIUS server responds to the X-Pedition RADIUS client to provide authentication.

You can configure up to five RADIUS server targets on the X-Pedition router. A timeout is set to tell the X-Pedition router how long to wait for a response from RADIUS servers.

To configure RADIUS security, enter the following commands in Configure mode:

Specify a RADIUS server.	**radius set server** <*hostname or IP-addr*>
Set the RADIUS time to wait for a RADIUS server reply.	**radius set timeout** <*number*>
Determine the X-Pedition action if no server responds.	**radius set last-resort password**|**succeed**|**deny**
Enable RADIUS.	**radius enable**
Cause RADIUS authentication at user login or when user tries to access Enable mode.	**radius authentication login**|**enable**
Logs specified types of command to RADIUS server.	**radius accounting command level** <*level*>

Logs to RADIUS server when shell is stopped or started on X-Pedition router.	**radius accounting shell start|stop|all**
Logs to RADIUS server SNMP changes to startup or active configuration.	**radius accounting snmp active|startup**
Logs specified type(s) of messages to RADIUS server.	**radius accounting system fatal|error|warning|info**

Monitoring RADIUS

You can monitor RADIUS configuration and statistics within the X-Pedition router. To monitor RADIUS, enter the following commands in Enable mode:

Show RADIUS server statistics.	**radius show stats**
Show all RADIUS parameters.	**radius show all**

The X-Pedition router also allows you to display information on up to five previous users who logged in to the X-Pedition router using TACACS+ or RADIUS. To display information on the last users to log in to the X-Pedition router enter the following:

```
xp-181-11# system show security-log

User ID                 :  johnny
tty                     :  tty1
Security Type           :  Radius
Last AAA server used    :  10.136.15.101
Number of Sessions      :  1
Start Session Time      :  2002-06-27 09:10:16
Connection Status       :  Currently Connected
Last Session ended at   :  2002-06-27 09:09:54
Time Last Accessed      :  2002-06-27 09:26:34
Last Command            :  system show security-log
Current Mode            :  Enable
Config command Cntr     :  0
Enable command Cntr     :  1
Login command Cntr      :  3
Total command Cntr      :  4
```

Configuring Passwords

The X-Pedition router provides password authentication for accessing the User and Enable modes. If RADIUS is not enabled on the X-Pedition router, only local password authentication is performed. To configure X-Pedition passwords, enter the following commands in Configure mode:

Set User mode password.	**system set password login** <*string*>
Set Enable mode password.	**system set password enable** <*string*>

RADIUS Authorization

The X-Pedition router allows users to log into the router using RADIUS authentication, a mechanism used to verify user identity prior to granting system access. RADIUS Authorization, the process of controlling facility use within the system once the router authenticates a user, permits the system administrator to assign one of three levels of facility rights (based on the CLI command mode) to the user: login, enable, and configure. Upon authentication, a user gains access to the system at the authorization level set by the Service-type attribute (Administrative, NAS Prompt, or Login). Although users may move to a lower level of access, users may not move to higher levels that require greater rights. For example, a user with enable rights enters enable mode directly—although access to login mode is allowed, access to configure mode is prohibited.

If the Service-type sent from the RADIUS server is set to **Administrative** (read-write privileges), the router sends the user to configure mode and grants access to all modes on the router. Users granted this level of authorization may perform all commands available on the router.

If the Service-type sent from the RADIUS server is set to **NAS Prompt** (read-only privileges), the router sends the user to enable mode and grants access to only enable and login modes. Users granted this level of authorization may perform only those commands listed in enable and login modes, including **reboot**, **copy tftp-server to startup**, and **ssh-server generate-host-key**. As a result, X-Pedition router and RADIUS server administrators should take this into account when configuring the Service-types for users.

If the Service-type sent from the RADIUS server is set to **Login** (logon privileges), the router sends the user to login mode and limits the use of commands to only those available from login mode.

Note: If there is no Service-type defined, or the Service-type sent from the RADIUS server is not Administrative, NAS Prompt, or Login, RADIUS will behave as it has in previous versions. This will maintain backwards compatibility with older RADIUS server configurations.

Users may also configure local passwords for each user mode with the **system set password** command. If a user changes from the initial user mode after authenticating and attempts to enter a higher mode, the X-Pedition router will prompt the user to enter a local password—even if the user already has sufficient privileges to enter the mode. The **radius authentication enable** command still functions in the same way it has in previous versions. When this command is present in the active configuration, the router will authenticate the user through RADIUS instead of through a local password if the user attempts to move from login mode to enable mode.

TACACS Plus

Configuring TACACS Plus

You can secure login or Enable mode access to the X-Pedition router by enabling a TACACS Plus client. A TACACS Plus server responds to the X-Pedition TACACS Plus client to provide authentication. You can configure up to five TACACS Plus server targets on the X-Pedition router. A timeout is set to tell the X-Pedition router how long to wait for a response from TACACS Plus servers. To configure TACACS Plus security, enter the following commands in Configure mode:

Specify a TACACS Plus server.	**tacacs-plus set server** <*hostname or IP-addr*>
Set the TACACS Plus time to wait for a TACACS Plus server reply.	**tacacs-plus set timeout** <*number*>
Determine the X-Pedition action if no server responds.	**tacacs-plus set last-resort password**|**succeed**|**deny**
Enable TACACS Plus.	**tacacs-plus enable**
Cause TACACS Plus authentication at user login or when user tries to access Enable mode.	**tacacs-plus authentication login**|**enable**
Cause TACACS Plus authentication at user login or when user tries to access Enable mode.	**tacacs-plus authentication login**|**enable**
Logs specified types of command to TACACS Plus server.	**tacacs-plus accounting command level** <*level*>
Logs to TACACS Plus server when shell is stopped or started on X-Pedition router.	**tacacs-plus accounting shell start**|**stop**|**all**
Logs to TACACS Plus server SNMP changes to startup or active configuration.	**tacacs-plus accounting snmp active**|**startup**
Logs specified type(s) of messages to TACACS Plus server.	**tacacs-plus accounting system fatal**|**error**|**warning**|**info**

Note: The X-Pedition router no longer supports TACACS and will ignore any commands used for it in the configuration—without generating an error.

Monitoring TACACS Plus

You can monitor TACACS Plus configuration and statistics within the X-Pedition router. To monitor TACACS Plus, enter the following commands in Enable mode:

Show TACACS Plus server statistics.	**tacacs-plus show stats**
Show all TACACS Plus parameters.	**tacacs-plus show all**

The X-Pedition router also allows you to display information on up to five previous users who logged in to the X-Pedition router using TACACS+ or RADIUS. To display information on the last users to log in to the X-Pedition router enter the following:

```
xp-181-11# system show security-log

User ID                 : johnny
tty                     : tty1
Security Type           : Tacacs+
Last AAA server used    : 10.136.15.101
Number of Sessions      : 1
Start Session Time      : 2002-06-27 09:10:16
Connection Status       : Currently Connected
Last Session ended at   : 2002-06-27 09:09:54
Time Last Accessed      : 2002-06-27 09:26:34
Last Command            : system show security-log
Current Mode            : Enable
Config command Cntr     : 0
Enable command Cntr     : 1
Login command Cntr      : 3
Total command Cntr      : 4
```

Configuring Passwords

The X-Pedition router provides password authentication for accessing the User and Enable modes. If TACACS Plus is not enabled on the X-Pedition router, only local password authentication is performed. To configure X-Pedition passwords, enter the following commands in Configure mode:

Set User mode password.	**system set password login** <*string*>
Set Enable mode password.	**system set password enable** <*string*>

Security Attack Monitor (SAM)

One of the major Control Module (CM) functions is to process unlearned flows. When the CM processes the first packet of a flow, the CM installs a hardware flow in the ingress line module to allow the module to forward subsequent packets for the flow without CM intervention—this works extremely well for networks with normal traffic patterns. However, if the router experiences a significant amount of network traffic for an extended duration, the traffic may negatively impact on the X-Pedition router's ability to keep the network functioning smoothly. Examples of excessive network traffic include Denial of Service (DoS) attacks and network management stations (NMS) configured to aggressively contact all devices on a network. For simplicity, this document will refer to DoS attacks and aggressive NMS configurations collectively as *attacks*.

The Security Attack Monitor (SAM) combats attacks by:

1. Detecting the attack.
2. Identifying the LAN segment from which the attack originated.
3. Assigning a higher priority to normal traffic than to attack traffic.
4. Rate-limiting attack traffic to protect adjacent network devices.
5. Monitoring network traffic until the attack stops. After the attack stops, the router will reassign normal priorities to traffic on all LAN segments.

SAM is not an IDS system. As such, it does not perform stateful inspection of packets, nor will it determine the contents of a packet and make decisions based on the payload. SAM does not protect against every kind of attack that exists today or that may develop in the future. For additional SAM limitations, please refer to Tech Bulletin ent15089.

Detection

When the X-Pedition router detects attack traffic, SAM will activate after a user-configurable delay. This delay prevents SAM activation with bursts of traffic caused by normal network activities, such as an STP topology change or a route change.

Analysis

After the activation delay expires, SAM will collect and analyze a sample of packets to determine which port is responsible for the most unlearned traffic.

Activation

When SAM activates, it reprioritizes traffic and assigns a higher priority to normal traffic than to attack traffic. If rate-limiting is configured, SAM will also rate-limit the attack traffic to the user-configured rate. The reprioritization and rate-limit apply to a range of ports. On an ER16, each activation will affect half of the ports on a line module. On all other X-Pedition systems, the reprioritization and rate-limit will affect all ports on a line module. Users with established flows on activated ports will remain unaffected as long as their flows remain in use; conversely, attempts to establish new flows will not likely succeed. Users on ports SAM hasn't taken action against should see no network performance degradation when SAM activates.

Note: After SAM activates, the CPU may continue to operate at or near 100% of capacity, but this does not indicate poor network performance.

Multiple Activations

If multiple ports are under attack, SAM will repeat the detection, analysis, and activation steps until it has taken action against all attacks.

Deactivation

The deactivation delay prevents temporary interruptions in attack traffic from deactivating SAM prematurely. Before SAM will deactivate, attacks on each activated port range must subside for the specified deactivation delay.

Configuration Options

Enable SAM

Enable the Security Attack Monitor (disabled by default).	**sam enable**

Set Activation and Deactivation Delays

Configure the activation delay (in seconds) to wait after the router detects an attack before SAM reprioritizes and rate-limits attack traffic.	**sam set activation-delay** <*num*>
Configure the deactivation delay (in seconds) to wait after an attack ceases before SAM deactivates.	**sam set deactivation-delay** <*num*>

Set the Rate-Limit

Configure the rate (in packets per second) at which the router will process and forward all attack traffic.	**sam set rate-limit** <*num*>
Disable rate-limiting. **Note:** CM will process and forward as much attack traffic as possible.	**sam set rate-limit 0**

Monitor SAM

To display SAM statistical data, enter the following commands from Enable mode.

Display SAM activation history	**sam show history**
Display SAM statistics	**sam show statistics**
Display SAM status	**sam show status**

To reset SAM statistics, enter the following commands from Enable mode.

Clear SAM activation history	**sam clear history**
Clear SAM statistics	**sam clear statistics**

Password Policy Management

Secure access to the X-Pedition router through password protection and policies is available in both single- and multi-user modes. Global password policies are established using the **system set password-policy** command and apply to all passwords in single- or multi-user mode unless specifically overridden by one of the command options described below.

The following policy elements are configurable in single-user and multi-user modes:

- Enable or disable password policy verification (**enabled** by default)
- Minimum password length (by default, **8 characters**)

  Note: For recommendations on selecting a password, refer to *Passwords* on page 36.
- Access management including:
  - The amount of time a user has to log in successfully (by default, **60 seconds**)
  - The number of login failures to allow before disabling a user's account (by default, **6**)
  - After an account is disabled, the amount of time that must pass before permitting another login attempt (by default, **60 minutes**)
- Password change management including:
  - Require a user to change the password following the first login (**off** by default)
  - The number of days the password is valid (by default, **90** days)
  - The number of days prior to password expiration to warn the user of the pending expiration (by default, **14 days**)
  - The number of previous passwords that cannot be re-used (by default, **5**)

Single-User Mode

Creating a Password

By default, the X-Pedition router operates in single-user mode with password access enabled and no passwords defined. To define a password for Login, Enable, or Configure mode, use the **system set password** command from Configure mode. The following example sets an Enable mode password:

```
xp(config)# system set password enable MyPassword
```

Note: Passwords are case *sensitive*. In other words, the X-Pedition router recognizes upper- and lower-case letters as different characters (e.g., “M” is not the same as “m”).

You must set the password for each mode individually (i.e., you may use a different password for each mode). If a password is configured for Enable mode, the X-Pedition router asks for the password when you enter the **enable** command. If no password is defined, the X-Pedition router will advise you to configure a password, then switch to Enable mode—from here you can access Configure mode and make configuration changes. Access to Configuration mode may be configured to require a password. For recommendations on selecting a password, refer to *Passwords* on page 36.

The X-Pedition router stores passwords in the startup configuration file. If you copy a configuration file from one X-Pedition router to another, the passwords in the file will automatically apply to the new router. When you activate a new password by copying the **system set password** command to the active configuration, the X-Pedition router replaces the command with a **system set hashed-password** command to hide the password text in the configuration file. This prevents others who access this file from viewing the password.

To remove the Enable mode password defined above, enter the following command from Configure mode:

```
xp(config)# system set password enable none
```

Password Policies

Once a password is established, the default password policies apply unless configured otherwise. The following example requires a user to change his password after logging in for the first time. All remaining password policies use the default values.

```
xp(config)# system set password-policy change-after-first-login on
```

In the next example, the password expiration notice changes to 10 days, the login failure retry time shortens to 30 minutes, and the minimum password length changes to 6 characters. All other parameters use their default values.

```
xp(config)# system set password-policy expire-warning 10 login-failure-grace-time 30
minimum-length 6
```

Note: Passwords will appear on the console as asterisks to prevent them from being observed.

Multi-User Mode

Multi-user mode password security employs individual user accounts to grant CLI permissions on a case-by-case basis—this requires that each user log in via username and password. The X-Pedition router supports up to 256 concurrent user accounts (although you may still employ either protocol, multi-user accounting no longer requires TACACS+ or RADIUS). To define a password for Login, Enable, or Configure mode, use the **system set password** command from Configure mode. In multi-user mode, you may configure the following information for each user account:

- Username
- New password
- Password lifetime limit
- Mode or privilege where the user can gain access to the system
- Whether or not to disable the user account after too many failed login attempts.

The following example creates a new user account for Jane and grants password access to configure mode. All other options remain at their default levels. For details on specific password and privilege options associated with the **system set user** command, consult the *Enterasys X-Pedition Native Command Line Interface Reference Manual*.

```
xp(config)# system set user Jane privilege-level config
```

Note: You must always enter a password when creating an account for a new user or when changing the access privileges for an existing user. Once users are configured, the login prompt changes to request the username and password.

Passwords are case *sensitive*. In other words, the X-Pedition router recognizes upper- and lower-case letters as different characters (e.g., “M” is not the same as “m”).

You must set the password for each mode individually (i.e., you may use a different password for each mode). If a password is configured for Enable mode, the X-Pedition router asks for the password when you enter the **enable** command. If no password is defined, the X-Pedition router will advise you to configure a password, then switch to Enable mode—from here you can access Configure mode and make configuration changes. Access to Configuration mode may be configured to require a password.

In addition to configuring multiple user accounts, you may enable the Syslog audit trail feature to track all user attempts to access the router or make configuration changes. Audit log messages record information such as the username, source IP address, and session type, and are especially useful when working with network problems and evaluating changes to system configuration over time. Although audit trails can be applied to single-user transactions, they are particularly useful in multi-user environments. For more information, see *Audit Trail* on page 66.

SNMP

SNMPv1, SNMPv2c, and SNMPv3 Agent Overview

This feature expands SNMP management support on the X-Pedition router to include the current IETF defined elements of SNMPv1, SNMPv2c, and SNMPv3 protocols.

Security

Simple Network Management Protocol Version 3 (SNMPv3) is an interoperable standards-based protocol for network management. SNMPv3 provides secure access to devices by a combination of authenticating and encrypting packets over the network. The security features provided in SNMPv3 are:

- **Message integrity**. Ensuring that a packet has not been tampered with in-transit.
- **Authentication**. Determining the message is from a valid source.
- **Encryption**. Scrambling the contents of a packet to prevent it from being seen by an unauthorized source.

SNMPv3 provides for both security models and security levels. A security model is an authentication strategy that is set up for a user and the group in which the user resides. A security level is the permitted level of security within a security model. A combination of a security model and a security level will determine which security mechanism is employed when handling an SNMP packet. Three security models are available: SNMPv1, SNMPv2c, and SNMPv3. Table 1 identifies what the combinations of security models and levels mean:

Table 12. SNMP Security Models and Levels

Model	Level	Authentication	Encryption	What Happens
v1	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for authentication.

Model	Level	Authentication	Encryption	What Happens
v3	authNoPriv	MD5 or SHA-1	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA1 algorithms.
v3	authPriv	MD5 or SHA-1	DES	Provides authentication based on the HMAC-MD5 or HMAC-SHA1 algorithms. Provides 56-bit encryption based on the CBC-DES (DES-56) standard in addition to authentication.

Access Control

SNMPv3 also provides fine-granularity access control to management information. The View-Based Access Control Model (VACM) allows subsets of management information to be organized into “views.” There are three types of views: read, write, and notify. Management information that is in a user's view gives the user the corresponding access-level to that management information -- either read, write, or notify. Individual users are organized into groups, and using VACM it is possible to pre-define which views will be available to a given group based on the security model and security level used to request access.

In other words, VACM gives the ability to allow or deny access to any individual item of management information depending on the user's group membership and the level of security provided by the communications channel.

Reliability

In addition to better security and better access control, SNMPv3 also provides a higher degree of reliability for notifying management stations when critical events occur.

Traditionally SNMP agents communicated events to SNMP managers via “traps.” However, if a temporary network problem prevents the reception of the trap by the manager, then the trap is lost. SNMPv3 provides “informs” which are a more reliable form of traps. The SNMP agent initiates the inform process by sending an inform request to the manager. The manger responds to the inform request to acknowledge receipt of the message. If the inform is not received by the manager, the inform request will timeout and a new inform request will be sent. Subsequent inform requests will be sent as previous requests time-out until either an acknowledgement is received from the manager, or until a pre-specified retry-count is reached.

Supported SNMPv3 MIBs

SNMP supports the following MIBs:

RFC No.	Title
RFC 1471	PPP LCP (Link Control Protocol)
RFC 1472	PPP Security Protocol
RFC 1473	PPP IP NCP (Network Control Protocol)
RFC 1474	PPP Bridge NCP
RFC 1493	Definitions of Managed Objects for Bridges
RFC 1512	FDDI MIB
RFC 1595	SONET / SDH MIB
RFC 1657	BGP4 MIB
RFC 1695	ATM MIB
RFC 1724	RIPv2 MIB
RFC 1742	AppleTalk Management Information Base II
RFC 1757	Remote Network Monitoring (RMON) Management Information Base
RFC 1850	OSPF MIB
RFC 1901	Introduction to Community-based SNMPv2 (SNMPV2c)
RFC 1907	SNMP v2 MIB
RFC 2011	Internet Protocol (IP) MIB using SMIv2
RFC 2012	Transmission Control Protocol (TCP) MIB using SMIv2
RFC 2013	User Datagram Protocol (UDP) MIB using SMIv2
RFC 2021	Remote Network Monitoring Version 2 (RMON 2)
RFC 2096	IP Forwarding MIB
RFC 2115	Frame Relay DTE using SMIv2
RFC 2358	Ethernet Like Interface MIB
RFC 2358	Ethernet-like Interface Types MIB
RFC 2495	E1 / DS1 MIB
RFC 2496	E3 / DS3 MIB
RFC 2576	Coexistence between Version1, Version2, and Version 3 of Internet-standard Network Management Framework
RFC 2618	RADIUS Authentication Client

RFC No.	Title
RFC 2668	IEEE 802.3 Medium Attachment Units (MAUs) MIB
RFC 2674	IETF Q MIB for Bridge with Traffic Classes, Multicast Filtering and VLAN Extension
RFC 2737	Entity MIB
RFC 2787	VRRP
RFC 2790	Host Resources MIB
RFC 2863	Interfaces Group using SMIv2
RFC 3411	An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks
RFC 3412	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
RFC 3413	Simple Network Management Protocol (SNMP) Applications
RFC 3414	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
RFC 3415	View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)
RFC 3416	Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)

Notes:

- The SNMPv3 management framework, security model, and protocol operations are defined in RFC 3411 through 3416. This framework handles SNMPv1, SNMPv2c, and SNMPv3 message types.
- RFC 2576 defines how the SNMPv1 and SNMPv2c community based security model coexists with the SNMPv3 user based security model.
- RFC 1901 defines the SNMPv2c message type. This message type allows for the protocol operations defined in RFC 3416 to be used with the community based security model.

Configuration Overview

Configuring the X-Pedition router for SNMPv3 support consists of the following steps:

1. **Creating Users**. Users are created and configured to use an optional secure authentication protocol and/or encryption.
2. **Creating Communities**. Communities are created to allow SNMPv1 and SNMPv2c coexistence with SNMPv3.
3. **Creating Groups**. Groups are created and assigned a read view, a write view, and a notify view.
4. **Assigning Users**. Users are assigned to their respective groups.
5. **Defining Views**. Views are defined as collections of OID subtrees. Appropriate views are created for the groups that were created previously.
6. **Defining Targets**. Management targets are defined for receiving traps or informs.
7. **Configuring Target Parameters**. Parameters for communicating with designated targets are defined.
8. **Creating Notification Filters**. Optionally, notification filters can be created to constrain the types of traps or informs that a given management target will receive.
9. **Configuring Informs**. Optionally, any targets can be configured to receive SNMPv3 Informs.
10. **Assigning Aliases to Interfaces**. Used to assign additional identification information to any interface handled by the ifXTable (i.e., physical ports, IP interfaces, IPX interfaces, VLANs, and SmartTRUNKs).

Creating Users

Users can be configured to use a combination of authentication and/or privacy. To take advantage of the security features of SNMPv3, it is recommended that users be configured with at least authentication which helps to eliminate most of the potential security risks associated with SNMPv1.

To create a new user, use the **snmp set user** command. For a detailed explanation of the **snmp set user** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual*. Most commonly, users will be configured to authenticate with the local SNMP engine only, however, individual users can be configured to authenticate with remote authoritative SNMP engines if necessary (see *Configuring Informs* on page 463). The example below illustrates the creation of a new user named “jane” configured to use the HMAC-SHA-96 authentication protocol.

```
xp(config)# snmp set user jane engine-id local auth sha1
```

Note: After the router executes the **snmp set user** command, the CLI will prompt the user for password information.

The next example shows how to configure a user to use both encryption and authentication. In this case the user will be configured to use the HMAC-MD5-96 authentication protocol along with CBC-DES encryption for privacy.

```
xp(config)# snmp set user john engine-id local auth md5 priv des
```

Note: After the router executes the **snmp set user** command, the CLI will prompt the user for password information.

After saving the configuration to make it active, authentication and privacy keys will be generated and localized to the Engine-ID specified (in this case the local Engine-ID). Note that the resulting keys will not be visible in the X-Pedition router's configuration file. If the user's passwords are lost they cannot be recovered and the user will need to be reconfigured using the **snmp set user** command.

Once the user accounts are created, the individual users can be grouped to assign access rights based on the level of security the user will have when remotely accessing the X-Pedition router (see *Creating Groups* on page 458).

Creating Communities

When using SNMPv1 and SNMPv2c, user accounts are not available. However, configuring a community in SNMPv3 allows the protocol to coexist with SNMPv1 and SNMPv2c. To configure access for SNMPv1 and SNMPv2c, use the following command:

snmp set community <*community*> **privilege [read read-write] [v1| v2c]**

This is the simplest form of the **snmp set community** command and maps to permanent default groups/views to grant the community access to the entire MIB tree. Users must enter a separate command to define community information for each security model (i.e., SNMPv1 and SNMPv2c). In cases where it is necessary to restrict access for a community, define a group (see *Creating Groups* on page 458) and use the command below to map the community to the group:

snmp set community <*community*> **group** <*group-name*> **[v1| v2c]**

Note: The first two versions of the **snmp set community** command described above imply a security-to-group mapping that uses the **privilege** or **group** option in conjunction with the **v1** or **v2c** option. However, when specifying the name, security-name, or tag option, the **privilege** and **group** options are not allowed. Instead, you must use another command, **snmp set community-to-group** to provide the security-to-group mapping. This allows multiple communities to use the same name or security-name.

Besides granting access to SNMPv1 and SNMPv2 protocols, the **snmp set community** command also allows other configurations. Using the command below in conjunction with the **snmp set target** command allows users to configure additional functionality specified in RFC 2576—including the ability to configure multiple snmpCommunityTable entries for the same community name or security name and to restrict access to only certain NMS systems. Before you begin working with communities, you must configure a community and assign a name, security name, or transport tag. To configure this information, use the following command—please note that the parameters used in this command are optional—if not specified, the name is the same as <*community*>, the security name is the same as the <*name*>, and the tag does not restrict access to certain NMS systems.

snmp set community <*community*> **name** <*name*> **security-name** <*security-name*> **tag** <*tag-name*>

When using this form of the **snmp set community** command, you must configure access for the community by using the **snmp set community-to-group** command. A separate **community-to-group** command (below) is required for each security model (v1 or v2c).

snmp set community-to-group <*security-name*> **to** <*group-name*> **[v1| v2c]**

To further limit access to a set of NMS systems, use a combination of the **snmp set target** and **snmp set target community** commands. The **snmp set target** command uses a "taglist" parameter composed of a list of strings (e.g., “tag1 tag2 tag3 tag3”) to connect the target with other configuration items.

The **snmp set community** command also uses a "tag" parameter. This parameter may contain only one tag (e.g., "tag2") and is used to relate this community with a target or targets. Any target with "tag2" in its "taglist" will be associated with the community. When using communities, the router uses the target’s ip-address, port, and mask to determine the allowed transport address of the NMS. Because the community "tag" value may appear in more than one target, messages with the same community may be authorized from more than one NMS.

snmp set target ip-address 10.10.10.10 port 161 mask 0xff:ff:ff:f0:00:00 tags "DefaultTraps mytag"
snmp set community public tag mytag

The configuration above allows access using community *public* from NMS stations with addresses 10.10.10.1 to 10.10.10.15 only. The first four bytes of the mask are used to mask the ip-address, the last 2 bytes are used to mask the port. For each bit in the mask that is set to 1, the corresponding target ip-address/port bit must match the transport ip-address/port bit of messages received from an NMS. If a bit is set to 0, no match is required.

Creating Groups

Groups facilitate the assignment of access rights to specific users. Users who require the same level of access should be grouped together into the same group. Different groups can then be created with the necessary access rights. To create groups, use the **snmp set group** command. For a detailed explanation of the **snmp set group** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual*. The example below illustrates the creation of the group named “opers.” Based on the example, users belonging to the opers group who are authenticated will have read access to the “restricted” view, no write access, and notify access to the “all” view. Users belonging to the opers group who are not authenticated will have no access.

```
xp(config)# snmp set group opers v3 auth read restricted notify all
```

The next example shows the creation of the “admins” group. In this case, group members will only have partial access if they are authenticated without privacy, but full access if they are authenticated with privacy.

```
xp(config)# snmp set group admins v3 auth read internet notify all
xp(config)# snmp set group admins v3 priv read all write all notify all
```

Both examples make use of several built-in views that are provided with the X-Pedition router. However, to fully utilize the access-control capabilities of SNMPv3, you should consider defining your own views (see *Defining Views* on page 459).

Assigning Users

Once users and groups have been created, the users must be assigned to the groups so that the individual users will inherit the access rights of the group. To assign a user to a group, use the **snmp set user-to-group** command. For a detailed explanation of the **snmp set user-to-group** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual*. The example below illustrates how to assign the user named “root” to the group named “admins.” The user root will then inherit the access rights configured for the “admins” group.

```
xp(config)# snmp set user-to-group root to admins
```

You may also assign an SNMPv1 or SNMPv2c community to a group through the **snmp set community** command. When using the **snmp set community** command, you must configure access for the community by using the **snmp set community-to-group** command. A separate **community-to-group** command is required for each security model (v1 or v2c).

snmp set community-to-group <*security-name*> **to** <*group-name*> **[v1| v2c]**

The next example illustrates the process of creating a user, creating a group, and assigning the user to the group. First we will create the user “jane” for use with the local SNMP engine, using the HMAC-SHA-96 authentication protocol, and the authentication password “foo”. Next the “operators” group will be created with read, write, and notify access to the built-in “all” view for users using the SNMPv3 protocol with authentication enabled. Then the user “jane” will be assigned to the “operators” group.

```
xp(config)# snmp set user jane engine-id local auth sha1
xp(config)# snmp set group operators v3 auth read all write all notify all
xp(config)# snmp set user-to-group jane to operators
```

This process will give the user “jane” read, write, and notify access to the built-in “all” view when she is successfully authenticated by the local SNMP engine. Note that this example makes use of one of the built-in views. For information on how to define custom views, see *Defining Views* on page 459.

Note: After the router executes the **snmp set user** command, the CLI will prompt the user for password information.

Defining Views

Views are a collection of subtrees of management information that are used to define a subset of the OID space that a group of users may have access to. There are three types of views: read, write, and notify. Read views give read access to management information, while write views give write access to management information. Notify views allow notifications—either traps or informs—to be sent to a management target. A view needs to be defined only once, and can then be used as any combination of read, write, or notify views when creating groups.

To define a new view, use the **snmp set view** command. For a detailed explanation of the **snmp set view** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual*. The example below illustrates how to define a view named “vrrp” that will give access to the 1.3.6.1.2.1.68 subtree (vrrpMIB). In this example, the vrrp view gives access to only the vrrpMIB. All other management information is excluded from the view. Groups can be assigned read, write, and/or notify access to the vrrp view when the groups are created.

```
router(config)# snmp set view vrrp subtree 1.3.6.1.2.1.68 type include
```

The next example illustrates the use of multiple subtrees and the **exclude** keyword to restrict access to a particular subtree. In this example, “myview” grants access to all of the Internet OID space *excluding* the 1.3.6.1.6.3.18 subtree (snmpCommunityMIB). If assigned to a group, this view can give read, write, and/or notify access to all of the Internet view except the snmpCommunityMIB.

```
router(config)# snmp set view myview subtree 1.3.6.1 type include
router(config)# snmp set view myview subtree 1.3.6.1.6.3.18 type exclude
```

Using Masks

The use of masks allows for complex selection of subtrees without needing to specify potentially several dozen **snmp set view** commands. Masks are particularly useful for selecting a particular row from a table. The example below uses the mask 0xff:bf to prevent access to instance 35 of the ifTable. The bits of the mask are used to indicate which bytes of the subtree OID are significant. A one in the mask indicates a significant byte in the OID while a zero indicates an insignificant, or "wild card" byte. Written out in bit notation the mask is: 1111 1111 1011 1111. Notice that the zero (10th bit) matches up with the column header of the ifTable OID (10th byte). Combined with the subtree 1.3.6.1.2.1.2.2.1.1.35, the zero has the effect of selecting ALL columns in the ifTable, while the trailing ones select ONLY the 35th row of the table. Without masks, this could only be accomplished by entering **snmp set view** commands for each of the 22 columns of the ifTable.

```
router(config)# snmp set view myview subtree 1.3.6.1.2.1.2.2 type include
router(config)# snmp set view myview subtree 1.3.6.1.2.1.2.2.1.1.35 mask 0xff:bf type exclude
```

Note that when creating a mask, as in the previous example, there will not always be enough bits to completely fill the last byte. In such a case the remaining bits should be padded with ones.

Defining Targets

Use the **snmp set target** command to define which management targets should receive notifications when events occur. For a detailed explanation of the **snmp set target** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual.*

Notifications can be sent in the form of either traps or informs. Traps are unreliable because the receiver does not send acknowledgments when it receives traps. The sender cannot determine if the traps were received. However, an SNMP entity that receives an inform request acknowledges the message with an SNMP response PDU. If the sender never receives the response, the inform request can be sent again. Thus, informs are more likely to reach their intended destination. However, informs consume more resources in the agent and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform request must be held in memory until a response is received or the request times out. Also, traps are sent only once, while an inform may be retried several times. The retries increase traffic and contribute to a higher overhead on the network.

The example below illustrates how to define an SNMPv1 target named "manager" whose IP address is 10.10.10.10 and can receive traps. Any event that generates a notification will cause a trap to be sent to the target. The X-Pedition router will use the security-name string "public" to authenticate with the SNMPv1 target.

```
router(config)# snmp set target manager ip-address 10.10.10.10 security-name public
```

The next example shows how to define a target named “foo” using SNMPv3 user-based authentication rather than security-name strings. In this context, the **security-name** option refers to the name of the user which will be used when communicating with the target. This example assumes that the user “jane” already exists.

```
router(config)# snmp set target foo ip-address 10.10.10.10 v3 auth security-name jane
```

The **type** option can be used to specify that informs should be sent to the target rather than traps. The sending of informs does require that an SNMPv3 user be created using the proper authoritative SNMP Engine-ID (see *Configuring Informs* on page 463 for more information). Again this example assumes that the user “jane” has already been created with the appropriate SNMP Engine-ID.

```
router(config)# snmp set target foo ip-address 10.10.10.10 v3 auth security-name jane type inform
```

Additionally, a target-params entry can be utilized (see *Configuring Target Parameters* on page 461) to apply the same configuration parameters to multiple targets. This can be accomplished via the **param** option. The following example illustrates the use of the **param** option. When used, the security model, security level, and security-name are all obtained from the specified target-params entry. In this case, both “foo” and “bar” will use the same security model, security level, and security-name. This example assumes that the “v3targets” target-params entry already exists.

```
router(config)# snmp set target foo ip-address 10.10.10.10 param v3targets
router(config)# snmp set target bar ip-address 10.10.10.11 param v3targets
```

Configuring Target Parameters

As mentioned in the previous section, target-parameters allow several targets to share the same security model, security level, and security-name. This can be useful if all of your targets are going to utilize the same security model. In this way, the security parameters need to only be specified once, and later if a parameter must be changed, it needs to only be changed once rather than needing to be changed for every target. To configure a set of target-parameters to be shared by multiple targets, use the **snmp set target-params** command. For a more detailed explanation of the **snmp set target-params** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual*.

The following example shows how to create a set of target-parameters named “global” and make use of those parameters in defining multiple targets. Both targets “foo” and “bar” will utilize the SNMPv1 security model and the “public” security-name string.

```
router(config)# snmp set target foo ip-address 10.10.10.10 param global
router(config)# snmp set target bar ip-address 10.10.10.11 param global
router(config)# snmp set target-params global v1 security-name public
```

Additionally, the **filter** option allows notification filtering to be applied to a group of targets. For more information on notification filters, see *Creating Notification Filters* on page 462.

Creating Notification Filters

The **snmp set filter** command facilitates the creation of notification filters. Notification filters can prevent specific types of notifications from being sent to a given group of management targets. Once the filter is created using the **snmp set filter** command, it is associated with targets via the **filter** option of the **snmp set target-params** command. For a detailed explanation of the various options available with the **snmp set filter** command, see the *Enterasys X-Pedition Native Command Line Interface Reference Manual*.

The **category** and **subtree** options are used to specify which notifications to filter. The **category** option acts as a shortcut for specifying common notification subtrees. When used with the **category** option, the **subtree** option is not available. As with view subtrees, filter subtrees can also be modified using the **mask** option (see the explanation on masking in *Defining Views* on page 459 for more information).

One important point that should be mentioned is that subtrees that are **included** in a filter will be *filtered out*. In other words, a subtree that is included in the filter will not have its notifications sent to the associated targets. Subtrees that are **excluded** from a filter will be the *only* subtrees that will be left *unfiltered* and have notifications sent.

The example below illustrates the use of a notification filter named “noVRRP” to prevent the target “foo” from receiving any VRRP notifications. Here we use the pre-defined “vrrp” category to specify the OID subtree to include in the filter.

```
router(config)# snmp set filter noVRRP category vrrp type included
router(config)# snmp set target foo ip-address 10.10.10.10 param bar
router(config)# snmp set target-params bar filter noVRRP
```

This next example illustrates the use of multiple subtrees to filter out all notifications except BGP and OSPF notifications. Here, the “routing” filter will prevent the target “foo” from receiving any notifications from sources other than **bgp** and **ospf**.

```
router(config)# snmp set filter routing category bgp type excluded
router(config)# snmp set filter routing category ospf type excluded
router(config)# snmp set target foo ip-address 10.10.10.10 param bar
router(config)# snmp set target-params bar filter routing
```

Configuring Informs

As mentioned in *Defining Targets* on page 460, informs provide a more reliable means by which to notify SNMP managers of events that occur at an SNMP agent. However, informs require that the inform request be sent using the manager's SNMP Engine-ID as the authoritative ID. This means that a user needs to be configured using the manager's Engine-ID as well as the local Engine-ID.

To configure the X-Pedition router to send SNMPv3 informs follow the steps outlined below.

1. Create a local user with appropriate notify access.
2. Configure the user with the necessary credentials for authenticating with the manager.
3. Create an SNMPv3 target configured to use the user for authentication.

Following is an example configuration using the user "Informer" for sending inform requests to the SNMP manager with the SNMP Engine-ID `0x00:11:22:33:44:55:66:77:88:99:aa:bb`. The user "Informer" will then be used to authenticate with the manager using the password "foo" when an inform is sent.

```
xp(config)# snmp set user Informer engine-id local
xp(config)# snmp set user Informer engine-id 0x00:11:22:33:44:55:66:77:88:99:aa:bb auth sha1
auth-password foo
xp(config)# snmp set group InformSenders v3 noauth notify all
xp(config)# snmp set user-to-group Informer to InformSenders
xp(config)# snmp set target InformTarget ip-address 10.10.10.10 v3 auth security-name Informer
type inform
```

Note that the user has been configured to work on both the remote manager and on the local system. When communicating with the manager, the user will use the HMAC-SHA-96 authentication protocol, but on the local system the user does not require authentication. However, the user has only been assigned notify access on the local system, so the user account cannot be used to read or write to or from the local system. Additionally, note that the user must also exist in the manager's SNMP configuration database in order for the user to successfully authenticate with the manager.

Configuration Notes

Built-In Users

Out-of-the-box the X-Pedition router is shipped with a "very-secure" configuration as described in RFC 2574 APPENDIX A. In other words, there are no built-in users configured. This means that at least one user will need to be configured locally using the X-Pedition router's CLI commands before more users can be created remotely using SNMP and the USM "clone-from" mechanism.

Built-In Views

As noted above, the X-Pedition router is shipped with no built-in users configured. However, there are built-in views based on the recommended initial configuration for "initial-semi-security-configuration" defined in RFC 2575 APPENDIX A. Since no users exist initially, the X-Pedition router will still be "very-secure" yet easy to configure since a built-in group and built-in views are already provided. The built-in entries are permanent -- in other words they cannot be deleted. They can, however, be modified to either increase or decrease the security they provide. The entries can be modified by using the CLI commands available on the X-Pedition router, or remotely by SNMP. If these entries are modified by SNMP, a corresponding CLI command will be saved in the startup configuration.

Assigning Aliases to Interfaces

The ability to assign an alias to an interface extends the set of supported SNMP objects to include the ifAlias column of the ifXTable. Used primarily for administrative purposes, the ifAlias SNMP object allows you to assign additional identification information to any interface handled by the ifXTable (i.e., physical ports, IP interfaces, IPX interfaces, VLANs, and SmartTRUNKs). The X-Pedition router allows one alias assignment per interface and limits each alias to a maximum length of 64 characters. You must use a remote SNMP manager to view, add, change, or delete an alias.

To assign an alias, enter the following from configuration mode:

```
router(config)# snmp set if-alias <interface-name> alias <alias-name>
```

Note: You cannot remove an interface until you remove any alias assigned to the interface. The X-Pedition router displays interface names up to 32 characters in length.

Examples

The following example demonstrates how to assign the alias "spare-port" to the physical port et.4.8.

```
router(config)# snmp set if-alias et.4.8 alias spare-port
```

The example below illustrates how to assign the alias "NEWALIAS" to the newly created "NEWIF" IP interface.

```
xp(config)# interface create ip NEWIF address-netmask 10.0.0.1/8 port et.1.1
xp(config)# snmp set if-alias NEWIF alias NEWALIAS
```

This example shows how to remove an alias:+

```
xp(config)# no snmp set if-alias
```

Layer-2 Security Filters

Layer-2 security filters on the X-Pedition router allow you to configure ports to filter specific MAC addresses. When defining a Layer-2 security filter, you specify to which ports you want the filter to apply. You can specify the following security filters:

- Address filters

  These filters block traffic based on the frame's source MAC address, destination MAC address, or both source and destination MAC addresses in flow bridging mode. Address filters are always configured and applied to the input port.

- Port-to-address lock filters

  These filters prohibit a user connected to a locked port or set of ports from using another port.

- Static entry filters

  These filters allow or force traffic to go to a set of destination ports based on a frame's source MAC address, destination MAC address, or both source and destination MAC addresses in flow bridging mode. Static entries are always configured and applied at the input port.

- Secure port filters

  A secure filter shuts down access to the X-Pedition router based on MAC addresses. All packets received by a port are dropped. When combined with static entries, however, these filters can be used to drop all received traffic but allow some frames to go through.

Configuring Layer-2 Address Filters

If you want to control access to a source or destination on a per-MAC address basis, you can configure an address filter. Address filters are always configured and applied to the input port. You can set address filters on the following:

- A source MAC address, which filters out any frame coming from a specific source MAC address
- A destination MAC address, which filters out any frame destined to specific destination MAC address
- A flow, which filters out any frame coming from a specific source MAC address that is also destined to a specific destination MAC address

To configure Layer-2 address filters, enter the following commands in Configure mode:

Configure a source MAC based address filter.	**filters add address-filter name** <*name*> **source-mac** <*MACaddr*> **source-mac-mask** <*mask*> **vlan** <*VLAN-num*> **in-port-list** <*port-list*>
Configure a destination MAC based address filter.	**filters add address-filter name** <*name*> **dest-mac** <*MACaddr*> **dest-mac-mask** <*mask*> **vlan** <*VLAN-num*> **in-port-list** <*port-list*>
Configure a Layer-2 flow address filter.	**filters add address-filter name** <*name*> **source-mac** <*MACaddr*> **source-mac-mask** <*mask*> **dest-mac** <*MACaddr*> **dest-mac-mask** <*mask*> **vlan** <*VLAN-num*> **in-port-list** <*port-list*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

Configuring Layer-2 Port-to-Address Lock Filters

Port address lock filters allow you to bind or “lock” specific source MAC addresses to a port or set of ports. Once a port is locked, only the specified source MAC address is allowed to connect to the locked port and the specified source MAC address is not allowed to connect to any other ports.

To configure Layer-2 port address lock filters, enter the following commands in Configure mode:

Configure a port address lock filter.	**filters add port-address-lock name** <*name*> **source-mac** <*MACaddr*> **vlan** <*VLAN-num*> **in-port-list** <*port-list*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

Configuring Layer-2 Static-Entry Filters

Static entry filters allow or force traffic to go to a set of destination ports based on a frame's source MAC address, destination MAC address, or both source and destination MAC addresses in flow bridging mode. Static entries are always configured and applied at the input port. You can set the following static entry filters:

Note: You may include up to 505 static-entry filters in a configuration.

- Source static entry, which specifies that any frame coming from source MAC address will be allowed or disallowed to go to a set of ports
- Destination static entry, which specifies that any frame destined to a specific destination MAC address will be allowed, disallowed, or forced to go to a set of ports
- Flow static entry, which specifies that any frame coming from a specific source MAC address that is destined to specific destination MAC address will be allowed, disallowed, or forced to go to a set of ports

To configure Layer-2 static entry filters, enter the following commands in Configure mode:

Configure a source static entry filter.	**filters add static-entry name** <*name*> **restriction allow|disallow|force source-mac** <*MACaddr*> **source-mac-mask** <*mask*> **vlan** <*VLAN-num*> **in-port-list** <*port-list*> **out-port-list** <*port-list*>
Configure a destination static entry filter.	**filters add static-entry name** <*name*> **restriction allow|disallow|force dest-mac** <*MACaddr*> **dest-mac-mask** <*mask*> **vlan** <*VLAN-num*> **in-port-list** <*port-list*> **out-port-list** <*port-list*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

Configuring Layer-2 Secure Port Filters

Secure port filters block access to a specified port. You can use a secure port filter by itself to secure unused ports. Secure port filters can be configured as source or destination port filters. A secure port filter applied to a source port forces all incoming packets to be dropped on a port. A secure port filter applied to a destination port prevents packets from going out a certain port.

You can combine secure port filters with static entries in the following ways:

- Combine a source secure port filter with a source static entry to drop all received traffic but allow any frame coming from specific source MAC address to go through.
- Combine a source secure port filter with a flow static entry to drop all received traffic but allow any frame coming from a specific source MAC address that is destined to specific destination MAC address to go through.
- Combine a destination secure port with a destination static entry to drop all received traffic but allow any frame destined to specific destination MAC address go through.
- Combine a destination secure port filter with a flow static entry to drop all received traffic but allow any frame coming from specific source MAC address that is destined to specific destination MAC address to go through.

To configure Layer-2 secure port filters, enter the following commands in Configure mode:

Configure a source secure port filter.	**filters add secure-port name** <*name*> **direction source vlan** <*VLAN-num*> **in-port-list** <*port-list*>
Configure a destination secure port filter.	**filters add secure-port name** <*name*> **direction destination vlan** <*VLAN-num*> **in-port-list** <*port-list*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

Monitoring Layer-2 Security Filters

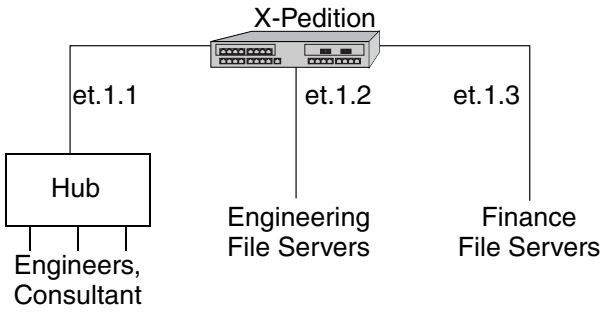
The X-Pedition router provides display of Layer-2 security filter configurations contained in the routing table. To display security filter information, enter the following commands in Enable mode.

Show address filters.	**filters show address-filter [all-source|all-destination|all-flow]** [**source-mac** <*MACaddr*> **dest-mac** <*MACaddr*>] [**ports** <*port-list*>] [**vlan** <*VLAN-num*>]
Show port address lock filters.	**filters show port-address-lock ports** [**ports** <*port-list*>] [**vlan** <*VLAN-num*>] [**source-mac** <*MACaddr*>]
Show secure port filters.	**filters show secure-port**
Show static entry filters.	**filters show static-entry** [**all-source|all-destination|all-flow**] **ports** <*port-list*> **vlan** <*VLAN-num*> [**source-mac** <*MACaddr*> **dest-mac** <*MACaddr*>]

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

Layer-2 Filter Examples

Figure 41. Source Filter Example



Example 1: Address Filters

Source filter: The consultant is not allowed to access any file servers. The consultant is only allowed to interact with the engineers on the same Ethernet segment – port et.1.1. All traffic coming from the consultant's MAC address will be dropped.

```
filters add address-filter name consultant source-mac 001122:334455 vlan 1 in-port-list et.1.1
```

Destination filter: No one from the engineering group (port et.1.1) should be allowed to access the finance server. All traffic destined to the finance server's MAC will be dropped.

```
filters add address-filter name finance dest-mac AABBCC:DDEEFF vlan 1 in-port-list et.1.1
```

Flow filter: Only the consultant is restricted access to one of the finance file servers. Note that port et.1.1 should be operating in flow-bridging mode for this filter to work.

```
filters add address-filter name consult-to-finance source-mac 001122:334455 dest-mac
    AABBCC:DDEEFF vlan 1 in-port-list et.1.1
```

Static Entries Example

Source static entry: The consultant is only allowed to access the engineering file servers on port et.1.2.

```
filters add static-entry name consultant source-mac 001122:334455 vlan 1 in-port-list et.1.1 out-port-list
    et.1.2 restriction allow
```

Destination static entry: Restrict “login multicasts” originating from the engineering segment (port et.1.1) from reaching the finance servers.

```
filters add static-entry name login-mcasts dest-mac 010000:334455 vlan 1 in-port-list et.1.1 out-port-list
    et.1.3 restriction disallow
```

or

```
filters add static-entry name login-mcasts dest-mac 010000:334455 vlan 1 in-port-list et.1.1 out-port-list
    et.1.2 restriction allow
```

Flow static entry: Restrict “login multicasts” originating from the consultant from reaching the finance servers.

```
filters add static-entry name consult-to-mcasts source-mac 001122:334455 dest-mac 010000:334455 vlan
    1 in-port-list et.1.1 out-port-list et.1.3 restriction disallow
```

Port-to-Address Lock Examples

You have configured some filters for the consultant on port et.1.1 If the consultant plugs his laptop into a different port, he will bypass the filters. To lock him to port et.1.1, use the following command:

```
filters add port-address-lock name consultant source-mac 001122:334455 vlan 1 in-port-list et.1.1
```

Note: If the consultant’s MAC is detected on a different port, all of its traffic will be blocked.

Example 2: Secure Ports

Source secure port: To block all engineers on port 1 from accessing all other ports, enter the following command:

```
filters add secure-port name engineers direction source vlan 1 in-port-list et.1.1
```

To allow ONLY the engineering manager access to the engineering servers, you must “punch” a hole through the secure-port wall. A “source static-entry” overrides a “source secure port”.

```
filters add static-entry name eng-mgr source-mac 080060:123456 vlan 1 in-port-list et.1.1 out-port-list
    et.1.2 restriction allow
```

Destination secure port: To block access to all file servers on all ports from port et.1.1 use the following command:

```
filters add secure-port name engineers direction dest vlan 1 in-port-list et.1.1
```

To allow all engineers access to the engineering servers, you must “punch” a hole through the secure-port wall. A “dest static-entry” overrides a “dest secure port”.

```
filters add static-entry name eng-server dest-mac 080060:abcdef vlan 1 in-port-list et.1.1 out-port-list et.1.2
    restriction allow
```

Layer-3 Security Controls

Access Control Lists (ACLs)

Access Control Lists (ACLs) allow you to restrict Layer-3/4 traffic going through the X-Pedition router. Each ACL consists of one or more rules describing a particular type of IP or IPX traffic. An ACL can be simple, consisting of only one rule, or complicated with many rules. Each rule tells the router to either permit or deny the packet that matches the rule’s packet description. For information about defining and using ACLs on the X-Pedition router, see *Access Control List Configuration Guide* on page 419.

Note: You may not apply ACLs to interface EN0 of the control module.

Rate Limiting

This configuration mode command allows the X-Pedition router to set limits on the amount of traffic any port, interface, or VLAN can receive.

Note: Rate limiting cannot be applied to IPv6 interfaces.

Features Available

There are five different kinds of rate limiting: per flow rate limiting, aggregate rate limiting, port-based rate limiting, flow aggregate rate limiting, and VLAN rate limiting (L4-Bridging must be enabled for VLAN rate limiting). Per flow and aggregate rate limiting have flows defined through a policy ACL—VLAN and port-based rate limiting do not. For a more in-depth description of the various kinds of rate limiting, refer to *Limiting Traffic Rate* on page 496.

Note: There are two rate limiting modes that can be set where appropriate: per flow (the default) and aggregate.

Note: Aggregate and flow-aggregate rate limiting are not supported on 802.1q trunk ports.

A big problem in security today is Denial of Service or DoS. DoS attacks are used to prevent normal operation of network hardware and end systems. DoS packets attempt to congest a network and render it useless. The X-Pedition router is resistant to many of these attacks, but some attacks may still cause problems—even attack packets being handled by a flow. By using rate limiting, traffic (good or bad) can be handled appropriately and network performance maintained.

Configuration

The rate limiting commands are available in configuration mode. If you know what kinds of traffic loads the X-Pedition router can expect, you can configure the router with rate limiting. Rate limiting can be applied to the following:

- A set of ports
- A flow
- A set of flows (aggregate)
- A VLAN (L4-Bridging must be enabled for VLAN rate limiting)

To configure rate limiting policies, enter the following commands while in configuration mode:

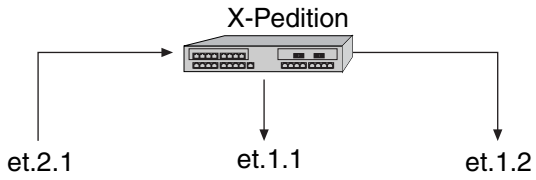
Enable aggregate rate limiting mode on the X-Pedition router.	**system enable aggregate-rate-limiting**
Define a per-flow rate limit policy.	**rate-limit** <*name*> **input acl** <*acl list*> **rate** <*number*> **exceed-action drop-packets** | **set-priority-low** | **set-priority-medium** | **set-priority-high** [**sequence** <*number*>] | [**burst-compensating**]

Apply a per-flow rate limit profile to an interface.	**rate-limit** <*name*> **apply interface** <*interface*> | **all**
Define a port rate limit policy to limit incoming traffic on a port.	**rate-limit** <*name*> **port-level input port** <*port list*> **rate** <*num*> [**drop-packets** | **no-action** | **lower-priority** | **lower-priority-except-control** | **tos-precedence-rewrite** <*num*> | **tos-precedence-rewrite-lower-priority** <*num*>]
Define a port rate limit policy to limit outgoing traffic on a port.	**rate-limit** <*name*> **port-level output port** <*port list*> **rate** <*num*> **drop-packets**
Specify that control traffic is not subject to output port rate limiting.	**rate-limit** <*name*> **port-level slot** <*num*> **ignore-control-priority**
Define an aggregate rate limit policy.	**rate-limit** <*name*> **aggregate acl** <*acl list*> **rate** <*num*> [**drop-packets** | **no-action** | **lower-priority** | **lower-priority-except-control** | **tos-precedence-rewrite** <*num*> | **tos-precedence-rewrite-lower-priority** <*num*>] [**allocate-resources-during-apply** | **allocate-resources-during-traffic**] | [**burst-compensating**]
Apply an aggregate rate limit policy to an interface.	**rate-limit** <*name*> **apply interface** <*interface*> | **all**
Define a VLAN rate limiting policy to limit incoming traffic on a VLAN.	**rate-limit** <*name*> **vlan** <*name*> **port** <*port list*> | **all-ports destport** <*port list*> | **all-ports rate** <*num*> **exceed-action drop-packets** | **set-priority-low** | **set-priority-medium** | **set-priority-high** [**burst-compensating**]
Define a VLAN rate limiting policy to limit outgoing traffic on a port belonging to a specific VLAN.	**rate-limit** <*name*> **vlan** <*name*> **port** <*port list*> | **all-ports destport** <*port list*> | **all-ports rate** <*num*> **exceed-action drop-packets** | **set-priority-low** | **set-priority-medium** | **set-priority-high** [**burst-compensating**]
Show rate limit policy information.	**rate-limit show** [**all**] |[**policy-type flow-policies** | **flow-aggregate-policies** | **aggregate-policies** | **portlevel-policies** | **all**] | [**policy-name** <*name*>] | [**interface** <*interface*>] | [**port-level port** <*port list*> |**all-port**] | [**port-level policy-name** <*name*>] | [**rate-limiting-mode**]

Note: The X-Pedition router displays VLAN and interface names up to 32 characters in length.

Note: Aggregate and flow-aggregate rate limiting are not supported on 802.1q trunk ports.

Sample Configuration: Per-Flow Rate Limiting



Consider the following configuration. Traffic from two IPv4 interfaces, ipNet1 with IP address 10.10.10.10 and ipNet2 with IP address 20.20.20.20, will have their packet priority lowered if they exceed 10Mbs for each flow:

```
vlan create net1 ip
vlan create outNet ip
vlan create net2 ip
vlan add ports et.1.1 to net1
vlan add ports et.1.2 to net2
vlan add ports et.2.1 to outNet
interface create ip ipNet1 vlan net1 address-netmask 10.10.10.10/8
interface create ip ipNet2 vlan net2 address-netmask 20.20.20.20/8
interface create ip ipOutNet vlan outNet address-netmask 30.30.30.30/8
acl net1Acl permit ip 10.1.1.1
acl net2Acl permit ip 20.2.2.2
rate-limit rlNet1 input acl net1Acl rate 10000000 exceed-action set-priority-low
rate-limit rlNet2 input acl net2Acl rate 10000000 exceed-action set-priority-low
rate-limit rlNet1 apply interface ipNet1
rate-limit rlNet2 apply interface ipNet2
```

Output

If the X-Pedition router successfully creates a rate-limiting policy, an %RL-I-CREATED message will appear. Additionally, %RL-I-INTERFACE messages will appear if the rate limit needs to be applied. The messages displayed for the commands in the previous example are as follows:

```
%VLAN-I-CREATED, VLAN net1 created with VLAN ID 2. To add ports to the VLAN, use the “vlan
        add ports” command.
%VLAN-I-CREATED, VLAN outNet created with VLAN ID 3. To add ports to the VLAN, use the “vlan
        add ports” command.
%VLAN-I-CREATED, VLAN net2 created with VLAN ID 4. To add ports to the VLAN, use the “vlan
        add ports” command.
%VLAN-I-ADDSUCCESS, 1 port et.1.1 successfully added to VLAN net1
%VLAN-I-ADDSUCCESS, 1 port et.1.2 successfully added to VLAN net2
%VLAN-I-ADDSUCCESS, 1 port et.2.1 successfully added to VLAN outNet
%INTERFACE-I-CREATEDIF, Interface ipNet1 was successfully created.
%INTERFACE-I-CREATEDIF, Interface ipNet2 was successfully created.
%INTERFACE-I-CREATEDIF, Interface ipOutNet was successfully created.
%RL-I-CREATED, Rate limit policy 'rlNet1' has been successfully created for ACL(s) 'net1Acl'
%RL-I-CREATED, Rate limit policy 'rlNet2' has been successfully created for ACL(s) 'net2Acl'
%RL-I-INTERFACE, Rate limit policy 'rlNet1' has been successfully attached to 'ipNet1' interface
%RL-I-INTERFACE, Rate limit policy 'rlNet2' has been successfully attached to 'ipNet2' interface
```

Enable / Disable

To disable a rate limiting policy, remove it from the active configuration.

Broadcast Monitor

Broadcast Monitor or *BMON* allows users to detect and react to certain kinds of DoS attacks. BMON is useful for mitigating the damage inflicted by DoS attacks by limiting the amount of abuse to which the routers and networks attached to them are exposed. The DoS attacks addressed by BMON are those intended to attack the CPU of the router. These attacks are characterized by very large amounts of traffic whose packets are known to require CPU attention. When the CPU falls victim to such an attack, the system may experience a degradation in the performance of other aspects of its operation, such as console management and the forwarding of legitimate traffic. BMON is implemented in two parts: *detection* and *reaction*.

Detection

BMON is capable of detecting two kinds of potential DoS attacks: *broadcast traffic* and *unlearned traffic*. Broadcast traffic is composed of packets whose destination MAC address is the broadcast MAC, or ff:ff:ff:ff:ff:ff. Broadcast packets can be used in a DoS attack on a router or network because all machines that receive broadcast packets (i.e., routers, smart switches, and end hosts) must send the packet to their CPU for processing.

Unlearned traffic consists of packets whose signature (a combination of one or more fields in a packet header) does not match any signature already recognized by the router. Unlearned packets must be sent to the CPU to compute (or learn) a new route, write the route into the hardware, then use the new route to transmit the packet. All subsequent packets whose signatures match the new route will be forwarded in hardware and will not require additional CPU attention.

Reaction

The X-Pedition router keeps a running total and rate of all broadcast and unlearned packets it receives. When BMON is enabled on a given port, BMON checks these statistics periodically to see if they exceed a configured threshold. Should the count or rate exceed the configured threshold, BMON triggers a reaction.

BMON uses two types of reactions: *shutdown* and *redirect*. Shutdown is achieved by administratively bringing the associated port down. Once a port is shut down, it no longer sends or receives any traffic—because the port can no longer receive traffic, the packets that triggered the reaction are no longer sent to the CPU. Redirect sends all packets that would normally go to the CPU to another port (all other packets remain unaffected). Although users may configure both reaction types to be in a triggered state for a limited amount of time, the redirect state may be configured to be in a triggered state indefinitely. Both reactions affect only Layer-2 and Layer-3 traffic on the port(s) for which they are configured and triggered—with one exception. When the redirect reaction is triggered on a port configured for unlearned traffic, all unlearned Layer-3 traffic received by the line card that contains the redirected port is also redirected.

Shutdown versus Redirect

Typically, most users will configure BMON with redirect because of its ability to stop all DoS traffic while still allowing most legitimate traffic to continue unaffected (a shutdown stops all traffic, legitimate or not). However, because some DoS attacks are viral in nature (e.g., the SQL Slammer), systems that require an extremely high level of security may warrant a shutdown reaction.

Configuration

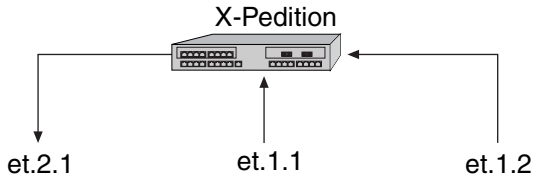
To function properly, users must configure BMON with each of the following: a port or set of ports, a set of thresholds to use for detection, and a reaction to take if the thresholds are exceeded. When using BMON’s redirect option, consider the following:

- A port cannot be redirected to itself.
- A port cannot be redirected to multiple ports; however, multiple ports can be redirected to a single port.
- A port receiving redirected traffic must be able to handle the combined load of the ports.
- When redirecting unlearned traffic, all unlearned Layer-3 traffic received by the line card containing the redirected port is also redirected.

To configure broadcast monitors, enter the following command from configuration mode (for additional information regarding command, refer to the *Enterasys X-Pedition Native Command Line Interface Reference Manual*):

Configure and attach a broadcast monitor on a port or set of ports.	**port bmon** <*port list*> **[rate** <*number*>**] [duration** <*number*>**] [expire** <*number*>**] [packet-limited all** | **broadcast] [redirect** <*port*>**] [unlimited-redirect** <*port*>**]**

Sample Configuration: Monitor All Unlearned Traffic



The following command will redirect unlearned traffic for ports et.1.1 and et.1.2 to port et.2.1 after the unlearned traffic rate reaches 2 Kpkts/sec for 30 sec. The traffic will be redirected for the default time of 5 min.

```
port bmon et.1.1-2 redirect et.2.1 rate 2 duration 30 packets-limited all
```

Output

When you enter a BMON command, no message appears. However, when you redirect a port or a port redirection expires, a %SYS-I-CPL_PORTRD or %SYS-I-CPL_PORTNRD message appears (respectively). If traffic conditions bring a port down, a %SYS-I-CPL_PORTDN message will appear. When the port resumes operation, a %SYS-I-CPL_PORTUP message will appear.

Enable / Disable

To disable bmon commands, remove them from the active configuration.

Port Mirroring

The port mirroring facility, available in configuration mode, allows the X-Pedition router to duplicate traffic between ports. For additional information, see *Configuring for Port Mirroring* on page 507.

Features Available

A port mirror can be used to duplicate traffic from a single port to another single port, a single port to multiple ports, multiple ports to a single port, or multiple ports to multiple ports. You may also use the Port Mirroring facility in conjunction with an ACL. When you set up a mirror for traffic coming into the X-Pedition router that matches a specific ACL, the router mirrors the traffic out to one or more ports.

Note: The X-Pedition router supports port mirroring, ACL, and Layer-2 filtering on a per-WAN-card basis, not a per-port basis.

Note: The X-Pedition router does not support port mirroring on ATM ports.

By itself, port mirroring does not protect the X-Pedition router from intrusion; however, it is very useful when you work in conjunction with an IDS (Intrusion Detection System) such as the Enterasys Dragon Intrusion Defense System. The X-Pedition router can mirror traffic to an IDS and the IDS can perform more sophisticated, customized processing of the packets without inhibiting the router's performance—if it finds a suspicious packet, the IDS can alert the system administrator. Most security systems have some way to communicate with routers to neutralize any potential threats (e.g., the Dragon Intrusion Defense System uses SNMP to communicate with the X-Pedition router). Refer to the Enterasys web site for more information on the Dragon product and other security offerings.

Configuration

If you want to duplicate traffic between ports, you may want set up a port mirror with the mirroring commands (available in configuration mode). Since you may use a port only once in a single mirror, you may not use a port in multiple mirrors or set it to mirror itself. Additionally, since the mirror administratively takes down destination ports in a port mirror, you cannot use them in smart trunks, ACLs, etc. You can apply mirrors to the following:

- A set of ports
- An ACL

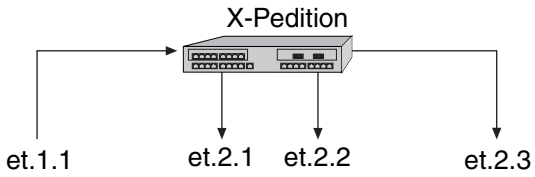
Note: The X-Pedition router does not support port mirroring on ATM ports.

To configure port mirrors, enter the following commands while in configuration mode:

Specify ports to be mirrored out another set of ports.	**port mirroring dst-ports** <*port-list*> **src-ports** <*port-list*>
Specify an ACL to be mirrored out the designated ports.	**port mirroring dst-ports** <*port-list*> **src-acl** <*acl-name*>

Sample Configurations

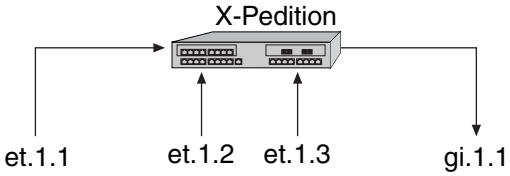
Example 1: One to Many Mirror



Port et.1.1 is being mirrored out 3 ports, et.2.1, et.2.2 and et.2.3.

```
port mirroring dst-ports et.2.(1-3) src-ports et.1.1
```

Example 2: Many To One Mirror



Ports et.1.1, et.1.2 and et.1.3 are being mirrored out gi.1.1

```
port mirroring dst-ports gi.1.1 src-ports et.1.(1-3)
```

Example 3: ACL Mirror

All traffic coming into the X-Pedition router matching the ACL, inNet, will be sent out port gi.1.1. For information on configuring ACLs see *Access Control List Configuration Guide* on page 419.

```
port mirroring dst-ports gi.1.1 src-acl inNet
```

Output

If the X-Pedition router creates a mirror successfully, it will print a message to the console. The message for example 1 would be as follows:

```
MIRRORING-I-MIRROR_INFO, Mirroring on ports et.2.1 et.2.2 et.2.3 for et.1.1 is enabled
```

Enable / Disable

To disable a port mirror, remove it or comment it out of the active configuration.

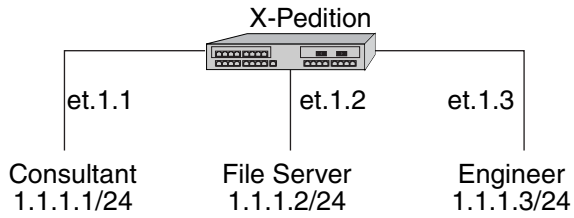
Layer-4 Bridging and Filtering

Layer-4 bridging is the X-Pedition router's ability to use layer-3/4 information to perform filtering or QoS during bridging. As described in *Layer-2 Security Filters* on page 465, you can configure ports to filter traffic using MAC addresses. Layer-4 bridging adds the ability to use IP addresses, layer-4 protocol type, and port number to filter traffic in a bridged network. Layer-4 bridging allows you to apply security filters on a "flat" network, where the client and server may reside on the same subnet.

Note: Layer-4 bridging is allowed on IP, IPv6, or IPX VLANs only.

To illustrate this, the following diagram shows an X-Pedition router serving as a bridge for a consultant host, file server, and an engineering host, all of which reside on a single subnet.

Figure 42. Sample VLAN for Layer-4 bridging



You may want to allow the consultant access to the file server for e-mail (SMTP) traffic, but not for Web (HTTP) traffic and allow e-mail, Web, and FTP traffic between the engineer and the file server. You can use Layer-4 bridging to set this up. Setting up Layer-4 bridging consists of the following steps:

- Creating an IP, IPv6, or IPX VLAN
- Placing the ports on the same VLAN
- Enabling Layer-4 Bridging on the VLAN
- Creating an ACL that specifies the selection criteria
- Applying an ACL to a port

Creating an IP, IPv6 or IPX VLAN for Layer-4 Bridging

The ports to be used in Layer-4 Bridging must all be on the same VLAN. To create an IP, IPv6, or IPX VLAN, enter the following command in Configure mode:

Create an IP VLAN.	**vlan create** <*vlan-name*> **ip id** <*num*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

For example, to create an IP VLAN called “blue” with an ID of 21, enter the following command in Configure Mode:

```
xp(config)# vlan create blue ip id 21
```

Placing the Ports on the Same VLAN

Once you create a VLAN for the ports you will use in layer-4 bridging, add the ports to the VLAN by entering the following command in Configure Mode:

Add ports to a VLAN.	**vlan add ports** <*port-list*> **to** <*vlan-name*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

To add the ports in the example in Figure 42 on page 479 to the blue VLAN, enter the following command:

```
xp(config)# vlan add ports et.1.1,et.1.2,et.1.3 to blue
```

Enabling Layer-4 Bridging on the VLAN

After you add the ports to the VLAN, enable Layer-4 Bridging on the VLAN by entering the following command in Configure Mode:.

Enable Layer-4 bridging.	**vlan enable l4-bridging on** <*vlan-name*>

Note: The X-Pedition router displays VLAN names up to 32 characters in length.

For example, to enable Layer-4 Bridging on the blue VLAN:

```
xp(config)# vlan enable l4-bridging on blue
```

Creating ACLs to Specify Selection Criteria for Layer-4 Bridging

Access control lists (ACLs) specify the kind of filtering to be done for Layer-4 Bridging.

In the example in Figure 42 on page 479, to allow the consultants access to the file server for e-mail (SMTP) traffic, but not for Web (HTTP) traffic—and allow e-mail, Web, and FTP traffic between the engineers and the file server, you would create ACLs that allow only SMTP traffic on the port to which the consultants are connected and allow SMTP, HTTP, and FTP traffic on the ports to which the engineers are connected.

The following is an example:

```
acl 100 permit ip any any smtp
acl 100 deny ip any any http

acl 200 permit any any smtp
acl 200 permit any any http
acl 200 permit any any ftp
```

ACL 100 explicitly permits SMTP traffic and denies HTTP traffic. Note that because of the implicit deny rule appended to the end of the ACL, all traffic (not just HTTP traffic) other than SMTP is denied.

ACL 200 explicitly permits SMTP, HTTP, and FTP traffic. The implicit deny rule denies any other traffic. See *Creating ACLs* on page 425 for more information on defining ACLs.

Applying a Layer-4 Bridging ACL to a Port

Finally, you apply the ACLs to the ports in the VLAN. To do this, enter the following command in Configure Mode:

Apply a Layer-4 bridging ACL to a port	**acl** <*name*> **apply port** <*port-list*>

For the example in Figure 42 on page 479, to apply ACL 100 (which denies all traffic except SMTP) to the consultant port:

```
xp(config)# acl 100 apply port et.1.1 output
```

To apply ACL 200 (which denies all traffic except SMTP, HTTP, and FTP) to the engineer port:

```
xp(config)# acl 200 apply port et.1.3 output
```

Notes

- Layer-4 Bridging works for IP, IPv6, and IPX traffic only. The X-Pedition router will drop non-IP/IPv6/IPX traffic on a Layer-4 Bridging VLAN. For Appletalk and DECnet packets, a warning is issued before the first packet is dropped.
- If you use a SmartTRUNK with a Layer-4 Bridging VLAN, the X-Pedition router maintains the packet order on a per-flow basis, rather than per-MAC pair. This means that for traffic between a MAC pair consisting of more than one flow, the packets may be disordered if they go through a SmartTRUNK. For traffic that doesn't go through a SmartTRUNK, the per-MAC pair packet order is kept.
- ACLs applied to a network interface (as opposed to a port) do not have an effect on Layer-4 Bridged traffic, even though the interface may include ports used in Layer-4 Bridging.
- When you use an SNMP MIB to create a VLAN, Layer-4 Bridging is auto-enabled.
- If you use Q MIB to add ports on line cards, Layer-4 Bridging will be disabled.
- You may not apply ACLs to interface EN0 of the control module.

Chapter 27

QoS Configuration Guide

QoS, Layer-2, Layer-3, and Layer-4 Flow Overview

The X-Pedition router allows network managers to identify traffic and set Quality of Service (QoS) policies without compromising wire speed performance. The X-Pedition router can guarantee bandwidth on an application by application basis, thus accommodating high-priority traffic even during peak periods of usage. QoS policies can be broad enough to encompass all the applications in the network, or relate specifically to a single host-to-host application flow.

The X-Pedition router provides four different features to satisfy QoS requirements:

- *Traffic prioritization* allows network administrators to identify and segregate mission-critical network traffic into different priority queues from non-critical network traffic. Once a packet has been identified, it can be assigned into any one of four priority queues in order to ensure delivery. Priority can be allocated based on any combination of Layer-2, Layer-3, or Layer-4 traffic.
- *Weighted Random Early Detection* (WRED) alleviates traffic congestion by randomly dropping packets before the queue becomes completely flooded. WRED should only be applied to TCP traffic.
- *Type of Service (ToS) rewrite* provides network administrators access to the ToS octet in an IP packet. The ToS octet is designed to provide feedback to the upper layer application. The administrator can "mark" packets using the ToS rewrite feature so that the application (a routing protocol, for example) can handle the packet based on a predefined mechanism.
- *Traffic rate limiting* provides network administrators with tools to manage bandwidth resources. The administrator can create an upper limit for a traffic profile, which is based on Layer-3 or Layer-4 information. Traffic that exceeds the upper limit of the profile can either be dropped or re-prioritized into another priority queue.

Within the X-Pedition router, QoS policies are used to classify Layer-2, Layer-3, and Layer-4 traffic into the following priority queues (in order from highest priority to lowest):

- Control (for router control traffic; the remaining classes are for normal data flows)
- High
- Medium
- Low

Separate buffer space is allocated to each of these four priority queues. By default, buffered traffic in higher priority queues is forwarded ahead of pending traffic in lower priority queues (this is the *strict priority* queuing policy). During heavy loads, low-priority traffic can be dropped to preserve the throughput of the higher-priority traffic. This ensures that critical traffic will reach its destination even if the exit ports for the traffic are experiencing greater-than-maximum utilization. To prevent low-priority traffic from waiting indefinitely as higher-priority traffic is sent, you can apply the *weighted fair queuing* (WFQ) queuing policy to set a minimum bandwidth for each class. You can also apply *weighted random early detection* (WRED) to keep congestion of TCP traffic under control.

Queuing Policies

You can use one of two queuing policies on the X-Pedition router:

- **Strict priority**: Assures the higher priorities of throughput but at the expense of lower priorities. For example, during heavy loads, low-priority traffic can be dropped to preserve throughput of control-priority traffic, and so on.
- **Weighted fair queuing**: Distributes priority throughput among the four priorities (control, high, medium, and low) based on percentages.

You can set the queuing policy on a per-port basis. The default queuing policy is strict priority.

Layer-2, Layer-3, and Layer-4 Flow Specification

In the X-Pedition router, traffic classification is accomplished by mapping Layer-2, -3, or -4 traffic to one of the four priorities. Each traffic classification is treated as an individual traffic flow in the X-Pedition router. The flows specify the contents of these fields. If you do not enter a value for a field, a wildcard value (all values acceptable) is assumed for the field.

Layer-2

For Layer-2 traffic, you can define a flow based on the following:

- MAC packet header fields, including source MAC address, destination MAC address and VLAN IDs. A list of incoming ports can also be specified.

Layer-3

For Layer-3 (IP and IPX) traffic you can define flows, blueprints, or templates of IP and IPX packet headers.

- The IP fields are source IP address, destination IP address, UDP/TCP source port, UDP/TCP destination port, TOS (Type of Service), transport protocol (TCP or UDP), and a list of incoming interfaces.
- The IPX fields are source network, source node, destination network, destination node, source port, destination port, and a list of incoming interfaces.

Precedence for Layer-3 Flows

A precedence from 1 - 7 is associated with each field in a flow. The X-Pedition router uses the precedence value associated with the fields to break ties if packets match more than one flow. The highest precedence is 1 and the lowest is 7. Here is the default precedence of the fields:

- IP: destination port (1), destination IP address (2), source port (3), source IP address (4), TOS (5), interface (6), protocol (7)
- IPX: destination network (1), source network (2), destination node (3), source node (4), destination port (5), source port (6), interface (7)

Use the **qos precedence ip** and **qos precedence ipx** commands to change the default precedence.

Layer-4

For Layer-4 traffic, you can define a flow based on source/destination TCP/UDP port number in addition to Layer-3 source/destination IP address.

Traffic Prioritization

Layer-2 Flows

QoS policies applied to layer-2 flows allow you to assign priorities based on source and destination MAC addresses. A QoS policy set for a layer-2 flow allows you to classify the priority of traffic from:

- A specific source MAC address to a specific destination MAC address (use only when the port is in flow bridging mode)
- Any source MAC address to a specific destination MAC address

Before applying a QoS policy to a layer-2 flow, you must first determine whether a port is in address-bridging mode or flow-bridging mode. If a port operates in address-bridging mode (default), you can specify the priority based on the destination MAC address and a VLAN ID. You can also specify a list of ports to apply the policy. If a port operates in flow-bridging mode, you can be more specific and configure priorities for frames that match both a source AND a destination MAC address and a VLAN ID. You can also specify a list of ports to apply the policy.

The VLAN ID in the QoS configuration must match the VLAN ID assigned to the list of ports to which the QoS policy is applied. In a Layer-2 only configuration, each port has only one VLAN ID associated with it and the QoS policy should have the same VLAN ID. When different VLANs are assigned to the same port using different protocol VLANs, the layer-2 QoS policy must match the VLAN ID of the protocol VLAN.

Note: In flow mode, you can also ignore the source MAC address and configure the priority based on the destination MAC address only.

Configuring Layer-2 QoS

When applying QoS to a Layer-2 flow, priority can be assigned as follows:

- The frame gets assigned a priority within the switch. Select “low,” medium, high or control.”
- The frame gets assigned a priority within the switch, AND if the exit ports are VLAN trunk ports, the frame is assigned an 802.1Q priority. Select a number from 0 to 7.

Note: A packet entering a Q-trunk has an 802.1Q header containing a priority field. Typically, users can change the 802.1Q priority using the **qos set l2** commands. However, current hardware restrictions ignore any request to overwrite the packet’s priority—the packet simply replicates at the exit port and continues with its original priority.

To set a QoS priority on a layer-2 flow, enter the following command in Configure mode:

Set a Layer-2 QoS policy.	**qos set l2 name** <*name*> **source-mac** <*MACaddr*> **dest-mac** <*MACaddr*> **vlan** <*vlanID*> **in-port-list** <*port-list*> **priority control|high|medium|low|**<*trunk-priority*>

802.1p Priority Mapping

The following table shows the default mapping of 802.1p class of service (CoS) values to internal priorities for frames that are received on a port on the X-Pedition router:

Table 13. Default Priority Map

802.1p CoS Values	Internal Priority Queue
0, 1	Low
2, 3	Medium
4, 5	High
6, 7	Control

You can create one or more priority maps that are different from the default priority map and then apply these maps to some or all ports of the X-Pedition router. The new priority mapping replaces the default mappings for those ports to which they are applied.

Creating and Applying a New Priority Map

To specify a priority map on a per-port basis, enter the following commands in Configure mode:

Create a new priority mapping.	**qos create priority-map** <*name*> <*CoS number*> **control|high|medium|low**
Apply new priority mapping to ports.	**qos apply priority-map** <*name*> **ports** <*port-list*>

For example, the following command creates the priority map “all-low” which maps all 802.1p priorities to the “low” internal priority queue:

```
qos create priority-map all-low 0 low 1 low 2 low 3 low 4 low 5 low 6 low 7 low
```

Once a priority map is created, it can then be applied to a set of ports, as shown in the following example:

```
qos apply priority-map all-low ports et.1.1-4, gi.4.*
```

In the above example, ports et.1.1-4 and ports gi.4.* will use the “all-low” priority map. All other ports, including ports et.1.5-8, will use the default priority map.

You do not need to specify mappings for *all* 802.1p values. If you do not specify a particular mapping, the default mapping for that 802.1p priority is used. The following example creates a priority map “no-ctrl” with the same mappings as the default priority map, except that the 802.1p priority of 7 is mapped to the internal priority “high” instead of “control.”

```
qos create priority-map no-ctrl 7 high
```

Removing or Disabling Per-Port Priority Map

Negating a **qos create priority-map** command removes the priority map. (Before you can remove a priority map, you must negate all commands that use the priority map.) Negating a **qos apply priority-map** command causes the configured ports to use the default priority mapping.

The ability to specify per-port priority maps is enabled on the X-Pedition router by default. You can disable use of per-port priority maps on the X-Pedition router; all ports on the X-Pedition router will then be configured to use the default priority map only. If the commands to create and apply priority maps exist in the active configuration, they will remain in the configuration but be ineffective.

To disable the use of priority maps, enter the following command in Configure mode:

Disable use of per-port priority maps on the X-Pedition router.	**qos priority-map off**

If the above command is negated, ports on the X-Pedition router can use per-port priority maps. If the commands to create and apply priority maps exist in the active configuration, they are reapplied.

Displaying Priority Map Information

To display priority maps and the ports on which they are applied, enter the following command in Enable mode:

Display priority mapping.	**qos show priority-map** <*name*>|**all**

Layer-3 & Layer-4 Flows

QoS policies applied at layer-3 and 4 allow you to assign priorities based on specific fields in the IP and IPX headers. You can set QoS policies for IP flows based on source IP address, destination IP address, source TCP/UDP port, destination TCP/UDP port, type of service (TOS) and transport protocol (TCP or UCP). You can set QoS policies for IPX flows based on source network, source node, destination network, destination node, source port and destination port. A QoS policy set on an IP or IPX flow allows you to classify the priority of traffic based on:

- Layer-3 source-destination flows
- Layer-4 source-destination flows
- Layer-4 application flows

Configuring IP QoS Policies

To configure an IP QoS policy, perform the following tasks:

1. Identify the Layer-3 or Layer-4 flow and set the IP QoS policy.
2. Specify the precedence for the fields within an IP flow.

Setting an IP QoS Policy

To set a QoS policy on an IP traffic flow, enter the following command in Configure mode:

Set an IP QoS policy.	**qos set ip** <*name*> <*priority*> <*srcaddr/mask*>|**any** <*dstaddr/mask*>|**any** <*srcport*>|**any** <*dstport*>|**any** <*tos*>|**any** <*port list*>|<*interface-list*>|**any** <*protocol*>|**any** <*tos-mask*>|**any** <*tos-precedence-rewrite*>|**any** <*tos-rewrite*>|**any**

For example, the following command assigns control priority to any traffic coming from the 10.10.11.0 network:

```
xp(config)# qos set ip xyz control 10.10.11.0/24
```

Specifying Precedence for an IP QoS Policy

To specify the precedence for an IP QoS policy, enter the following command in Configure mode:

Specify precedence for an IP QoS policy.	**qos precedence ip** [**sip** <*num*>] [**dip** <*num*>] [**srcport** <*num*>] [**destport** <*num*>] [**tos** <*num*>] [**protocol** <*num*>] [**intf** <*num*>]

Configuring IPX QoS Policies

To configure an IPX QoS policy, perform the following tasks:

1. Identify the Layer-3 or Layer-4 flow, and set the IPX QoS policy.
2. Specify the precedence for the fields within an IPX flow.

Setting an IPX QoS Policy

To set a QoS policy on an IPX traffic flow, enter the following command in Configure mode:

Set an IPX QoS policy.	**qos set ipx** <*name*> <*priority*> <*srcnet*>|**any** <*srcmask*>|**any** <*srcport*>|**any** <*dstnet*>|**any** <*dstmask*>|**any** <*dstport*>|**any** <*port list*>|<*interface-list*>|**any**

Specifying Precedence for an IPX QoS Policy

To specify the precedence for an IPX QoS policy, enter the following command in Configure mode:

Specify precedence for an IPX QoS policy.	**qos precedence ipx** [**srcnet** <*num*>] [**srcnode** <*num*>] [**srcport** <*num*>] [**dstnet** <*num*>] [**dstnode** <*num*>] [**dstport** <*num*>] [**intf** <*num*>]

Configuring the X-Pedition Queuing Policy

The X-Pedition queuing policy is set on a system-wide basis. The X-Pedition default queuing policy is strict priority. To change the queuing policy to weighted-fair queuing on the X-Pedition router, enter the following command in Configure mode:

Set queuing policy to weighted-fair.	**qos set queuing-policy weighted-fair port** <*port list*>|**all-ports input** <*slot num*> |**all-modules**

If you want to revert the X-Pedition queuing policy from weighted-fair to strict priority (default), enter the following command in Configure mode:

Revert the X-Pedition queuing policy to strict priority.	**negate** <*line within active-configuration containing qos set queuing-policy weighted-fair*>

Allocating Bandwidth for a Weighted-Fair Queuing Policy

If you enable the weighted-fair queuing policy on the X-Pedition router, you can allocate bandwidth for the queues on the X-Pedition router. To allocate bandwidth for each X-Pedition queue, enter the following command in Configure mode:

Allocate bandwidth for a weighted-fair queuing policy.	**qos set weighted-fair control** <*percentage*> **high** <*percentage*> **medium** <*percentage*> **low** <*percentage*> **port** <*port list*>|**all-ports** **input** <*slot num*> |**all-modules**

Weighted Random Early Detection (WRED)

WRED is a dynamic process used to control congestion on WRED-enabled ports and the segments of the network associated with them by selectively dropping packets before the queue becomes completely flooded. The WRED process consists of setting a *minimum queue threshold* (min-threshold) and a *maximum queue threshold* (max-threshold) on any of the four queues (low, medium, high, and control) that belong to a port. Associated with these thresholds is an *average queue size* which is dynamically calculated as the instantaneous average buffer depth. If the average queue size is below the min-threshold, no packets are discarded; however, if the average queue size rises above the min-threshold, WRED uses a *packet-marking probability* algorithm to randomly mark packets for discard.

As the average queue size approaches the max-threshold, the probability of packet drop increases—if the average queue size exceeds the max-threshold, the probability of packet drop becomes 1 (100% of packets are dropped). Generally speaking, the increase in the probability of packet drop depends on bandwidth—the more packets sent by a particular connection, the greater the probability that the connection will drop packets.

WRED’s Effect on the Network

WPRED’s full congestion-reducing capabilities are best used with TCP and other connection-oriented protocols. As TCP traffic on a WRED port increases and the average queue size rises above the min-threshold, some TCP packets begin to drop. Each TCP source interprets dropped packets as an indication of congestion. As a result, each TCP source to experience a dropped packet reduces its window, the average queue size decreases, and congestion is alleviated.

Although connection-less protocols do not have TCP’s ability to detect congestion, WPRED’s technique of dropping packets based on rising probability assures that connections sending the most packets or using the greatest amount of bandwidth are more likely to drop packets than connections using lower bandwidths. This provides greater throughput equality on a WRED port for connection-less protocols.

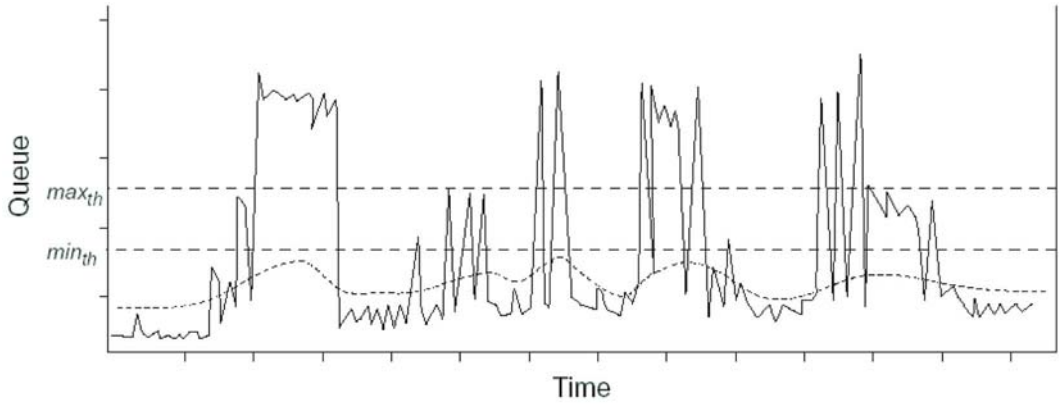
Weighting Algorithms in WRED

WRED provides the ability to fine-tune the *average queue size* and *packet-marking probability* algorithms. Control over the average queue size algorithm is provided via the **exponential-weighting-constant** parameter, while control over the packet-marking probability algorithm is provided by the **mark-prob-denominator** parameter.

The **exponential-weighting-constant** parameter specifies how quickly the average queue size adjusts to reflect changes in the actual queue depth—altering this parameter allows you to dampen response time. With WRED applied to *input* queues, the **exponential-weighting-constant** accepts a value of zero (0) to three (3). If applied to *output* queues, WRED allows you to assign a value of zero (0) to seven (7). Please note that a value of 0 provides the least amount of dampening.

The ability to dampen the response time for changes to the average queue size changes the way that WRED responds to bursty traffic. For example, notice in Figure 43 that while the traffic (solid line) bursts at times, the average queue size (dotted curve) is dampened such that it does not rise above the minimum threshold within the duration of the bursts. This prevents the port from discriminating against periodic bursts of traffic.

Figure 43. Average Queue Size and Bursty Traffic



The **mark-prob-denominator** parameter is used to determine the probability of a packet being dropped when the average queue size is between the minimum and maximum thresholds. With WRED applied to *input* queues, the **mark-prob-denominator** accepts a value of zero (0) to three (3). If applied to *output* queues, WRED allows you to assign a value of zero (0) to seven (7). Please note that the lower the value, the higher the probability of dropping packets.

Both the **exponential-weighting-constant** value and the **mark-prob-denominator** value are somewhat allegorical in the sense that neither value has a direct numerical significance other than acting as control values for WRED. For example, if you increase the value for **exponential-weighting-constant** from 1 to 2, the dampening of the average queue size response is not twice as slow. Because of the non-specific nature of **exponential-weighting-constant** and **mark-prob-denominator** and the fact that each network is different, a discussion of recommended, specific values for these parameters is beyond the scope of this text.

When first implementing WRED on an X-Pedition router, Enterasys Networks recommends that you initially use the default values for min-threshold, max-threshold, the weighting constant, and the probability denominator. If you begin to experience congestion or if congestion continues (especially with TCP traffic), try adjusting WRED by making small changes to the various parameters (one at a time) and observing the effect on congestion.

To enable WRED on input queues of specific ports, enter the following command in Configure mode:

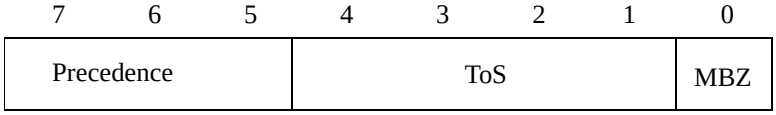
Enable WRED on input or output queue of specified ports.	**qos wred input** [**port** <*port list*>|**all-ports**] [**queue control**|**high**|**medium**|**low**] [**exponential-weighting-constant** <*num*>] [**min-queue-threshold** <*num*>] [**max-queue-threshold** <*num*>] [**mark-prob-denominator** *num*>]

ToS Rewrite

In the Internet, IP packets that use different paths are subject to delays, as there is little inherent knowledge of how to optimize the paths for different packets from different applications or users. The IP protocol actually provides a facility, which has been part of the IP specification since the protocol's inception, for an application or upper-layer protocol to specify how a packet should be handled. This facility is called the Type of Service (ToS) octet.

The ToS octet part of the IP specification, however, has not been widely employed in the past. The IETF is looking into using the ToS octet to help resolve IP quality problems. Some newer routing protocols, like OSPF and IS-IS, are designed to be able to examine the ToS octet and calculate routes based on the type of service.

The ToS octet in the IP datagram header consists of three fields:



- The three-bit Precedence field is used to indicate the priority of the datagram.
- The four-bit ToS field is used to indicate trade-offs between throughput, delay, reliability, and cost.
- The one-bit "must be zero" (MBZ) field is not currently used. (In the X-Pedition configuration, there is no restriction on this bit and it is included as part of the ToS field.)

For example, setting the ToS field to 0010 specifies that a packet will be routed on the most reliable paths. Setting the ToS field to 1000 specifies that a packet will be routed on the paths with the least delay. (Refer to RFC 1349 for the specification of the ToS field value.)

With the ToS rewrite command, you can access the value in the ToS octet (which includes both the Precedence and ToS fields) in each packet. The upper-layer application can then decide how to handle the packet, based on either the Precedence or the ToS field or both fields. For example, you can configure a router to forward packets using different paths, based on the ToS octet. You can also change the path for specific applications and users by changing the Precedence and/or ToS fields.

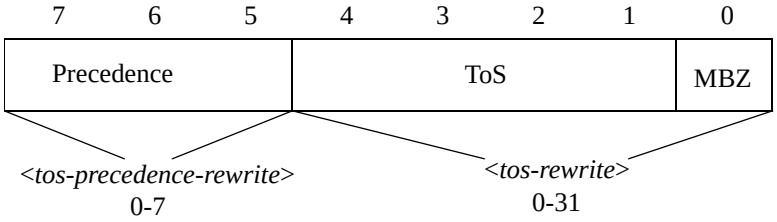
Note: In RFC 2474, the IETF redefined the ToS octet as the "DiffServ" byte. You will still be able to use the ToS rewrite feature to implement DiffServ when this standard is deployed.

Configuring ToS Rewrite for IP Packets

The ToS rewrite for IP packets is set with the **qos set** command in Configure mode. You can define the QoS policy based on any of the following IP fields: source IP address, destination IP address, source port, destination port, ToS, port, or interface.

When an IP packet is received, the ToS field of the packet is ANDed with the <*tos-mask*> and the resulting value is compared with the ANDed value of <*tos*> and <*tos-mask*> of the QoS policy. If the values are equal, the values of the <*tos-rewrite*> and <*tos-precedence-rewrite*> parameters will be written into the packet.

The <*tos*> and <*tos-mask*> parameters use values ranging from 0 to 255. They are used in conjunction with each other to define which bit in the <*tos*> field of the packet is significant. The <*tos-precedence-rewrite*> value ranges from 0 to 7 and is the value that is rewritten in the ToS Precedence field (the first three bits of the ToS octet). The <*tos-rewrite*> value ranges from 0 to 31 and is the value that is rewritten in the ToS field (the last five bits of the ToS octet, which includes both the ToS field and the MBZ bit).



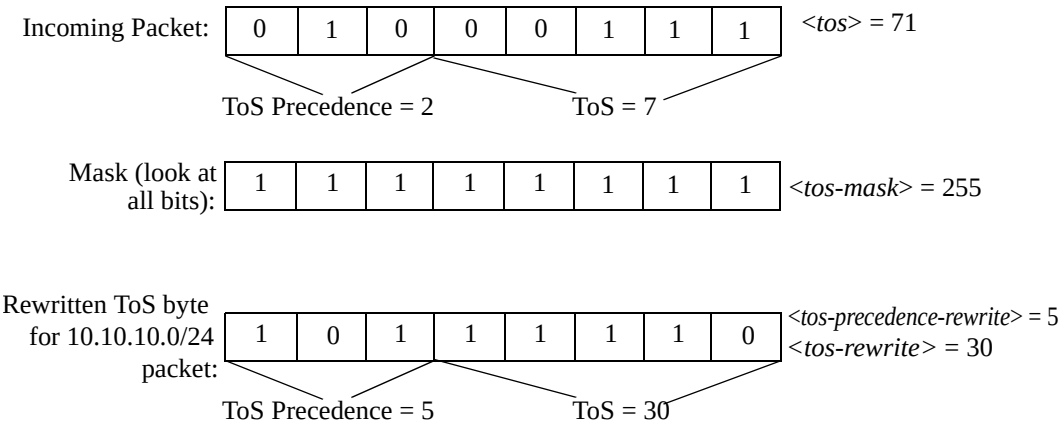
The ToS byte rewrite is part of the QoS priority classifier group. The entire ToS byte can be rewritten or only the precedence part of the ToS byte can be rewritten. If you specify a value for <*tos-precedence-rewrite*>, then only the upper three bits of the ToS byte are changed. If you set <*tos-precedence-rewrite*> to **any** and specify a value for <*tos-rewrite*>, then the upper three bits remain unchanged and the lower five bits are rewritten. If you specify values for both <*tos-precedence-rewrite*> and <*tos-rewrite*>, then the upper three bits are rewritten to the <*tos-precedence-rewrite*> value and the lower five bits are rewritten to the <*tos-rewrite*> value.

For example, the following command will rewrite the ToS Precedence field to 7 if the ToS Precedence field of the incoming packet is 6:

```
xp(config)# qos set ip tosp6to7 low any any any any 222 any any 224 7
```

In the previous example, the <*tos*> value of 222 (binary value 1101 1110) and the <*tos-mask*> value of 224 (binary value 1110 0000) are ANDed together to specify the ToS Precedence field value of 6 (binary value 110). Changing the value in the <*tos-mask*> parameter determines the bit in the ToS octet field that will be examined.

The following example will rewrite the ToS Precedence and the ToS fields to 5 and 30 if the incoming packet is from the 10.10.10.0/24 network with the ToS Precedence field set to 2 and the ToS field set to 7. (In this example, the MBZ bit is included in the ToS field.) The figure below shows how the parameter values are derived.



The <*tos-mask*> value determines the ToS bit to be examined, which is all eight bits in this example. The following command configures the ToS rewrite for the example:

```
xp(config)# qos set ip tos30to7 low 10.10.10.0/24 any any any 71 any any 255 5 30
```

Monitoring QoS

The X-Pedition router provides display of QoS statistics and configurations contained in the X-Pedition router.

To display QoS information, enter the following commands in Enable mode:

Show all IP QoS flows.	**qos show ip**
Show all IPX QoS flows.	**qos show ipx**
Show all Layer-2 QoS flows.	**qos show l2 all-destination all-flow ports** <*port-list*> **vlan** <*vlanID*> **source-mac** <*MACaddr*> **dest-mac** <*MACaddr*>
Show RED parameters for each port.	**qos show red** [**input port** <*port-list*>|**all-ports**] [**output port** <*port-list*>|**all-ports**] [**port** <*port-list*>|**all-ports**]
Show IP or IPX precedence values.	**qos show precedence ip**|**ipx**
Show WFQ bandwidth allocated for each port.	**qos show wfq** [**port** <*port-list*> |**all-ports**] [**input** <*slot num*> | **all-modules**]
Show priority mappings.	**qos show priority-map all**

Limiting Traffic Rate

Note: Some commands in this facility require updated X-Pedition hardware. For a complete list of hardware and the features they support, consult the Release Notes on the Enterasys Networks web site: **www.enterasys.com**

Rate limiting provides the ability to control the usage of a fundamental network resource, bandwidth. It allows you to limit the rate of traffic (in bits per second) that flows through the specified interfaces, thus reserving bandwidth for critical applications. The X-Pedition router supports the following types of rate limiting:

- **Per-flow Rate Limiting**—Configure policies that limit individual flows to a specified rate. This policy requires that the line card be in per-flow rate limiting mode. See *Rate Limiting Modes* on page 498.
- **Flow-Aggregate Rate Limiting**—Configure policies that limit an aggregation of flows (all flows that match an ACL) to a specified rate. For example, limit traffic to or from a particular subnet. This type of rate limiting is performed mostly in software; however, packet forwarding is performed in the hardware. This policy requires that the line card be in per-flow rate limiting mode (see *Rate Limiting Modes* on page 498). Flow-aggregate rate limiting is designed for use with line cards that do not support aggregate rate limiting.

Note: Flow-Aggregate rate limiting (software-based) was implemented as a work-around where aggregate rate limiting is not supported in the hardware. The difference between *flow-aggregate* and *aggregate* rate limiting is that the allowable bandwidth is distributed among the flows in the *software*—not the *hardware*. Flow-Aggregate rate limiting allows you to rate limit the aggregation of flows when the hardware does not support aggregate rate limiting. Flow-aggregate rate limiting is not supported on 802.1q trunk ports.

- **VLAN Rate Limiting**—Configure policies that limit traffic coming into or leaving a particular VLAN. This type of policy can be used to limit any type of traffic. Specifying the aggregate option will aggregate all flows matching this policy and distribute the specified rate among these flows. If you do not specify this option, each matching flow will be limited to the full rate. This policy requires that the line card be in per-flow rate limiting mode (see *Rate Limiting Modes* on page 498).

- **Aggregate Rate Limiting**—Configure policies that limit an aggregation of flows (all flows that match an ACL) to a specified rate. For example, you can limit traffic to or from a particular subnet. Aggregate rate limiting requires that the line card be in aggregate rate limiting mode (see *Rate Limiting Modes* on page 498). This type of policy can be used to limit any type of traffic.

  Note: Aggregate rate limiting is supported on certain line cards only. If the line card does not support aggregate rate limiting, use software-based flow-aggregate rate limiting. Aggregate rate limiting is not supported on 802.1q trunk ports.

- **Port-level Input Rate Limiting**—Configure policies that limit traffic coming into a particular port. This type of policy can be used to limit any type of traffic and requires that the line card be in aggregate rate limiting mode. See *Rate Limiting Modes* on page 498.

- **Port-level Output Rate Limiting**—Configure policies that limit traffic going out of a particular port. This type of policy can be used to limit any type of traffic and will work for line cards in per-flow or aggregate mode. See *Rate Limiting Modes* on page 498.

Note: You can configure a maximum of 24 port and aggregate rate-limiting policies per line card. Credit buckets are reserved in hardware for both port-level and aggregate policies. You are limited to 8 aggregate policies per line card unless you make the credit buckets reserved for port-level policies available for aggregate policies. To do this, enter the **system disable inputportlevel-rate-limiting slot** command from Configure mode.

Traffic Profiles

A *traffic profile* is used to define the traffic characteristics before an upper limit is assigned—these are not used for VLAN and port-level policies. The traffic profile is created using an ACL, which can utilize any combination of the parameters supported in IP ACL. A rate limiting policy can then be defined by using the ACL and traffic rate limitations. You define the action to be taken on the traffic that exceeds the upper limit. For example, dropping packets. Except for VLAN and port-level rate limiting, the rate limiting policy is then applied to a logical IP interface.

Rate limiting policies work in one direction only. That is, only the traffic coming into the interface to which a policy is applied will be subject to rate limiting (except for output port rate limiting policies, which are applied to egress ports). If you need to rate limit incoming and outgoing traffic to a network or subnet, create and apply separate policies to each interface.

Burst-Compensating

When you choose the burst-compensating option, the X-Pedition router invokes a different algorithm for calculating the rate limit values used by the hardware. This algorithm is better at compensating for *burst capacity*—the ability to maintain an average close to the specified rate—even with large bursts of traffic. The burst-compensating option is available on all rate-limiting policies except port-level output and requires that you enter a burst-compensator value of 1-100 to represent how much burst capacity (in Mbps) to build into the rate limit. For example, setting a low burst-compensator value on a rate limit policy to restrict the flow of an FTP server and client that are capable of very high transfer rates will choke off the flow and produce realized rates that are smaller than the specified rate limit. Conversely, setting a higher burst-compensator value than the flow's unrestricted capabilities will result in realized rates that are higher than the specified rate limit.

Note: Due to hardware constraints, the realized rates for rate limits set above 20 Mbps will become increasingly less consistent and accurate.

The X-Pedition router uses a *credit bucket* bandwidth policing scheme to perform rate limiting. This policy creates a *credit bucket* and *time slice* in the Input Packet Processor hardware and, for a given rate limit, calculates a credit bucket size to represent the amount of traffic that can pass through the processor within a specific time period (i.e., the time slice value). When you specify the burst-compensating option, the credit bucket and time slice values are calculated to take into account traffic spikes or *bursts* and to achieve an average rate that is as close as possible to the specified rate. When the bucket is *filled* within the specified time slice, the X-Pedition router will drop packets or change the priority of the packets, depending on the exceed-action specified. Because the burst compensating option allows you to create larger credit buckets and smaller time slices, you can prevent constricting the flow of rate-limited, bursty traffic.

Note: If you do not specify this option, rate-limiting will provide accurate results (to within 10-15%) for "smooth" traffic only (e.g., traffic created by a traffic generator). For example, if you use a small credit bucket and a large time slice, a burst of traffic can fill the bucket and cause the X-Pedition router to drop traffic (until the time slice expires and refreshes the credit bucket). This can choke off the traffic rate. For bursty traffic such as TCP and most traffic running on a live network, use the burst-compensating option.

Rate Limiting Modes

There are two rate limiting modes available on the X-Pedition router: *Per-flow* and *Aggregate*. The X-Pedition router requires that you enable per-flow rate limiting mode (the default) on any line card that has Per-flow, Flow-aggregate, or Vlan policies applied to its interfaces or ports. You must enable aggregate mode on any line card that has Aggregate or Port-level Input policies applied to its interfaces or ports. Port-level Output policies can be applied to line cards in either mode. To enable aggregate rate limiting mode on a line card, enter the following command in Configure mode:

```
system enable aggregate-rate-limiting slot
```

To change the rate limiting mode, no conflicting rate limiting policies may exist. For example, before you can enable aggregate rate limiting mode on a line card, you need to delete any existing per-flow, flow-aggregate, or VLAN rate limiting policies on that line card.

Note: To change the rate limiting mode on the line card back to per-flow, negate the above command.

Rate Limiting Policies

Per-Flow Rate Limiting

Use a per-flow rate limiting policy if an individual traffic flow needs to be limited to a particular rate. A single per-flow rate limiting policy can have multiple ACLs to define different traffic profiles and traffic rate limitations. When there are multiple traffic profiles, a sequence number is used to identify the order in which the profiles are applied. Per-flow rate limiting policies require that the line card be in Per-flow rate limiting mode.

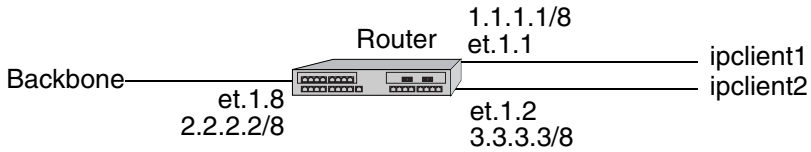
To define a per-flow rate limit policy and apply the policy to an interface, enter the following commands in Configure mode:

Define a per-flow rate limit policy.	**rate-limit** <*name*> **input acl** <*acl list*> **rate** <*number*> **exceed-action drop-packets| set-priority-low| set-priority-medium| set-priority-high [sequence** <*number*>**]| [burst-compensating]**
Apply a per-flow rate limit profile to an interface.	**rate-limit** <*name*> **apply interface** <*interface*> | **all**

Note: You cannot use non-IP ACLs for per-flow rate limit policies.

Example of Per-Flow Rate Limiting

The following is an example of configuring per-flow rate limiting on the X-Pedition router.



Traffic from two interfaces, ipclient1 with IP address 1.2.2.2 and ipclient2 with IP address 3.1.1.1, is restricted to 10 Mbps for each flow with the following configuration:

```
vlan create client1 ip
vlan create backbone ip
vlan create client2 ip
vlan add ports et.1.1 to client1
vlan add ports et.1.2 to client2
vlan add ports et.1.8 to backbone
interface create ip ipclient1 vlan client1 address-netmask 1.1.1.1/8
interface create ip ipclient2 vlan client2 address-netmask 3.3.3.3/8
interface create ip backbone vlan backbone address-netmask 2.2.2.2/8
acl 100 permit ip 1.2.2.2
acl 200 permit ip 3.1.1.1
rate-limit client1 input acl 100 rate 10000000 exceed-action drop-packets
rate-limit client2 input acl 200 rate 10000000 exceed-action drop-packets
rate-limit client1 apply interface ipclient1
rate-limit client2 apply interface ipclient2
```

Aggregate Rate Limiting

Use an aggregate rate limiting policy if an aggregation of flows needs to be limited to a particular rate. For example, you can use aggregate rate limiting to rate limit traffic to or from a particular subnet.

Note: You cannot apply an aggregate rate limiting policy to an interface that spans ports on more than one line card. For example, you cannot apply an aggregate rate limiting policy to the interface ip2, if it interfaces to a VLAN that consists of ports et.1.(1-4) and et.2.(1-4).

Note: Aggregate and flow-aggregate rate limiting are not supported on 802.1q trunk ports.

To configure aggregate rate limiting policies, you must first enable aggregate rate limiting mode on the line card (see *Rate Limiting Modes* on page 498). To define an aggregate rate limit policy and apply the policy to an interface, use the following commands in the Configure mode:

Define an aggregate rate limit policy.	**rate-limit** <*name*> **aggregate acl** <*acl list*> **rate** <*num*> [**drop-packets**| **no-action**| **lower-priority** | **lower-priority-except-control** | **tos-precedence-rewrite** <*num*> | **tos-precedence-rewrite-lower-priority** <*num*>] [**allocate-resources-during-apply** | **allocate-resources-during-traffic**] | [**burst-compensating**]
Apply an aggregate rate limit policy to an interface.	**rate-limit** <*name*> **apply interface** <*interface*> | **all**

Note: You cannot use non-IP ACLs for aggregate rate-limit policies. The X-Pedition router displays interface names up to 32 characters in length.

Example of Aggregate Rate Limiting

In the following example, incoming FTP and HTTP traffic to the subnetwork 122.132.0.0/16 will be rate limited to 4 Mbps and 2 Mbps, respectively:

```
system enable aggregate-rate-limiting slot 1
interface create ip engintf address-netmask 122.132.10.23/16 port et.1.6
acl engftp permit ip 122.132.0.0/16 any any 20
rate-limit engftp aggregate acl engftp rate 4000000 drop-packets
acl enghttp permit ip 122.132.0.0/16 any any 80
rate-limit enghttp aggregate acl enghttp rate 2000000 drop-packets
rate-limit engftp apply interface engintf
rate-limit enghttp apply interface engintf
```

In the above example, the first configuration command is needed to enable the aggregate rate limiting mode on the line card in slot 1 (per-flow is the default rate limiting mode).

Flow-Aggregate Rate Limiting

The Flow-Aggregate Rate Limiting policy allows you to limit an aggregation of flows to a particular rate. For example, you can use aggregate rate limiting to rate limit traffic to or from a particular subnet; however, you do not need to enable the aggregate rate limiting mode on the line card to use flow-aggregate rate limiting. See *Rate Limiting Modes* on page 498 for more information. Flow-aggregate rate limiting was created as a work-around to be used on line cards that do not support Aggregate rate limiting.

Note: You cannot use non-IP ACLs for flow-aggregate rate limit policies.

Note: Aggregate and flow-aggregate rate limiting are not supported on 802.1q trunk ports.

To define a flow-aggregate rate limit policy use the following commands in the Configure mode:

Define a software-based flow-aggregate rate limit policy.	**rate-limit** <*name*> **flow-aggregate acl** <*acl list*> **rate** <*rate*> **exceed-action** <*action*> [**sequence** <*number*>] | [**burst-compensating**] | **min-bandwidth** <*min-bw*> | **distribute-among** <*number-of-flows*>]
Apply a flow-aggregate rate limit policy to an interface.	**rate-limit** <*name*> **apply interface** <*interface*> | **all**

Example of Flow-Aggregate Rate Limiting

In the following example, traffic from the subnetwork 122.132.0.0/16 to the Internet will be rate limited to 256 Kbps while traffic to the subnetwork will be rate limited to 64 Kbps.

```
acl cust1 permit ip any 122.132.0.0/16
acl cust1 permit ip 122.132.0.0/16 any
rate-limit cust1 flow-aggregate acl cust1 rate 256000 exceed-action drop-packets min-bandwidth 4000
rate-limit apply cust1 interface tonet
rate-limit cust1 flow-aggregate acl cust1 rate 64000 exceed-action drop-packets min-bandwidth 2000
rate-limit apply cust1 interface in1
```

Note: The X-Pedition router displays interface names up to 32 characters in length.

When using Flow-Aggregate Rate-Limiting, the amount of traffic exiting a serial WAN port running PPP may be significantly less than the specified rate-limit. This is due to a PPP priority queue overflow. To work around this problem, use the following commands to change the PPP priority queue depths on the serial port:

ppp define service <*name*> [**low-priority-queue-depth** | **med-priority-queue-depth** | **high priority-queue-depth**] <*number*>

ppp apply service <*name*> **ports** <*wan-port*>

Port Rate Limiting

Use a port rate limiting policy if incoming or outgoing traffic on a particular port needs to be rate limited. Unlike other types of rate limiting policies, you do not specify an ACL when defining this type of policy. Port rate limiting policies do not need to be applied to an interface and take effect when they are created.

To configure port rate limiting policies for input ports, you must first enable the aggregate rate limiting mode on the line card (see *Rate Limiting Modes* on page 498). You do not need to enable the aggregate rate limiting mode to configure a policy to limit outgoing traffic on a port. You can configure port-level rate limiting policies on output ports in either per-flow or aggregate rate limiting mode.

Note: Aggregate and flow-aggregate rate limiting are not supported on 802.1q trunk ports.

To define a port rate limit policy, use the following commands in the Configure mode:

Define a port rate limit policy to limit incoming traffic on a port.	**rate-limit** <*name*> **port-level input port** <*port list*> **rate** <*num*> [**drop-packets** | **no-action** | **lower-priority** | **lower-priority-except-control** | **tos-precedence-rewrite** <*num*> | **tos-precedence-rewrite-lower-priority** <*num*>]
Define a port rate limit policy to limit outgoing traffic on a port.	**rate-limit** <*name*> **port-level output port** <*port list*> **rate** <*num*> **drop-packets**

Note that for output port policies, the only action that you can specify if traffic exceeds the specified rate is to drop packets.If you configure output port policies, all types of outgoing IP traffic will be rate limited, including control traffic. If you do not want control traffic to be subject to rate limiting, enter the following command in the Configure mode:

Specify that control traffic is not subject to output port rate limiting.	**rate-limit** <*name*> **port-level slot** <*num*> **ignore-control-priority**

Because you specify a slot number in the above command, output port rate limiting policies on any port on the specified slot will not be applied to control traffic.

VLAN Rate Limiting

Use a VLAN rate limiting policy if incoming or outgoing traffic on a particular VLAN needs to be rate limited. Like Port rate limiting policies, you do not specify an ACL when defining this type of policy. VLAN rate limiting policies do not need to be applied to an interface and take effect when they are created. Specifying the aggregate option will aggregate all flows matching this policy and distribute the specified rate among these flows. If you do not specify this option, each matching flow will be limited to the full rate. To configure VLAN rate limiting policies you must be in per-flow mode (see *Rate Limiting Modes* on page 498) and you must have L4-bridging enabled on the VLAN (see *Layer-4 Bridging and Filtering* on page 479).

To define a VLAN rate limiting policy, enter one of the following commands in Configure mode:

Define a VLAN rate limiting policy to limit incoming or outgoing traffic on a VLAN.	**rate-limit** <*name*> **vlan** <*name*> **port** <*port list*> | **all-ports destport** <*port list*> | **all-ports rate** <*num*> **exceed-action drop-packets** | **set-priority-low** | **set-priority-medium** | **set-priority-high** [**burst-compensating**] **aggregate**

Displaying Rate Limit Information

To show information about rate limit policies, use the following command in the Enable mode:

Show rate limit policy information.	**rate-limit show** [**all**] | [**policy-type flow-policies** | **flow-aggregate-policies** | **aggregate-policies** | **portlevel-policies** | **all**] | [**policy-name** <*name*>] | [**interface** <*interface*>] | [**port-level port** <*port list*> | **all-port**] | [**port-level policy-name** <*name*>] | [**rate-limiting-mode**]

Note: The X-Pedition router displays interface names up to 32 characters in length.

Chapter 28

Performance Monitoring Guide

Performance Monitoring Overview

The X-Pedition router is a full wire-speed Layer-2, -3 and -4 switching router. As packets enter the X-Pedition router, Layer-2, -3, and -4 flow tables are populated on each line card. The flow tables contain information on performance statistics and traffic forwarding. Thus the X-Pedition router provides the capability to monitor performance at Layer-2, -3, and -4. Layer-2 performance information is accessible to SNMP through MIB-II and can be displayed by using the **l2-tables** command in the CLI. Layer-3 and Layer-4 performance statistics are accessible to SNMP through RMON/RMON2 and can be displayed by using the **statistics show** command in the CLI. In addition to the monitoring commands listed, you can find more monitoring commands listed in each chapter of the *Enterasys X-Pedition Command Line Interface Reference Manual*.

To access statistics on the X-Pedition router, enter any of the following commands in Enable mode:

Show DVMRP routes.	**dvmrp show route**
Show all TCP/UDP connections and services.	**ip show connections**
Show all IPv4 routes.	**ip show routes**
Show all IPv6 routes.	**ipv6 show routes**
Show all IPX routes.	**ipx show tables routing**
Show all MAC addresses currently in the L2 tables.	**l2-tables show all-macs**
Show info about MACs residing in a port's L2 table.	**l2-tables show port-macs** <*port-list*>

Show all L2 flows (for ports in flow-bridging mode.	**l2-tables show all-flows**
Show information about the master MAC table.	**l2-tables show mac-table-stats**
Show information about a particular MAC address.	**l2-tables show mac**
Show info about multicasts registered by IGMP.	**l2-tables show igmp-mcast-registrations**
Show whether IGMP is on or off on a VLAN.	**l2-tables show vlan-igmp-status**
Show info about MACs registered by the system.	**l2-tables show bridge-management**
Show SNMP statistics.	**snmp show statistics**
Show ICMP statistics.	**statistics show icmp**
Show IPv4 interfaces statistics.	**statistics show ip**
Show IPv6 interfaces statistics.	**statistics show ipv6-interface**
Show unicast routing statistics.	**statistics show ip-routing**
Show IPX statistics.	**statistics show ipx**
Show IPX interface's statistics.	**statistics show ipx-interface**
Show IPX routing statistics.	**statistics show ipx-routing**
Show multicast statistics.	**statistics show multicast**
Show port error statistics.	**statistics show port-errors**
Show port normal statistics.	**statistics show port-stats**
Show RMON statistics.	**statistics show rmon**
Show traffic summary statistics.	**statistics show summary-stats**
Show most active tasks.	**statistics show most-active**
Show TCP statistics.	**statistics show tcp**
Show UDP statistics.	**statistics show udp**
Show broadcast monitoring information for ports.	**port show bmon** [**config**][**detail**][**port** <*port list*>][**stats**]
Show all VLANs.	**vlan list**

Configuring for Port Mirroring

The X-Pedition router allows you to monitor activity with port mirroring. Port mirroring allows you to monitor the performance and activities of ports on the X-Pedition router or for traffic defined by an ACL through one or more separate ports. While in Configure mode, you can configure your X-Pedition router for port mirroring with a simple command line like the following:

Configure Port Mirroring.	**port mirroring dst-ports** <*port_list*> [**src-ports** <*port_list*> | **src-acl** <*acl_name*>]

Note: Port mirroring is available for WAN ports. However, port mirroring, ACL, and Layer-2 filtering are supported on a per-WAN-card basis, not a per-port basis. (You can only configure port mirroring for the entire WAN card).

Only IP ACLs can be specified for port mirroring.

The X-Pedition router does not support port mirroring on ATM ports.

For additional information, see *Port Mirroring* on page 477.

Chapter 29

RMON Configuration Guide

RMON Overview

You can employ Remote Network Monitoring (RMON) to help monitor traffic at remote points on the network. With RMON, data collection and processing is done with a remote *probe*, namely the X-Pedition router. The X-Pedition router also includes RMON *agent* software that communicates with a network management station via SNMP. Because information transmits from the X-Pedition router to the management station only when required, SNMP traffic on the network and the management station's processing load are reduced.

The X-Pedition router provides support for both RMON 1 and RMON 2 MIBs, as specified in RFCs 1757 and 2021, respectively. While non-RMON SNMP products allow the monitoring and control of specific network *devices*, RMON 1 returns statistics on network *segments* at the MAC layer. RMON 2 collects statistics on network and application layer *traffic* to show host-to-host connections and the applications and protocols being used. For example, the RMON 2 network layer matrix MIB group can show protocol-specific traffic between pairs of systems which can help to diagnose protocol problems. Note that RMON 2 is not a superset of RMON 1; on the X-Pedition router, you can configure both RMON 1 and RMON 2 statistics collection.

Note: RMON1 (RFC1757) statistic accounts for bridged traffic in host and matrix statistics, but **not** routed traffic. RMON2 (RFC2021) accounts for routed traffic in host, matrix, and protocol distribution statistics, but **not** bridged traffic. This is a hardware limitation.

Memory Allocation Requirements

The memory used by RMON is determined automatically by the number of ports and the level of functionality (i.e., lite, standard, or professional) you select. When you use the **rmon enable** command, the RMON memory allocation requirements for your system are calculated using the following values

Table 14. RMON Memory Allocation Requirements

500k	**base initialization**
60k	**lite only (per port)**
120k	**standard only (per port)**
140k	**professional only (per port)**
140k	**lite and standard (per port)**
160k	**lite and professional (per port)**
220k	**standard and professional (per port)**
240k	**lite, standard, and professional (per port)**

The table below lists the maximum amount of additional memory you may manually allocate with the "**rmon set memory**" command:

X-Pedition 2000	**X-Pedition 8000**	**X-Pedition 8600**	**ER16**
12M	32M	96M	96M

To manually allocate additional memory, use the following:

```
rmon set memory <memory in Megabytes>
```

When you remove the "**rmon enable**" command, the RMON functionality is disabled and all of the memory set aside for its use is returned to the system.

Examples

The examples below demonstrate RMON memory allocation:

32-port X-Pedition 2000 with lite and standard enabled:
500k + (32 * 140k) = 4.98M of memory

120-port X-Pedition 8600 with lite, standard, and professional enabled:
500k + (120 * 240k) = 29.3M of memory

Configuring and Enabling RMON

By default, RMON is disabled on the X-Pedition router. To configure and enable RMON on the X-Pedition router, follow these steps:

1. Turn on the Lite, Standard, or Professional RMON groups by entering the **rmon set lite|standard|professional** command. You can also configure default control tables for the Lite, Standard, or Professional RMON groups by including the **default-tables yes** parameter.
2. Enable RMON on specified ports with the **rmon set ports** command.
3. Optionally, you can configure control tables for the Lite, Standard, or Professional RMON groups. For example, if you chose *not* to create default control tables for the Lite, Standard, or Professional groups, you can configure control table entries for specific ports on the X-Pedition router.
4. Use the **rmon enable** command to enable RMON on the X-Pedition router.

Example of RMON Configuration Commands

The following are examples of the commands to configure and enable RMON on the X-Pedition router:

```
 1: port flow-bridging et.5.(3-8) *
  !
 2: interface add ip en0 address-netmask 10.50.6.9/16
  !
 3: system set contact “usama”
 4: system set location Enterasys Networks
 5: system set name “xp“
   !
 6: rmon set ports all-ports
 7: rmon set lite default-tables yes
 8: rmon set standard default-tables yes
  !
  ! Set RMON Pro Group with Default Tables ON, cap memory at 4 meg
  ! Pro: protocolDir, protocolDist, addressMap, al/nl-Matrix, al/nl-Host,
  ! al/nl-matrixTopN. userHistory, probeConfig.
  ! Default Tables: one control row per dataSource for protocolDist,
  ! addressMap, al/nl-Host, al/nl-Matrix.
  !
 9: rmon set professional default-tables yes
10: rmon set memory 4
11: rmon enable
```

* To collect layer 2 matrix information, port must be configured for flow-bridging mode. By default, ports on the X-Pedition router operate in address-bridging mode.

The next sections describe Lite, Standard, and Professional RMON groups and control tables.

RMON Groups

The RMON MIB groups are defined in RFCs 1757 (RMON 1) and 2021 (RMON 2). On the X-Pedition router, you can configure one or more levels of RMON support for a set of ports. Each level—Lite, Standard, or Professional—enables different sets of RMON groups (described later in this section). You need to configure at least one level before you can enable RMON on the router. To specify the support level for RMON groups, use the following CLI command line in Configure mode:

Specifies Lite, Standard, or Professional RMON groups.	**rmon set lite**|**standard**|**professional default-tables yes**|**no**

To specify the ports on which RMON is to be enabled, use the following CLI command line in Configure mode:

Specifies the ports on which RMON is enabled.	**rmon set ports** <*port list*>|**allports**

You can configure each level of RMON support independently of each other with default tables on or off. For example, you can configure Lite with default tables on for ports et.1.(1-8) and then configure Standard with no default tables for the same ports. You cannot configure Lite on one set of ports and Standard on another set of ports.

Lite RMON Groups

This section describes the RMON groups that are enabled when you specify the Lite support level. The Lite RMON groups are shown in the table below.

Table 15. Lite RMON Groups

Group	Function
EtherStats	Records Ethernet statistics (for example, packets dropped, packets sent, etc.) for specified ports.
Event	Controls event generation and the resulting action (writing a log entry or sending an SNMP trap to the network management station).
Alarm	Generates an event when specified alarm conditions are met.
History	Records statistical samples for specified ports.

Standard RMON Groups

This section describes the RMON groups that are enabled when you specify the Standard support level. The Standard RMON groups are shown in the table below.

Table 16. Standard RMON Groups

Group	Function
Host	Records statistics about the hosts discovered on the network.
Host Top N	Gathers the top n hosts, based on a specified rate-based statistic. This group requires the hosts group.
Matrix	Records statistics for source and destination address pairs.
Filter	Specifies the type of packets to be matched and how and where the filtered packets should flow (the channel).
Packet Capture	Specifies the capture of filtered packets for a particular channel.

Professional RMON Groups

The Professional RMON groups correspond to the RMON 2 groups defined in RFC 2021. While RMON 1 groups allow for the monitoring of packets at the MAC layer, RMON 2 groups focus on monitoring traffic at the network and application layers.

The Professional RMON groups are shown in the table below.

Table 17. Professional RMON Groups

Group	Function
Protocol Directory	Contains a list of protocols supported by the X-Pedition router and monitored by RMON. See the RMON 2 Protocol Directory appendix in the *Enterasys X-Pedition Command Line Interface Reference Manual*.
Protocol Distribution	Records the packets and octets for specified ports on a per protocol basis.
Application Layer Host	Monitors traffic at the application layer for protocols defined in the protocol directory.
Network Layer Host	Monitors traffic at the network layer for protocols defined in the Protocol Directory.
Application Layer Matrix (and Top N)	Monitors traffic at the application layer for protocols defined in the Protocol Directory. Top N gathers the top n application layer matrix entries.
Network Layer Matrix (and Top N)	Monitors traffic at the network layer for protocols defined in the Protocol Directory. Top N gathers the top n network layer matrix entries.
Address Map	Records MAC address to network address bindings discovered for specified ports.
User History	Records historical data on user-defined statistics.

Control Tables

Many RMON groups contain both control and data tables. Control tables specify what statistics are to be collected. For example, you can specify the port for which statistics are to be collected and the owner (name, phone, or IP address) for that port. You can change many of the entries in a control table with **rmon** commands. Data tables contain the collected statistics. You cannot change any of the entries in a data table; you can only view the data.

When you specify the Lite, Standard, or Professional RMON groups, you have the option of creating default control tables. A default control table creates a control table entry for every port on the X-Pedition router. Creating default control tables essentially configures data collection for every port on the router for certain RMON groups. If you do not want this, you can choose not to create the default control tables and then configure the appropriate control tables for the data you wish to collect. Even if you use the default control tables, you can always use the **rmon** commands to modify control table entries.

If you choose to create default control tables, entries are created in the control tables for each port on the X-Pedition router for the following groups:

Lite groups:	Etherstats History
Standard groups:	Host Matrix
Professional groups:	Protocol Distribution Address Map Application Layer/Network Layer Host Application Layer/Network Layer Matrix

A row in the control table is created for each port on the X-Pedition router, with the owner set to “monitor”. If you want, you can change the owner by using the appropriate **rmon** command. See the section *Configuring RMON Groups* in this chapter for more the command to configure a specific group.

Note: Control tables other than the default control tables must be configured with CLI commands, as described in *Configuring RMON Groups*.

Using RMON

RMON on the X-Pedition router allows you to analyze network traffic patterns, set up alarms to detect potential problems before they turn into real congestive situations, identify heavy network users to assess their possible candidacy for moves to dedicated or higher speed ports, and analyze traffic patterns to facilitate more long-term network planning.

RMON 1 provides layer 2 information. Traffic flowing through the X-Pedition router's layer 2 ASIC is collected by RMON 1 groups. RMON 2 in the router provides layer 3 traffic information for IP and IPX protocols. Traffic flowing through the router's layer 3 ASIC is collected by RMON 2 groups. The router's RMON 2 protocol directory contains over 500 protocols that can be decoded for UDP and TCP ports. You can use RMON to see the kinds of protocol traffic being received on a given port.

For example, use the **rmon show protocol-distribution** command to see the kinds of traffic received on a given port:

```
xp# rmon show protocol-distribution et.5.5
RMON II Protocol Distribution Table

Index: 506, Port: et.1.7, Owner: monitor
    Pkts    Octets   Protocol
    ----    ------   --------
      19      1586   ether2
      19      1586   ether2.ip-v4
      19      1586   *ether2.ip-v4
       2       192   *ether2.ip-v4.icmp
      17      1394   *ether2.ip-v4.tcp
      17      1394   *ether2.ip-v4.tcp.www-http
```

In the example output above, only HTTP and ICMP traffic is being received on this port.

To find out which host or user is using these applications/protocols on this port, use the following command:

```
xp# rmon show al-matrix et.5.5
RMON II Application Layer Host Table

Index: 500, Port: et.5.5, Inserts: 4, Deletes: 0, Owner: monitor
SrcAddr        DstAddr          Packets    Octets       Protocol
-------        -------          -------    ------       --------
10.50.89.88    15.15.15.3          1771    272562      *ether2.ip-v4
10.50.89.88    15.15.15.3          1125    211192      *ether2.ip-v4.tcp
10.50.89.88    15.15.15.3          1122    210967      *ether2.ip-v4.tcp.telnet
10.50.89.88    15.15.15.3             3       225     *ether2.ip-v4.tcp.www-http
```

Configuring RMON Groups

As mentioned previously, control tables in many RMON groups specify the data that is to be collected for the particular RMON group. If the information you want to collect is in the default control tables, then you only need to turn on the default tables when you specify the RMON groups (Lite, Standard, or Professional); you do not need to configure entries in the default tables.

The following table shows the **rmon** command that you use to configure each RMON group:

To configure the Address Map group.	**rmon address-map index** <*index-number*> {**port** <*port*> [**owner** <*string*>] [**status enable**|**disable**]} | **max-number** <*number*>
To configure the Application Layer Matrix top n entries.	**rmon al-matrix-top-n index** <*index-number*> **matrix-index** <*number*> **ratebase terminal-packets** | **terminal-octets** | **all-packets** | **all-octets duration** <*number*> **size** <*number*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the Alarm group.	**rmon alarm index** <*index-number*> **variable** <*string*> [**interval** <*seconds*>] [**falling-event-index** <*num*>] [**falling-threshold** <*num*>] [**owner** <*string*>] [**rising-event-index** <*num*>] [**rising-threshold** <*num*>] [**startup rising**|**falling**|**both**] [**status enable**|**disable**] [**type absolute-value**|**delta-value**]
To configure the Packet Capture group.	**rmon capture index** <*index-number*> **channel-index** <*number*> [**full-action lock**|**wrap**] [**slice-size** <*number*>] [**download-slice-size** <*number*>] [**download-offset** <*number*>] [**max-octets** <*number*>] [**owner** <*string*>] [**status enable**|**disable**]
To configure the Filter group, you must configure both the Channel and Filter control tables.	**rmon channel index** <*index-number*> **port** <*port*> [**accept-type matched**|**failed**] [**data-control on**|**off**] [**turn-on-event-index** <*number*>] [**turn-off-event-index** <*number*>] [**event-index** <*number*>] [**channel-status ready**|**always-ready**] [**description** <*string*>] [**owner** <*string*>] [**status enable**|**disable**] **rmon filter index** <*index-number*> **channel-index** <*number*> [**data-offset** <*number*>] [**data** <*string*>] [**data-mask** <*string*>] [**data-not-mask** <*string*>] [**pkt-status** <*number*>] [**status-mask** <*number*>] [**status-not-mask** <*number*>] [**owner** <*string*>] [**status enable**|**disable**]
To configure the Etherstats group.	**rmon etherstats index** <*index-number*> **port** <*port*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the Event group.	**rmon event index** <*index-number*> **type none**|**log**|**trap**|**both** [**community** <*string*>] [**description** <*string*>] [**owner** <*string*>] [**status enable**|**disable**]
To configure the History group.	**rmon history index** <*index-number*> **port** <*port*> [**interval** <*seconds*>] [**owner** <*string*>] [**samples** <*num*>] [**status enable**|**disable**]

To configure the Application Layer and Network Layer Host groups.	**rmon hl-host index** <*index-number*> **port** <*port*> **nl-max-entries** <*number*> **al-max-entries** <*number*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the Application Layer and Network Layer Matrix groups.	**rmon hl-matrix index** <*index-number*> **port** <*port*> **nl-max-entries** <*number*> **al-max-entries** <*number*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the Host group.	**rmon host index** <*index-number*> **port** <*port*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the Host Top N entries.	**rmon host-top-n index** <*index-number*> **host-index** <*number*> [**base** <*statistics*>] [**duration** <*time*>] [**size** <*size*>] [**owner** <*string*>] [**status enable**|**disable**]
To configure the Matrix group.	**rmon matrix index** <*index-number*> [**port** <*port*>] [**owner** <*string*>] [**status enable**|**disable**]
To configure the Network Layer Matrix top n entries.	**rmon nl-matrix-top-n index** <*index-number*> **matrix-index** <*number*> **ratebase terminal-packets** | **terminal-octets** | **all-packets** | **all-octets duration** <*number*> **size** <*number*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the Protocol Distribution group.	**rmon protocol-distribution index** <*index-number*> **port** <*port*> [**owner** <*string*>] [**status enable**|**disable**]
To configure the User History group, you must configure the group of objects to be monitored and apply the objects in the group to the User History control table.	**rmon user-history-control index** <*index-number*> **objects** <*number*> **samples** <*number*> **interval** <*number*> [**owner** <*string*>] [**status enable**|**disable**] **rmon user-history-objects** <*groupname*> **variable** <*oid*> **type absolute**|**delta** [**status enable**|**disable**] **rmon user-history-apply** <*groupname*> **to** <*user-history-index*>

Configuration Examples

This section shows examples of configuration commands that specify an event that generates an SNMP trap and the alarm condition that triggers the event.

The RMON Alarm group allows the X-Pedition router to poll itself at user-defined intervals. Alarms that constitute an event are logged into the Event table that can then be polled by the management station. The management station is able to poll more network devices this way, as it only needs to poll the RMON Event table and not the device itself. The management station can also be sent trap information.

The following examples configure the X-Pedition router to create an event when a module is hot swapped into the chassis or any new IP interface is configured. The managed object ifTableLastChanged from RFC 2233) has an object identifier (OID) of 1.3.6.1.2.1.31.1.5.0 and the router will poll this OID every 5 minutes (300 seconds).

The command line below is an example of an RMON Event group configuration with the following attributes:

- Index number 15 to identify this entry in the Event control table.
- The event is both logged in the Event table and an SNMP trap generated with the community string “public”.
- Event owner is “help desk”.

```
xp#(config) rmon event index 15 type both community public description “Interface added or
module hot swapped in” owner “help desk”
```

The command line below is an example of an RMON Alarm group configuration with the following attributes:

- Index number 20 to identify this entry in the Alarm control table.
- The OID 1.3.6.1.2.1.31.1.5.0 identifies the attribute to be monitored.
- Samples taken at 300 second (5 minute) intervals.
- A “Startup” alarm generation condition instructing the router to generate an alarm if the sample is greater than or equal to the rising threshold or less than or equal to the falling threshold.
- Compare value at time of sampling (absolute value) to the specified thresholds.
- Rising and falling threshold values are 1.
- Rising and falling event index values are 15, which trigger the previously-configured Event.

```
xp#(config) rmon alarm index 20 variable 1.3.6.1.2.1.31.1.5.0 interval 300 startup both type
absolute-value rising-threshold 1 falling-threshold 1 rising-event-index 15 falling-event-index 15
owner “help desk”
```

Displaying RMON Information

The CLI **rmon show** commands allow you to display the same RMON statistics that can be viewed from a management station. To display RMON statistics for the X-Pedition router, use the following CLI command lines in Enable mode:

[1]Show Ethernet statistics.	**rmon show etherstats** <*port-list*>|**all-ports**
Show all events and logs.	**rmon show events**
Show all alarms.	**rmon show alarms**
Show histories and logs.	**rmon show history** <*port-list*>|**all-ports**
Show hosts and logs.	**rmon show hosts** <*port-list*>|**all-ports** [**summary**]
Show all Host Top N and logs.	**rmon show host-top-n**
Show matrices and logs.	**rmon show matrix** <*port-list*>|**all-ports**
Show all channels.	**rmon show channels**
Show all filters.	**rmon show filters**
Show all packet captures and logs.	**rmon show packet-capture**
Display the RMON 2 Protocol Directory.	**rmon show protocol-directory**
Display the RMON 2 Protocol Distribution.	**rmon show protocol-distribution** <*port-list*>|**all-ports**
To display the RMON 2 Address Map table.	**rmon show address-map** <*port-list*>|**all-ports**
Show Network Layer Host logs.	**rmon show nl-host**<*port-list*>|**all-ports** [**summary**]
Show Application Layer Host logs.	**rmon show al-host** <*port-list*>|**all-ports** [**summary**]
Show Network Layer Matrix logs.	**rmon show nl-matrix** <*port-list*>|**all-ports** [**order-by srcdst**|**dstsrc**] [**summary**]
Show Application Layer Matrix logs.	**rmon show al-matrix** <*port-list*>|**all-ports** [**order-by srcdst**|**dstsrc**] [**summary**]
Show all Network Layer Matrix Top N.	**rmon show nl-matrix-top-n**
Show all Application Layer Matrix Top N.	**rmon show al-matrix-top-n**
Show all user history logs.	**rmon show user-history**
Show probe configuration.	**rmon show probe-config** [**basic**] [**net-config**] [**trap-dest**]

[1]To display Ethernet statistics and related statistics for WAN ports, RMON has to be activated on that port.

Note: WAN traffic received on a WAN port will reflect on the first physical port of the module only. Furthermore, when routing in destination-based forwarding mode, hardware limitations prevent the collection of source port information on all ports; thus, nl/al-matrix and addressMap tables cannot be updated, and only destination hosts will be updated in the nl/al-host tables.

To activate RMON on a port, use the **frame-relay define service** or **ppp define service** command, and the **frame-relay apply service** or **ppp apply service** command. For additional information, refer to *Setting up a Frame Relay Service Profile* on page 555 (for frame relay) and *Setting up a PPP Service Profile* on page 559 (for PPP).

RMON CLI Filters

Because a large number of statistics can be collected for certain RMON groups, you can define and use CLI filters to limit the amount of information displayed with the **rmon show** commands. An RMON CLI filter can only be applied to a current Telnet or Console session.

The following shows Host table output *without* a CLI filter:

```
xp# rmon show hosts et.5.4
RMON I Host Table

Index: 503, Port: et.5.4, Owner: monitor
Address           InPkts   InOctets   OutPkts   OutOctets   Bcst  Mcst
-------           ------   --------   -------   ---------   ----  ----
00001D:921086          0          0       102        7140     0     0
00001D:9D8138       1128      75196       885      114387      0     0
00001D:A9815F          0          0       102        7140     0     0
00105A:08B98D          0          0       971      199960     0     0
004005:40A0CD          0          0        51        3264     0     0
006083:D65800          0          0      2190      678372     0     0
0080C8:E0F8F3          0          0       396       89818     0     0
00E063:FDD700          0          0       104       19382     0     0
01000C:CCCCCC       2188     678210         0           0     0     0
01005E:000009        204      14280         0           0     0     0
0180C2:000000       1519      97216         0           0     0     0
030000:000001        168      30927         0           0     0     0
080020:835CAA        885     114387      1128       75196     0     0
980717:280200          0          0      1519       97216     0     0
AB0000:020000          2        162         0           0     0     0
FFFFFF:FFFFFF       1354     281497         0           0     0     0
```

The following shows the same **rmon show hosts** command with a filter applied so that only hosts with inpkts greater than 500 are displayed:

```
xp# rmon apply cli-filter 4
xp# rmon show hosts et.5.4
RMON I Host Table
Filter: inpkts > 500
Address            Port    InPkts  InOctets  OutPkts  OutOctets  Bcst  Mcst
-------            ----    ------  --------  -------  ---------  ----  ----
00001D:9D8138      et.5.4    1204     80110      941     121129     0     0
01000C:CCCCCC      et.5.4    2389    740514        0          0     0     0
0180C2:000000      et.5.4    1540     98560        0          0     0     0
080020:835CAA      et.5.4     940    121061     1204      80110     0     0
FFFFFF:FFFFFF      et.5.4    1372    285105        0          0
```

RMON CLI filters can only be used with the following groups:

- Hosts
- Matrix
- Protocol Distribution
- Application Layer Host
- Network Layer Host
- Application Layer Matrix
- Network Layer Matrix

Creating RMON CLI Filters

To create RMON CLI filters, use the following CLI command in Configure mode:

Creates an RMON CLI filter.	**rmon set cli-filter** <*filter-id*> <*parameter*>

Using RMON CLI Filters

To see and use RMON CLI filters, use the following CLI command in User or Enable mode:

Displays RMON CLI filters.	**rmon show cli-filters**
Applies a CLI filter on current Telnet or Console session.	**rmon apply cli-filters** <*filter-id*>
Clears the currently-selected CLI filter.	**rmon clear cli-filter**

Troubleshooting RMON

If you are not seeing the information you expected with an **rmon show** command, or if the network management station is not collecting the desired statistics, first check that the port is up. Then, use the **rmon show status** command to check the RMON configuration on the X-Pedition router. Check the following fields on the **rmon show status** command output:

```
xp# rmon show status
RMON Status
-----------
* RMON is ENABLED (1)
* RMON initialization successful.

          (2)
+--------------------------+
| RMON Group Status        |
+-------+--------+---------+
| Group | Status | Default |
+-------+--------+---------+
| Lite  |    On  |    Yes  |  (4)
+-------+--------+---------+
| Std   |    On  |    Yes  |
+-------+--------+---------+
| Pro   |    On  |    Yes  |
+-------+--------+---------+

RMON is enabled on: et.5.1, et.5.2, et.5.3, et.5.4, et.5.5, et.5.6, et.5.7, et.5.8 (3)

RMON Memory Utilization
-----------------------
      Total Bytes Available:   48530436

Total Bytes Allocated to RMON:    4000000
         Total Bytes Used:    2637872    (5)
         Total Bytes Free:    1362128
```

1. Make sure that RMON has been enabled on the X-Pedition router. When the router is booted, RMON is off by default. RMON is enabled with the **rmon enable** command.
2. Make sure that at least one of the RMON support levels—Lite, Standard, or Professional—is turned on with the **rmon set lite|standard|professional** command.
3. Make sure that RMON is enabled on the port for which you want statistics. Use the **rmon set ports** command to specify the port on which RMON will be enabled.
4. Make sure that the control table is configured for the report that you want. Depending upon the RMON group, default control tables may be created for all ports on the X-Pedition router. Or, if the RMON group is not one for which default control tables can be created, you will need to configure control table entries using the appropriate **rmon** command.

   If you or your application are unable to create a control table row, check the **snmp show status** output for errors. Make sure that there is a read-write community string. Verify that you can ping the X-Pedition router and that no ACLs prevent you from using SNMP to access the router.
5. Make sure that RMON has not run out of memory.

Allocating Memory to RMON

RMON allocates memory depending on the number of ports enabled for RMON, the RMON groups that have been configured, and whether or not default tables have been turned on or off. Enabling RMON with all groups (Lite, Standard, and Professional) with default tables uses approximately 300 Kbytes per port. If necessary, you can dynamically allocate additional memory to RMON.

To display the amount of memory that is currently allocated to RMON, use the following CLI command in Enable mode:

Displays the memory allocated to RMON.	**rmon show status**

Any memory allocation failures are reported. The following is an example of the information shown with the **rmon show status** command:

```
xp# rmon show status
RMON Status
-----------
* RMON is ENABLED
* RMON initialization successful.

+--------------------------+
| RMON Group Status |
+-------+--------+---------+
| Group | Status | Default |
+-------+--------+---------+
| Lite  |     On |     Yes |
+-------+--------+---------+
| Std   |     On |     Yes |
+-------+--------+---------+
| Pro   |     On |     Yes |
+-------+--------+---------+

RMON is enabled on: et.5.1, et.5.2, et.5.3, et.5.4, et.5.5, et.5.6, et.5.7, et.5.8

RMON Memory Utilization
-----------------------
      Total Bytes Available:     48530436
Total Bytes Allocated to RMON:     4000000
          Total Bytes Used:      2637872
          Total Bytes Free:      1362128
```

To set the amount of memory allocated to RMON, use the following CLI command in User or Enable mode:

Specifies the total amount of Mbytes of memory allocated to RMON.	**rmon set memory** *<number>*

Chapter 30

NetFlow Configuration Guide

Customers who invest in state-of-the-art switch-routers demand more than just routing services—they also need to know capacity utilization. Usage accounting, traffic profiling, traffic engineering, intrusion detection, network surveillance, QoS monitoring, and data warehousing and mining all require system utilization information. Enterasys X-Pedition routers provide a mechanism to gather and export utilization data for such applications.

NetFlow Activation and Data Collection Strategy

NetFlow deployment requires careful planning and service activation on strategically located edge/aggregation routers and WAN access routers which capture the data required for planning, monitoring, and accounting applications. Key deployment considerations include the following:

- Use NetFlow services as an edge metering and access list performance acceleration tool—not activated on “hot” core/backbone routers or routers running at very high CPU utilization rates.
- Understand your application-driven data collection requirements: accounting applications may require only originating and terminating router flow information whereas monitoring applications may require a more comprehensive, data-intensive end-to-end view.
- Understand the impact of your network topology and routing policy on the flow collection strategy. For example, avoid collecting duplicate flows by activating NetFlow on key aggregation routers where traffic originates or terminates—not on backbone or intermediate routers that would provide duplicate views of the same flow information.

Note: NetFlow cannot monitor traffic exits in a multicast environment. If you want to collect statistics on traffic moving through your system, you must monitor the input port(s).

Note: Hardware restrictions do not allow NetFlow to report a destination port for ICMP flows—the destination port is reported as 0.

Flow Accounting

Flow accounting refers to the gathering and management of data-forwarding, statistical information. The basic unit used to classify this information is called a *flow.* A flow is a set of packets that pass an observed point in the network during a specific time interval. Packets that belong to the same flow share common properties derived from the data contained in each packet or from packet treatment at the observing device. The *observed point* is the location in the router where flows are learned. The X-Pedition router uses the parameters below to identify layer-3 flows in destination-based, host-based, and application-based forwarding modes.

Note: Flow definitions will change if you adjust the forwarding mode. For additional information about forwarding modes, see **ip set port** in the *Enterasys X-Pedition Command Line Interface Reference Manual*.

- Source IP Address
- Destination IP Address
- UDP/TCP Source Address
- UDP/TCP Destination Address
- Type of Service Byte
- Protocol Byte
- Input Interface

The X-Pedition router records the following information for each flow:

- Flow start time
- Cumulative number of packets
- Cumulative number of bytes
- Flow termination time

Note: NetFlow is used to account for Layer-3 flows only.

Flow Accounting Applications

Because the requirements for NetFlow are derived from a variety of applications, selecting the appropriate application is crucial. The following sections contain NetFlow applications that are of significant importance in today's and future IP networks, and identify those requirements that can be met with reasonable technical effort. NetFlow applications are not limited to the implementations described here.

Since requirement details and weighting differ for specific implementations, NetFlow derives this information from the general functionality of the application. Furthermore, the application itself should lead to a better understanding of the requirements—particularly when designing or implementing a traffic flow measuring device.

Usage Accounting

NetFlow data provides fine-grained metering for highly flexible, detailed resource utilization accounting (e.g., flow data includes IP addresses, packet and byte counts, timestamps, type-of-service, and application ports). Service providers can utilize this information to migrate away from single-fee or flat-rate billing to more flexible charging mechanisms based on time of day, bandwidth usage, application usage, quality of service, or volume. Enterprise customers may use the information to establish a departmental charge-back or cost allocation for resource utilization.

Traffic Profiling

Traffic profiling is the process of characterizing IP flows and flow aggregates through a model used to represent key flow parameters (e.g., flow duration and volume). It is a prerequisite to network planning and dimensioning, trend analysis, developing business models, and other activities. Traffic profiling relies heavily on the particular traffic profiling objective(s)—what statistics and level of accuracy are required from the measurements. Typical information needed for traffic profiling includes the distribution of used network services and protocols, the amount of specific packet types, and flow profiles. Since objectives for traffic profiling vary, this application requires a highly flexible measurement infrastructure—especially with regard to the options for measurement configuration and packet classification. NetFlow provides this infrastructure.

Traffic Engineering

NetFlow enables network managers to gain a detailed, time-based view of application usage over the network. NetFlow data provides key information necessary to optimize strategic network planning and tactical network engineering decisions—whom to peer with, backbone upgrade planning, and routing policy planning. With this data, content and service providers can plan and allocate network and application resources (e.g., Web server sizing and location) to responsively meet customer demands. This will minimize the total cost of network operations and maximize network performance, capacity, and reliability.

Traffic Engineering (*TE*) is composed of methods for measurement, modeling, characterization, and network control. The goal of TE is to optimize network resource utilization and traffic performance [RFC2702]. Since control and administrative reaction to measurement results requires access to the involved network nodes, TE mechanisms and the required measurement function are usually performed within one administrative domain. Typical parameters required for TE are link utilization, load between specific network nodes, number, size and entry/exit points of the active flows, and routing information.

Attack/Intrusion Detection

Capturing flow information plays an important role in network security—especially when it comes to security violation detection and subsequent defense. In the case of a Denial of Service (DOS) attack, flow monitoring can detect unusual load situations or suspicious flows. NetFlow can export information about these flows and allow you to derive a defense strategy. Intrusion detection, a potentially more demanding application, looks not only at specific flow characteristics, but may also use a *stateful* packet flow analysis to detect suspicious or unusually frequent activities. Such activities may be characterized by specific communication patterns, detectable by characteristic sequences of certain packet types.

Network Surveillance

NetFlow data enables network managers to gain a detailed understanding of customer/user utilization of network and application resources. Network surveillance requires the collection and storage of information that characterizes flows—this information will help you detect and resolve potential security and policy violations and allow you to efficiently plan and allocate access to backbone and application resources. Since data collected for the purpose of network surveillance often contains sensitive information, adequate security mechanisms are required in order to prevent the misuse of sensitive information.

QoS Monitoring

NetFlow data enables extensive near-real-time network monitoring capabilities. Flow-based analysis techniques allow you to visualize *individual* router traffic patterns, plus *network-wide* patterns that include aggregate traffic-based or application-based views. These techniques provide proactive problem detection, efficient troubleshooting, and rapid problem resolution.

QoS monitoring is the non-intrusive, *passive* measurement of quality parameters for single flows or traffic aggregates. In contrast to intrusive, *active* measurements, non-intrusive measurements utilize existing network traffic for QoS analysis. Since they do not send test traffic, non-intrusive measurements most commonly apply in situations where the traffic of interest is already present in the network. For example, consider the validation of QoS parameters negotiated in a service-level specification (SLS). While non-intrusive measurements cannot provide the kind of controllable experiments you can achieve with active measurements, non-intrusive measurements do not suffer from undesired side effects caused by sending test traffic (e.g., additional load, “Heisenberg” effects, potential differences in treatment of test traffic, and real customer traffic).

QoS monitoring often requires correlating data from multiple measurement instances to measure one-way metrics. This requires proper clock synchronization of the involved measurement instances. For some measurements, you must correlate packet events at different measurement points—provisioning post-processing functions (e.g., packet ID generation) at the measurement instances may also be useful. QoS monitoring can lead to a huge amount of measurement result data; therefore, it may be beneficial to reduce measurement data such as aggregation of results and sampling.

NetFlow Data Warehousing and Mining

When you warehouse NetFlow data or derived information for later retrieval and analysis, you can determine which applications internal and external users utilize and target these services for improvement. This is especially useful when determining the relevant *who*, *what*, *where*, and *how long* information that will permit you to add greater depth to your services.

Supported Interfaces

NetFlow is supported only on interfaces that carry IP traffic. These interfaces are as follows:

- Ethernet
- Fast Ethernet
- Gigabit Ethernet
- ATM

Data Capture

NetFlow data capture is performed on a per-port basis. To enable accounting on a specific port, you must activate the port though the CLI. Once you enable a port, NetFlow accounts for all unicast and multicast traffic entering the port.

```
xp(config)# netflow set collector 10.136.2.9
xp(config)# netflow set ports et.6.1,et.7.1-3
xp(config)# netflow set memory 10000
xp(config)# netflow enable
```

Life of a Flow

The X-Pedition router is a hardware-based, packet forwarding device. With each packet received, the X-Pedition router uses hardware to determine if the packet destination is known. If the flow is new (i.e., not known), the X-Pedition router sends the packet to the control module to learn. Once the control module discovers or *learns* the packet's destination, the X-Pedition router initializes the line module hardware to recognize and forward subsequent packets with the same destination. For each packet received, the X-Pedition router establishes a flow entry containing the flow's start time, the number of packets received, and the cumulative packet byte count. For an illustration of this process, refer to Figure 44 on page 530.

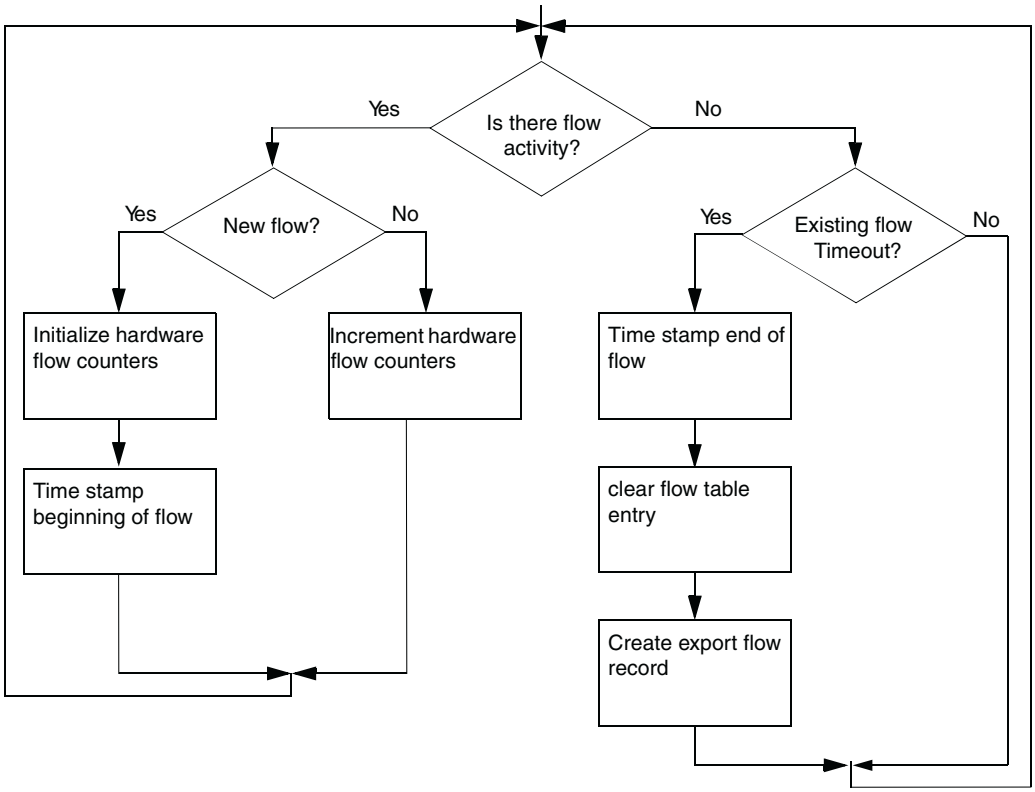
If a flow remains inactive after 30 seconds, its layer-3 hardware entry will age-out and the data will export according to the exportation policy.

Active flows export according to the interval timer setting. By default, the interval timer is set to 30. To adjust the interval timer setting, enter the following from configure mode:

```
xp# netflow set interval <minutes>
```

More information about the active flows export interval can be found in *Periodic Exportation* on page 535.

Figure 44. Life of a Flow



NetFlow Architecture

A NetFlow design is composed of four elements:

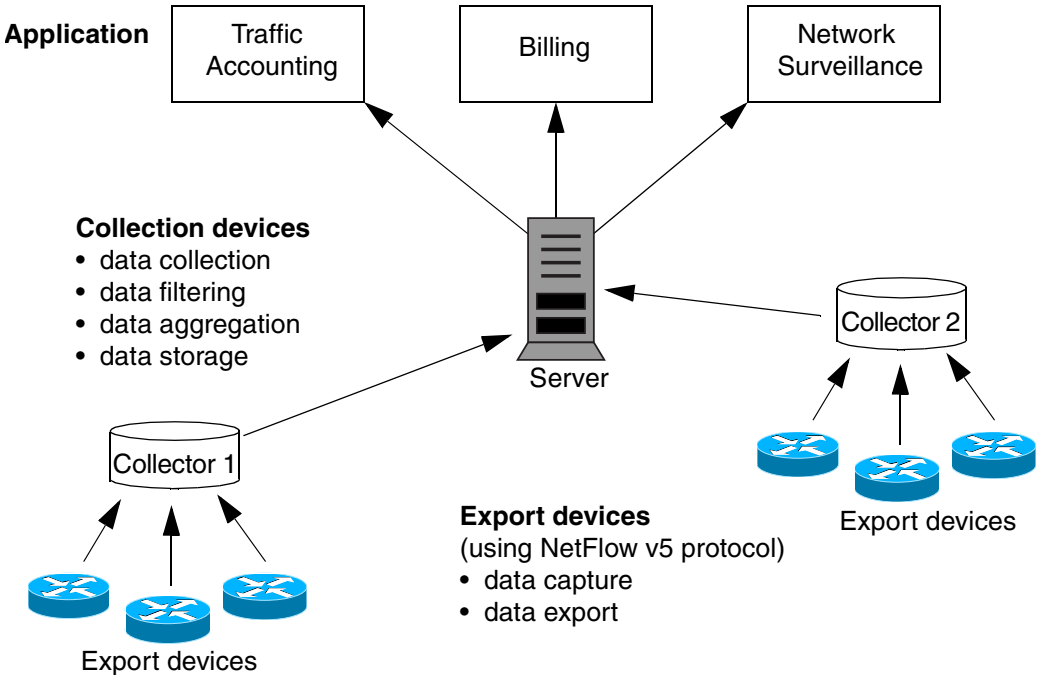
- Export device(s)
- Collection device(s)
- Server
- Applications

Export device(s) Detects flow characteristics and exports accounting information to a collector. The X-Pedition router is an export device that uses NetFlow v5 protocol.

Collection device(s) Gathers flow accounting information from one or more export devices.

Server Acts as a central repository for all data and serves as a single point of access for end-user applications (to formulate end-user reports). Servers can also perform post-collection data aggregation and time based consolidation. Applications access data on the server to formulate end-user reports.

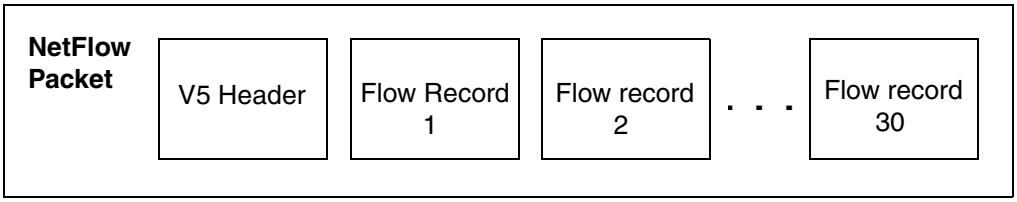
Applications In this architecture, Enterasys provides the capability to generate and export flow accounting data on the X-Pedition platforms. Third-party application vendors provide the collector, server, and end-user report generating components. This allows you to create complete accounting solutions where all the elements in a solution are compatible, they are not simply architectural pieces. X-Pedition routers support NetFlow v5 data exportation and are likely to be compatible with v5 applications. X-Pedition routers have been qualified to be compatible with the Inmon Traffic Server application.



How Does NetFlow Account for a Flow?

The X-Pedition router examines the flow parameters of each incoming packet. The system records the arrival time of the first packet in a new flow and updates the flow's cumulative byte and cumulative packet counters with each received packet. The system removes flows that have been idle for 30 seconds and records their termination time. NetFlow exports accounting information based on four flow metrics: start-time, end-time, cumulative byte count, and cumulative packet count. Figure 45 depicts the elements that comprise a NetFlow version 5 packet. Because NetFlow packets are UDP packets, packet delivery is a *best effort* delivery—the system will not attempt retries if delivery fails. A NetFlow packet may contain up to 30 flow records and one header.

Figure 45. NetFlow Version 5 Packets



The following tables describe the format of NetFlow Version 5 headers and flow records.

Table 18. Version 5 Header Format

Bytes	Content	Description
0 to 1	Version	NetFlow export format version number (in this case, 5).
2 to 3	Count	Number of flows (1-30) exported in this packet.
4 to 7	SysUptime	Number of milliseconds since the routing device was last booted.
8 to 11	unix_secs	Number of seconds since 0000 UTC 1970.
12 to 15	unix_nsecs	Number of residual nanoseconds since 0000 UTC 1970.
16 to 19	flow_sequence	Sequence counter of total flows seen.
20	engine_type	Type of flow switching engine.
21	engine_id	ID number of the flow switching engine.
22 to 23	reserved	

Table 19. Version 5 Flow Record Format

Bytes	Content	Description
0 to 3	srcaddr	Source IP address.
4 to 7	dstaddr	Destination IP address.
8 to 11	nexthop	IP address of the next hop routing device.
12 to 13	input	SNMP index of the input interface.
14 to 15	output	SNMP index of the output interface.
16 to 19	dPkts	Packets in the flow.
20 to 23	dOctets	Total number of Layer-3 bytes in the flow's packets.
24 to 27	First	SysUptime at start of flow.
28 to 31	Last	SysUptime at the time the last packet of flow was received.
32 to 33	srcport	TCP/UDP source port number or equivalent.
34 to 35	dstport	TCP/UDP destination port number or equivalent.
36	pad1	Pad 1 is unused (zero) bytes.
37	tcp_flags	Cumulative OR of TCP flags.
38	prot	IP protocol (e.g., 6=TCP, 17=UDP).
39	tos	IP ToS.
40 to 41	src_as	AS of the source address (origin or peer).
42 to 43	dst_as	AS of the destination address (origin or peer).
44	src_mask	Source address prefix mask bits.
45	dst_mask	Destination address prefix mask bits.
46 to 47	pad2	Pad 2 is unused (zero bytes).

Export Policy

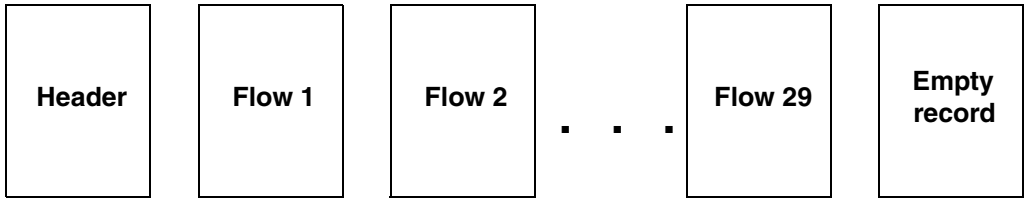
Policy governs the rate at which the X-Pedition router exports data. While some information exports asynchronously, other data outputs periodically according to a user-defined time interval.

Asynchronous Exportation

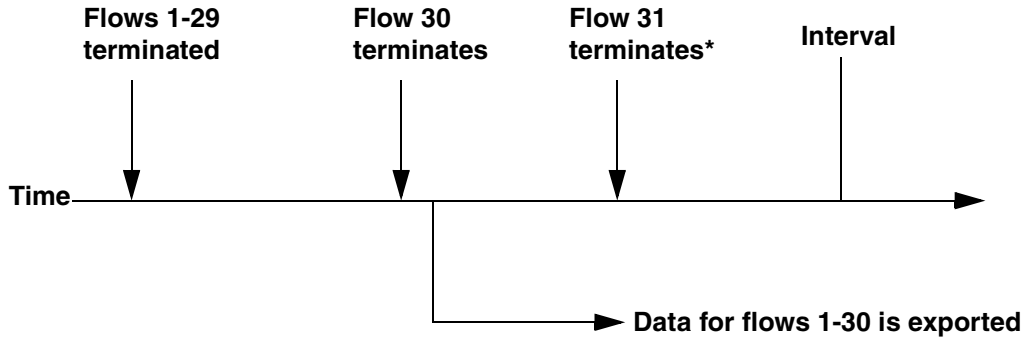
When a flow expires, NetFlow creates an export record. This record waits on the output queue until the total number of records in the queue reaches 30 (or until the interval time expires)—then the X-Pedition router transmits the expired flow data. Because the router must wait until the output queue is filled or until the interval time expires before it can export the data, it is not possible to determine precisely when the information for an expired flow will export to a collector. In general, systems with many expiring flows will export data more frequently.

Example

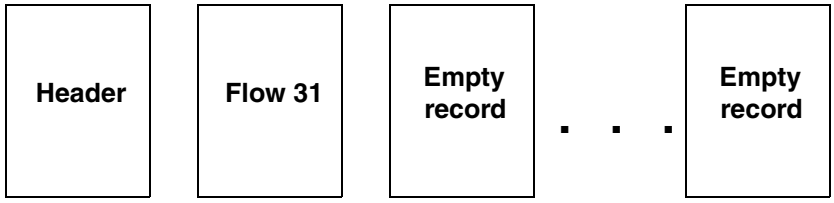
Since the previous interval, 29 flows terminated. A record for each resides in the current export buffer.



As demonstrated in the following figure, when flows 30 and 31 terminate, each termination results in the creation of an export record. Since the record for flow 30 fills the export buffer, data for flows 1-30 exports to a collector.



***** After flow 31 terminates, the export buffer appears as follows:



Periodic Exportation

Periodically, the system exports a flow accounting record for any *remaining inactive* flows and *all active* flows. By default, the periodic export occurs every 30 minutes. To change the export rate (1 to 1440 minutes—24 hours), enter the following command:

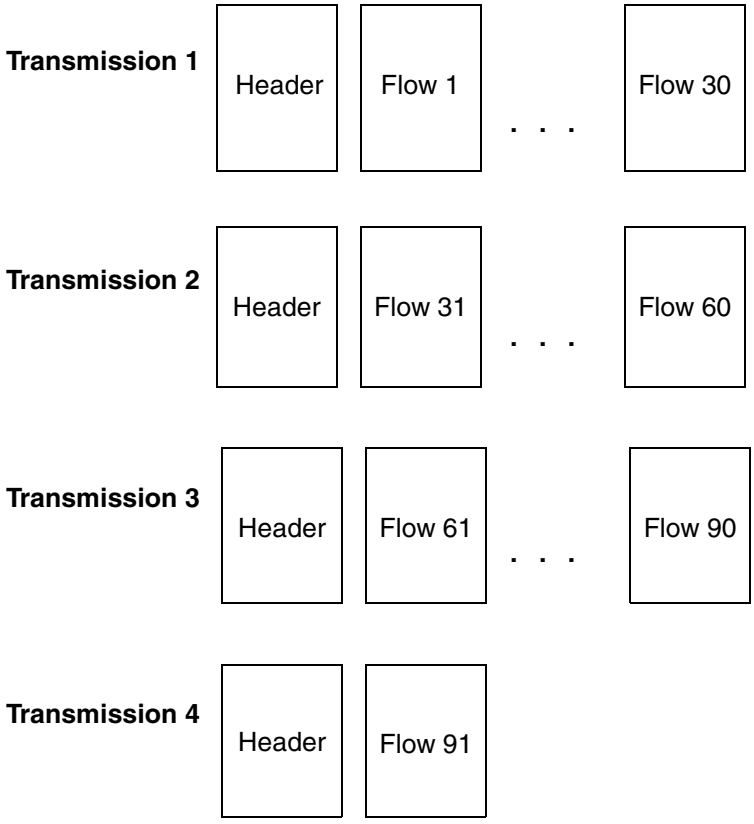
```
xp# netflow set interval <minutes>
```

Note: Because NetFlow packets are UDP packets, packet delivery is a *best effort* delivery—the system will not attempt retries if delivery fails.

At the conclusion of each interval, the X-Pedition router exports any buffered, expired flows along with all active flow data (a header and 30 flow records at a time). However, the final transmission may contain fewer than 30 records. For example, consider a system with 91 flows when the interval timer expires. Four export transmissions result—three consist of a header and 30 flow records, the final consists of a header and a single record. The transmission of flow records occurs as quickly as the router can buffer and output the data.

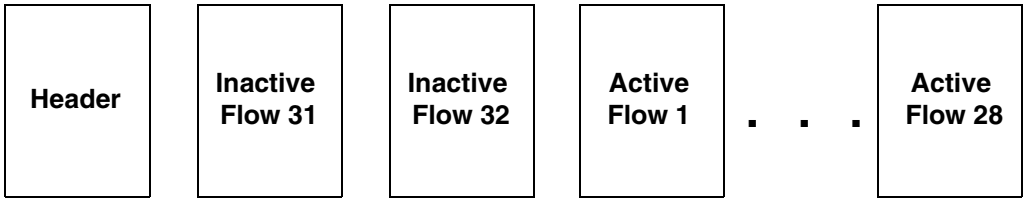
Example

In the following example, the X-Pedition router will export 91 flow records when the interval timer expires. Shown here, 4 transmissions result.



Asynchronous and Periodic Exportation

If the interval timer expires before NetFlow fills the export buffer with inactive flows, NetFlow will fill the remaining space with active flows. For example, consider a case where two expired flows queue for export when the interval expires. If twenty-eight or more flows are active, the first interval transmission will appear as follows:



Prerequisites

Before configure NetFlow, do the following:

- **Enable SNMP**. A system initialization task automatically enables the SNMP agent unless the **snmp stop** command is in the configuration file.
- **Disable RMON Professional**.

Restrictions

The following list the restrictions or limitations of using NetFlow:

- NetFlow cannot monitor traffic exits in a multicast environment. If you want to collect statistics on traffic moving through your system, you must monitor the input port(s).
- Hardware restrictions do not allow NetFlow to report a destination port for ICMP flows—the destination port is reported as 0.
- NetFlow is used to account for Layer-3 flows only.
- Virtual circuits and channelized interfaces do not support NetFlow.

Collector Relationship Management

Specifying Collectors

Described previously, the X-Pedition router exports accounting data to a collector. The **netflow set collector** command allows you to set NetFlow-related parameters, even override the default settings. Although a collector may service multiple routers, you may not enable multiple collectors for the same router.

netflow set collector <*collector_IPaddr*> [**flow-destination-port** <*number*>]

For detailed information about this command and its parameters, consult the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Configuring NetFlow Data Collection

The following table contains a summary of the NetFlow commands:

Table 20. NetFlow Commands

Mode	Syntax	Usage
Enable	**netflow clear statistics**	Optional
Configure	**netflow enable**	**Required**
Configure	**netflow set engine id** <*engine id*> **type** <*engine type*>	Optional
Configure	**netflow set flow-destination-port** <*port-number*>	Optional **Required prior to E9.0.0.0**
Configure	**netflow set interval** <*minutes*>	Optional
Configure	**netflow set memory** <*size*>	Recommended
Configure	**netflow set ports** <*port list*> | **all-ports**	Optional
Configure	**netflow set collector** <*collector_IPaddr*> **[flow-destination-port** <*number*>**]**	**Required**
Enable	**netflow show configuration| collector| statistics| status| historical max memory bytes| historical max memory time| all**	Optional

NetFlow commands set configuration parameters and display data. Configuration commands establish NetFlow internal task parameters and define the X-Pedition router/collector relationship. The commands used in the following example are required to set up accounting and exportation. These commands specify the ports whose flows will be accounted, identify the collector used to receive exported data, and activate the NetFlow feature. In this example, accounting and data exportation occur for the flows on all the Ethernet ports in slot 1. The collector application communicates on the default port, 2055. All other system defaults apply, including the 30-minute export interval.

```
netflow set ports et.1.*
netflow set collector 134.141.135.124
netflow enable
```

Overriding Default Parameters

Export Interval

By default, the X-Pedition router initializes the default export interval to 30 minutes. When setting the export interval, select an interval according to the type of application used. To set the interval (from 1 to 1440 minutes—24 hours), enter the following:

```
xp# netflow set interval <minutes>
```

System Memory Utilization

The NetFlow client acquires memory as needed from the system to store flow information entries. Unless otherwise configured by the **netflow set memory** command, the default limit on NetFlow's memory usage is 450 kilobytes.

The system calculates a NetFlow memory limit. Since the amount of memory being used by the system can change it frequently recalculates the limit. The calculation excludes memory NetFlow uses to store flow information entries. When NetFlow is enabled, the system will use the smaller of the system calculated NetFlow memory limit or the "netflow set memory" configuration value. To see the system calculated NetFlow memory limit, the limit from the configuration, or the limit the system is currently operating with, use the **netflow show statistics** command.

If the volume of flows exceeds the memory limit, new flows are not stored, flow information will be lost, and this error message will be displayed:

```
%NETFLOW-W-WARN_MSG, NetFlow Warning:  maximum flow memory limit crossed - insufficient
memory
```

NetFlow will resume storing new flow information once memory usage drops below the current operating limit. To view the number of flows not reported by NetFlow (discarded) due to lack of memory, use the **netflow show statistics** command.

NetFlow System Memory Utilization

While a flow is active, the flow information that NetFlow stores consumes 144 bytes per flow. A system anticipated to monitor 2000 active flows requires the following:

2,000 * 144 = 288,000 bytes

288,000 bytes is 281.3 kilobytes (288,000 / 1024)

When a flow remains inactive for 30 seconds it's data will be exported according to the exportation policy and the memory being used to store the flow information will be freed. To monitor NetFlow memory utilization or display information about peak NetFlow memory utilization, enter the following in enable mode:

```
xp# netflow show statistics
```

Collector Application Port

The *collector application port* refers to the flow destination port to which the X-Pedition router will send NetFlow exportation packets. The flow destination port (a UDP port) is part of the UDP header and the default destination port is 2055. To set a common port address for all collector applications, use the following command:

```
xp (config)# netflow set flow-destination-port <port-number>
```

Note: Hardware restrictions do not allow NetFlow to report a destination port for ICMP flows—the destination port is reported as 0.

Collector Engine ID and Type

The **netflow set engine** command allows you to modify the engine identification and type sent with a NetFlow packet header, affecting those NetFlow collectors that require specific engine values.

```
xp (config)# netflow set engine <engine id> type <engine type>
```

Show Command Output

To display all information about the NetFlow client on the X-Pedition router, use the **netflow show all** command. A sample output follows:

```
xp# netflow show all
NetFlow Status:
  NetFlow is ENABLED
  NetFlow Started at              : 2004-03-17 23:52:31

NetFlow Default Configuration:
  NetFlow Version                 : 5
  NetFlow Engine ID               : 0
  NetFlow Engine Type             : 0
  Active Flows Polling Interval   : 30
  Default Port                    : 2055

NetFlow Statistics:

-Intervals:
   Time of Last Reporting Interval:          Interval has not expired
   Time of Next Reporting Interval:             2004-03-18 00:22:31

-Memory:
   System limit on flow tracking memory:                    92259 K
   Configured limit on flow tracking memory:                27500 K
   Current limit on flow tracking memory:                   27500 K
   Amount of flow tracking memory in use:                    7704 K
   Percent of flow tracking memory in use:                     28 %

   Limit on flow tracking memory at peak:                   27500 K
   Flow tracking memory used at peak:                        7704 K
   Percent of flow tracking memory used at peak:               28 %
   Time of peak flow tracking memory usage:    2004-03-17 23:54:12

   Number of times NetFlow failed to get requested memory:       0

-Counters:
   Current number of active flows:                          54784
   Number of times netflow has sent reports:                   31
   Number of packets used to send reports:                    603
   Number of flows created in NetFlow:                      72890
   Number of flows deleted in NetFlow:                      18106
   Number of flows pending delete:                              0
   Number of flows not reported by NetFlow (discarded):         0
   Number of reported records (flows):                      18090

Ports Enabled for NetFlow:
              Port        Tracked       Tracked
ifIndex       Name         In Flows     Out Flows     Monitored
 ------------------------------------------------------------------------------
      1       et.6.1       18935         53958         ON
      9       et.7.1       53958         18935         ON
  Unknown Ports Flows :                    0

(Continued on next page....)
```

Continued from previous page:

```
Total Flows Count  :      72893       72893
 ------------------------------------------------------------------------------
  Number of Ports Being Monitored : 2

NetFlow Collector:
 IP Address:             10.136.2.9
 Accounting Port:        <Default>
```

Field Definitions

Field	Description
NetFlow is ENABLED/DISABLED	Status of the NetFlow agent.
NetFlow Started at	What time NetFlow started.
NetFlow Version	The version of NetFlow you are running.
NetFlow Engine ID	The NetFlow engine ID.
NetFlow Engine Type	The NetFlow engine type.
Active Flows Polling Interval	The configured reporting interval.
Threshold & Heap	Maximum percentage of heap to reach before NetFlow stops reporting flows.
Default Port	Default UDP destination port used to send export data to the collector.
Time of Last Reporting Interval	Date and time when the reporting interval last expired.
Time of Next Reporting	Date and time when the next synchronous update will be sent to the collector.
System limit on flow tracking memory	Memory limit the system calculated for NetFlow, in kilobytes.
Configured limit on flow tracking memory	**netflow set memory** configuration limit, in kilobytes.
Current limit on flow tracking memory	The current memory limit memory NetFlow is operating with, in kilobytes. It is the smaller of the two previous values.
Amount of flow tracking memory in use	Amount of memory NetFlow is currently using, in kilobytes.
Percent of flow tracking memory in use	Percent of NetFlow’s memory limit currently in use.

Field	Description
Limit on flow tracking memory at peak	The NetFlow memory limit when NetFlow memory usage peaked, in kilobytes.
Flow tracking memory used at peak	Kilobytes of NetFlow memory used when NetFlow memory usage peaked. This value typically will be more than what is currently in use. This value can be used to identify if memory usage is approaching the current NetFlow memory limit. This value resets when the **netflow clear statistics** command is entered.
Percent of flow tracking memory used at peak	Percentage of NetFlow's memory limit when NetFlow memory usage peaked. This value typically will be more than what is currently in use. This value resets when the **netflow clear statistics** command is entered.
Time of Peak memory usage	Date and time when memory used by NetFlow peaked.
Number of times NetFlow failed to get requested memory	When the NetFlow client's "Amount of flow tracking memory in use" exceeds the "Current limit on flow tracking memory", requests for memory to track flows fail and flow information is lost. This may indicate the NetFlow task requires more memory to collect flow information. If the "Configured limit on flow tracking memory" is less than the "System limit on flow tracking memory" then the amount of memory configure by **netflow set memory** command can be increased. Otherwise, decrease the number of ports NetFlow is configured on to prevent loss of flow information.
Current number of active flows	A count of the current active flows being recorded by the NetFlow task.
Number of times NetFlow has sent reports	A count of the number of NetFlow packets sent to the collector.
Number of packets used to send reports	A count of the number of packets NetFlow used to send the NetFlow records to the collector.
Number of flows created in NetFlow	A count of the number of created flows. You may reset this value.
Number of flows deleted in NetFlow	A count of the number of deleted flows.
Number of flows pending delete	A count of terminated flows that have been buffered and are waiting to be sent to the collector.
Number of flows not reported by NetFlow (discarded)	A count of flows discarded because of insufficient memory to store the flow information.

Field	Description
Number of reported records (flows)	A count of the number of NetFlow records sent to the collector.
Ports enabled for NetFlow	A list of ports which have NetFlow enabled.
Total Flows Count	The total number of flows.
Number of ports being monitored	The number of ports which have been enabled for NetFlow.
NetFlow Collectors Listed in Order of Priority	
Collector	IP address of collector
Accounting Port	The destination UDP port in use on this collector

Performance Measurements

Lab tests reveal that the NetFlow feature has very little impact on X-Pedition Control Module (CM) processor utilization. Measured impacts are in the .5 – 2.5% utilization range.

Chapter 31

WAN Configuration Guide

This chapter provides an overview of Wide Area Network (WAN) applications as well as an overview of both Frame Relay and PPP configuration for the X-Pedition router. In addition, you can view an example of a multi-router WAN configuration complete with diagram and configuration files in *WAN Configuration Examples* on page 562.

WAN Overview

On the X-Pedition router, WAN routing is performed over a serial interface using two basic protocols: Frame Relay and Point-to-Point Protocol (PPP). Both protocols have their own set of configuration and monitoring CLI commands described in the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Note: The X-Pedition router does not support an IPv6 interface configured over a WAN port. However, IPv6 tunnel traffic can egress the router via an IPv4 interface configured over a WAN port.

WAN Ports

Consider the following when working with WAN ports:

- WAN ports do not support STP per VLAN.
- You must add WAN ports to a non-default VLAN to allow flooding of packets.
- Layer-2 traffic will not bridge on WAN ports in the default VLAN.

High-Speed Serial Interface (HSSI) and Standard Serial Interfaces

In both the Frame Relay and PPP environments on the X-Pedition router, you can specify ports to be High-Speed Serial Interface (HSSI) or standard serial interface ports, depending, of course, on the type of hardware you have. Each type of interface plays a part in the nomenclature of port identification. You must use either the “hs.” or “se.” prefix for HSSI and serial interfaces, respectively, when specifying WAN port identities.

For example, you would specify a frame relay serial WAN port located at router slot 4, port 1, on VC 100 as “se.4.1.100”.

Using the same approach, a PPP high-speed serial interface (HSSI) WAN port located at router slot 3, port 2 would be identified as “hs.3.2”.

Configuring WAN Interfaces

Configuring IP & IPX interfaces for the WAN is generally the same as for the LAN. As with any IP network, when you assign an IP/IPX address to each interface and define routing mechanisms such as OSPF or RIP, you can configure the IP interface on the physical port or you can configure the interface as part of a VLAN. In the case of IP interfaces, you can configure multiple IP addresses for each interface. Please refer to *Configuring IP Interfaces and Parameters* on page 166 and *Configuring IPX Interfaces and Parameters on page 412*.

Note: A VLAN containing both LAN and WAN ports will not support interfaces configured with IPX.

Note: The maximum MTU for WAN interfaces is 1500 bytes.

Special considerations that apply to WAN interfaces are detailed in this section.

Primary and Secondary Addresses

Like LAN interfaces, WAN interfaces can have primary and secondary IP addresses. For Frame Relay, you can configure primary and secondary addresses which are static or dynamic. For PPP, however, the primary addresses may be dynamic or static, but the secondary addresses must be static. This is because the primary addresses of both the local and peer routers are exchanged during IPCP/IPXCP negotiation.

Note: There is no mechanism in PPP for obtaining any secondary addresses from the peer. For PPP, IPX network numbers for local and peer routers must match.

Static, Mapped, and Dynamic Peer IP/IPX Addresses

The following sections describe the difference between static, mapped, and dynamic peer IP and IPX addresses and provide simple command line examples for configuration.

Static Addresses

If the peer IP/IPX address is known before system setup, you can specify the peer address when the interface is created. This disables Inverse ARP (InArp) for Frame Relay on that source/peer address pair; however, InArp will still be enabled for any other addresses on that interface or other interfaces. A static peer address for PPP means that the address the peer supplies during IP Control Protocol (IPCP) or IPX Control Protocol (IPXCP) negotiations will be ignored.

The following command line displays an example for a port:

```
interface create ip IPWAN address-netmask 10.50.1.1/16 peer-address 10.50.1.2 port hs.3.1
```

The following command line displays an example for a VLAN:

```
interface create ip IPWAN address-netmask 10.50.1.1/16 peer-address 10.50.1.2 vlan BLUE
```

Mapped Addresses

Mapped peer IP/IPX addresses are very similar to static addresses in that InArp is disabled for Frame Relay and the address negotiated in IPCP/IPXCP is ignored for PPP.

Mapped addresses are most useful when you do not want to specify the peer address using the **interface create** command. This would be the case if the interface is created for a VLAN and there are many peer addresses on the VLAN. If any of the peers on the VLAN do not support InArp or IPCP/IPXCP, use a mapped address to configure the peer address.

The following command lines display two examples for Frame Relay:

```
frame-relay set peer-address ip-address 10.50.1.1/16 ports se.4.1.204
```

```
frame-relay set peer-address ipx-address a1b2c3d4.aa:bb:cc:dd:ee:ff ports se.6.3.16
```

The following command line displays two examples for PPP:

```
ppp set peer-address ip-address 10.50.1.1/16 ports se.4.1
```

```
ppp set peer-address ipx-address a1b2c3d4.aa:bb:cc:dd:ee:ff ports se.6.3
```

Dynamic Addresses

If the peer IP/IPX address is unknown, you do not need to specify it when creating the interface. When in the Frame Relay environment, the peer address will be automatically discovered via InArp.

Note: The default interface type is “broadcast.” If you connect an interface to a router capable of “point-to-point” only, you must specify the interface type as point-to-point within the configuration.

Similarly, the peer address will be automatically discovered via IPCP/IPXCP negotiation in a PPP environment.

The following command lines display examples for a port and a VC:

```
interface create ip IPWAN address-netmask 10.50.1.1/16 port hs.3.1
```

```
interface create ip IPWAN address-netmask 10.50.1.1/16 port hs.5.2.19
```

The following command line displays an example for a VLAN:

```
interface create ip IPWAN address-netmask 10.50.1.1/16 vlan BLUE
```

Forcing Bridged Encapsulation

WAN for the X-Pedition router has the ability to force bridged packet encapsulation. This feature has been provided to facilitate seamless compatibility with Cisco routers, which expect bridged encapsulation in certain operating modes.

The following command line displays an example for Frame Relay:

```
frame-relay set fr-encaps-bgd ports hs.5.2.19
```

The following command line displays an example for PPP:

```
ppp set ppp-encaps-bgd ports hs.5.2
```

Packet Compression

Packet compression can increase throughput and shorten the times needed for data transmission. You can enable packet compression for Frame Relay VCs and for PPP ports, however, both ends of a link must be configured to use packet compression.

Enabling compression on WAN serial links should be decided on a case by case basis. Important factors to consider include:

- average packet size
- nature of the data
- link integrity
- latency requirements

Each of these factors is discussed in more detail in the following sections and should be taken into consideration before enabling compression. Since the factors are dependent on the environment, you should first try running with compression histories enabled. If compression statistics do not show a very good long-term compression ratio, then select the “no history” option. If the compression statistics do not improve or show a ration of less than 1, then compression should be disabled altogether.

Average Packet Size

In most cases, the larger the packet size, the better the potential compression ratio. This is due to the overhead involved with compression, as well as the compression algorithm. For example a link which always deals with minimum size packets may not perform as well as a link whose average packet size is much larger.

Nature of the Data

In general, data that is already compressed cannot be compressed any further. In fact, packets that are already compressed will grow even larger. For example, if you have a link devoted to streaming MPEG videos, you should *not* enable compression as the MPEG video data is already compressed.

Link Integrity

Links with high packet loss or links that are extremely over-subscribed may not perform as well with compression enabled. If this is the situation on your network, you should *not* enable compression histories (this applies only to PPP compressions; in Frame Relay compression, histories are always used).

Compression histories take advantage of data redundancy *between* packets. In an environment with high packet loss or over-subscribed links, there are many gaps in the packet stream resulting in very poor use of the compression mechanism. Compression histories work best with highly-correlated packet streams. Thus, a link with fewer flows will generally perform better than a link with many flows when compression histories are utilized.

The “no history” (max-histories = 0) option causes packets to be compressed on a packet-by-packet basis, thus packet loss is not a problem. Also, the number of flows is not an issue with this option as there is no history of previous packets.

Latency Requirements

The use of compression may affect a packet’s latency. Since the compressed packet is smaller, less time is needed to transmit it. On the other hand, each packet must undergo a compression/decompression process. Since the compression ratio will vary, the amount of latency will also vary.

Example Configurations

The following command line displays an example for Frame Relay:

```
frame-relay set payload-compress ports se.3.1.300
```

The following command line displays an example for PPP:

```
ppp set payload-compress port se.4.2
```

Packet Encryption

Packet encryption allows data to travel through unsecured networks. You can enable packet encryption for PPP ports, however, both ends of a link must be configured to use packet encryption:

```
ppp set payload-encrypt port se.4.2, mp.1
```

Note: After the router executes the **ppp set payload-encrypt** command, the CLI will prompt the user for password information.

WAN Quality of Service

Increasing concentrations of audio, video, and data traffic are now presenting the networking industry with the significant challenge of employing the most effective use of WAN Quality-of-Service (QoS) as possible to ensure reliable end-to-end communication. For example, critical and time-sensitive traffic such as audio should have higher levels of bandwidth allocated than less time-sensitive traffic such as file transfers or e-mail. Simply adding more and more bandwidth to a network is not a viable solution to the problem. WAN access is extremely expensive, and there is a limited (albeit huge) supply. Therefore, making the most effective use of existing bandwidth is now a more critical issue than ever before.

The fact that IP communications to the desktop are clearly the most prevalent used today has made it the protocol of choice for end-to-end audio, video, and data applications. This means that the challenge for network administrators and developers has been to construct their networks to support these IP-based audio, video, and data applications along with their tried-and-true circuit-based applications over a WAN.

In addition, these audio, video, and data traffic transmissions hardly ever flow at a steady rate. Some periods will see relatively low levels of traffic, and others will temporarily surpass a firm's contracted Committed Information Rate (CIR). Carrier-based packet-switched networks such as Frame Relay and ATM are designed to handle these temporary peaks in traffic, but it is more cost- and resource- efficient to employ effective QoS configuration(s), thus relaxing the potential need to up your firm's CIR. By applying some of the following sorts of attributes to interfaces on your network, you can begin to shape your network's QoS configuration to use existing bandwidth more effectively.

Source Filtering and ACLs

Source filtering and ACLs can be applied to a WAN interface; however, they affect the entire module, not an individual port. For example, if you want to apply a source MAC address filter to a WAN serial card located in slot 5, port 2, your configuration command line would look like the following:

```
xp(config)# filters add address-filter name wan1 source-mac 000102:030405 vlan 2 in-port-list se.5
```

Port se.5 is specified instead of se.5.2 because source filters affect the entire WAN module. Hence, in this example, **source-mac 000102:030405** would be filtered from ports se.5.1, se.5.2, se.5.3, and se.5.4 (assuming that you are using a four-port serial card).

ACLs work in a similar fashion. For example, if you define an ACL to deny all http traffic on one of the WAN interfaces, it will apply to the other WAN interfaces on that module as well. In practice, by making your ACLs more specific, for example by specifying source and destination IP addresses with appropriate subnet masks, you can achieve your intended level of control.

Weighted-Fair Queueing

Through the use of Weighted-Fair Queueing QoS policies, WAN packets with the highest priority can be allotted a sizable percentage of the available bandwidth and “whisked through” WAN interface(s). Meanwhile, the remaining bandwidth is distributed for “lower-priority” WAN packets according to the user’s percentage-of-bandwidth specifications. Please refer to the *Enterasys X-Pedition Command Line Interface Reference Manual* for more detailed QOS command configuration information.

Note: Weighted-Fair Queueing applies only to best-effort traffic on the WAN card. If you apply any of the WAN specific traffic shaping commands, then weighted fair queuing will no longer be applicable.

Congestion Management

One of the most important features of configuring the X-Pedition router to ensure Quality of Service is the obvious advantage gained when you are able to avoid network congestion. The following topics touch on a few of the most prominent aspects of congestion avoidance when configuring the X-Pedition router.

Random Early Discard (RED)

RED allows network operators to manage traffic during periods of congestion based on policies. Random Early Discard (RED) works with TCP to provide fair reductions in traffic proportional to the bandwidth being used. Weighted Random Early Discard (WRED) works with IP Precedence or priority, as defined in the **qos** configuration command line, to provide preferential traffic handling for higher-priority traffic.

The CLI commands related to RED in both the Frame Relay and PPP protocol environments allow you to set maximum and minimum threshold values for each of the low-, medium-, and high-priority categories of WAN traffic.

Adaptive Shaping

Adaptive shaping implements the congestion-sensitive rate adjustment function and has the following characteristics:

- No blocking of data flow under normal condition if the traffic rate is below Bc+Be
- Reduction to a lower CIR upon detection of network congestion
- Progressive return to the negotiated information transfer rate upon congestion abatement

The CLI command related to adaptive shaping allows you to set threshold values for triggering the adaptive shaping function.

Frame Relay Overview

Frame relay interfaces are commonly used in a WAN to link several remote routers together via a single central switch. This eliminates the need to have direct connections between all of the remote members of a complex network, such as a host of corporate satellite offices. The advantage that Frame Relay offers to this type of geographic layout is the ability to switch packet data across the interfaces of different types of devices like switch-routers and bridges, for example.

Frame Relay employs the use of Virtual Circuits (VCs) when handling multiple logical data connections over a single physical link between different pieces of network equipment. The Frame Relay environment, by nature, deals with these connections quite well through its extremely efficient use of precious (sometimes scarce) bandwidth.

You can set up frame relay ports on your X-Pedition router with the frame-relay commands described in the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Virtual Circuits

You can think of a Virtual Circuit (VC) as a “virtual interface” (sometimes referred to as “sub-interfaces”) over which Frame Relay traffic travels. Frame Relay interfaces on the X-Pedition router use one or more VCs to establish bidirectional, end-to-end connections with remote end points throughout the WAN. For example, you can connect a series of multi-protocol routers in various locations using a Frame Relay network.

Permanent Virtual Circuits (PVCs)

WAN interfaces can take advantage of connections that assure a minimum level of available bandwidth at all times. These standing connections, called Permanent Virtual Circuits (PVCs), allow you to route critical packet transmissions from host to peer without concern for network congestion significantly slowing, let alone interrupting, your communications. PVCs are the most prevalent type of circuit used today and are similar to dedicated private lines in that you can lease and set them up through a service provider.

In a corporate setting, network administrators can use PVCs in an internal network to set aside bandwidth for critical connections, such as videoconferencing with other corporate departments.

Note: Interfaces configured with PVCs do not support LSNAT and VRRP.

Configuring Frame Relay Interfaces for the Router

This section provides an overview of configuring a host of WAN parameters and setting up WAN interfaces. When working in the Frame Relay protocol environment, you must first define the type and location of the WAN interface. Having established the type and location of your WAN interfaces, you need to (optionally) define one or more service profiles for your WAN interfaces, then apply a service profile to the desired interface(s). An example of this process is covered in *Frame Relay Port Configuration* on page 556.

Defining the Type and Location of a Frame Relay and VC Interface

To configure a frame relay WAN port, you need to first define the type and location of one or more frame relay WAN ports or virtual circuits (VCs) on your X-Pedition router. The following command line displays a simplified example of a frame relay WAN port definition:

Define the type and location of a frame relay WAN port.	**port set** <*port*> **wan-encapsulation frame-relay speed** <*number*>

Note: If the port is a HSSI port that will be connected to a HSSI port on another router, you can also specify **clock** <*clock-source*> in your definition.

Then, you must set up a frame relay virtual circuit (VC). The following command line displays a simplified example of a VC definition:

Define the type and location of a frame relay VC.	**frame-relay create vc** <*port*>

Setting up a Frame Relay Service Profile

Once you have defined the type and location of your Frame Relay WAN interface(s), you can configure your X-Pedition router to more efficiently utilize available bandwidth for Frame Relay communications.

Note: The X-Pedition router comes with a set of "default values" for Frame Relay interface configuration settings, which means that setting up a Frame Relay service profile is not absolutely necessary to begin sending and receiving Frame Relay traffic on your router.

After you configure one or more service profiles for your Frame Relay interface(s), you can then apply a service profile to active Frame Relay WAN ports, specifying their behavior when handling Frame Relay traffic. The following command line displays all of the possible attributes used to define a Frame Relay service profile:

Define a frame relay service profile.	**frame-relay define service** <*service name*> [**Bc** <*number*>] [**Be** <*number*>] [**becn-adaptive-shaping** <*number*>] [**cir** <*number*>] [**high-priority-queue-depth** <*number*>] [**low-priority-queue-depth** <*number*>] [**med-priority-queue-depth** <*number*>] [**red on**|**off**] [**red-maxTh-high-prio-traffic** <*number*>] [**red-maxTh-low-prio-traffic** <*number*>] [**red-maxTh-med-prio-traffic** <*number*>] [**red-minTh-high-prio-traffic** <*number*>] [**red-minTh-low-prio-traffic** <*number*>] [**red-minTh-med-prio-traffic** <*number*>] [**rmon on**|**off**]

Applying a Service Profile to an Active Frame Relay WAN Port

Once you have created one or more Frame Relay service profiles, you can specify their use on one or more active Frame Relay WAN ports on the X-Pedition router. The following command line displays a simplified example of this process:

Apply a service profile to an active WAN port.	**frame-relay apply service** <*service name*> **ports** <*port list*>

Monitoring Frame Relay WAN Ports

Once you have configured your frame relay WAN interface(s), you can use the CLI to monitor status and statistics for your WAN ports. The following table describes the monitoring commands for WAN interfaces, designed to be used in Enable mode:

Display a particular frame relay service profile	**frame-relay show service** <*service name*>
Display all available frame relay service profiles	**frame-relay show service all**
Display the last reported frame relay error	**frame-relay show stats port** <*port name*> **last-error**
Display active frame relay LMI parameters	**frame-relay show stats port** <*port name*> **lmi**
Display MIBII statistics for frame relay WAN ports	**frame-relay show stats port** <*port name*> **mibII**
Display a summary of all LMI statistics	**frame-relay show stats port** <*port name*> **summary**

Frame Relay Port Configuration

To configure Frame Relay WAN ports, you must first define the type and location of the WAN interface, optionally “set up” a library of configuration settings, then apply those settings to the desired interface(s). The following examples are designed to give you a small model of the steps necessary for a typical Frame Relay WAN interface specification.

To define the location and identity of a serial Frame Relay WAN port located at slot 5, port 1 with a speed rating of 45 million bits per second:

```
xp(config)# port set se.5.1 wan-encapsulation frame-relay speed 45000000
```

To define the location and identity of a High-Speed Serial Interface (HSSI) VC located at slot 4, port 1 with a DLC of 100:

```
xp(config)# frame-relay create vc hs.4.1.100
```

Suppose you wish to set up a service profile called “profile1” that includes the following characteristics:

- Committed burst value of 2 million and excessive burst value of 1 million
- BECN active shaping at 65 frames
- Committed information rate (CIR) of 20 million bits per second
- Leave high-, low-, and medium-priority queue depths set to factory defaults
- Random Early Discard (RED) disabled
- RMON enabled

The command line necessary to set up a service profile with the above attributes would be as follows:

```
xp(config)# frame-relay define service profile1 Bc 2000000 Be 10000000 becn-adaptive-shaping 65 cir
20000000 red off rmon on
```

To assign the above service profile to the VC interface created earlier (slot 4, port 1):

```
xp(config)# frame-relay apply service profile1 ports hs.4.1.100
```

Point-to-Point Protocol (PPP) Overview

Because of its ability to quickly and easily accommodate IP and IPX protocol traffic, Point-to-Point Protocol (PPP) routing has become a very important aspect of WAN configuration. Using PPP, you can set up router-to-router, host-to-router, and host-to-host connections.

Note: For PPP, IPX network numbers for local and peer routers must match.

Establishing a connection in a PPP environment requires that the following events take place:

- The router initializing the PPP connection transmits Link Control Protocol (LCP) configuration and test frames to the remote peer to set up the data link.
- Once the connection has been established, the router that initiated the PPP connection transmits a series of Network Control Protocol (NCP) frames necessary to configure one or more network-layer protocols.
- Finally, when the network-layer protocols have been configured, both the host and remote peer can send packets to one another using any and all of the configured network-layer protocols.

The link will remain active until explicit LCP or NCP frames instruct the host and/or the peer router to close the link, or until some external event (i.e., user interruption or system time-out) takes place.

You can set up PPP ports on your X-Pedition router with the commands described in the *Enterasys X-Pedition Command Line Interface Reference Manual*.

Use of LCP Magic Numbers

LCP magic numbers enable you to detect situations where PPP LCP packets are looped back from the remote system, resulting in an error message. The use of LCP magic numbers is enabled on the X-Pedition router by default; however, should you employ a service profile in which the use of LCP magic numbers has been disabled, undetected “loopback” behavior may become a problem.

Note: In the event that a PPP WAN interface remains unrecognized at startup due to loopback interference, you can use the **ppp restart** command in the CLI to remedy the situation.

Configuring PPP Interfaces

This section provides an overview of configuring a host of WAN parameters and setting up WAN interfaces. When working in the PPP environment, you must first define the type and location of your WAN interfaces. Having established the type and location of your WAN interfaces, you need to (optionally) define one or more service profiles for your WAN interfaces, then apply a service profile to the desired interface(s). Examples of this process are displayed in *PPP Port Configuration* on page 561.

Defining the Type and Location of a PPP Interface

To configure a PPP WAN port, you need to first define the type and location of one or more PPP WAN ports on your X-Pedition router. The following command line displays a simplified example of a PPP WAN port definition:

Define the type and location of a PPP WAN port.	**port set** <*port*> **wan-encapsulation ppp speed** <*number*>

If the port is an HSSI port that will be connected to a HSSI port on another router, you can specify **clock** <*clock-source*> in the definition. (This feature is supported on HSSI boards, part number XP-HSSI-02-AA.)

Setting up a PPP Service Profile

Once you have defined the type and location of your PPP WAN interface(s), you can configure your X-Pedition router to more efficiently utilize available bandwidth for PPP communications.

Note: The X-Pedition router comes with a set of "default values" for PPP interface configuration settings, which means that setting up a PPP service profile is not absolutely necessary to begin sending and receiving PPP traffic on your router.

After you configure one or more service profiles for your PPP interface(s), you can then apply a service profile to active PPP WAN ports, specifying their behavior when handling PPP traffic. The following command line displays all of the possible attributes used to define a PPP service profile:

Define a PPP service profile.	**ppp define service** <*service name*> [**bridging enable**|**disable ip enable**|**disable ipx enable**|**disable**] [**high-priority-queue-depth** <*number*>] [**lcp-echo on**|**off**] [**lcp-magic on**|**off**] [**low-priority-queue-depth** <*number*>] [**max-configure** <*number*>] [**max-failure** <*number*>] [**max-terminate** <*number*>] [**med-priority-queue-depth** <*number*>] [**red on**|**off**] [**red-maxTh-high-prio-traffic** <*number*>] [**red-maxTh-low-prio-traffic** <*number*>] [**red-maxTh-med-prio-traffic** <*number*>] [**red-minTh-high-prio-traffic** <*number*>] [**red-minTh-low-prio-traffic** <*number*>] [**red-minTh-med-prio-traffic** <*number*>] [**retry-interval** <*number*>] [**rmon on**|**off**]

Note: If it is necessary to specify a value for Bridging, IP, and/or IPX, you must specify all three of these values at the same time. You cannot specify just one or two of them in the command line without the other(s).

Applying a Service Profile to an Active PPP Port

Once you have created one or more PPP service profiles, you can specify their use on one or more active PPP ports on the X-Pedition router. The following command line displays a simplified example of this process:

Apply a service profile to an active WAN port.	**ppp apply service** <*service name*> **ports** <*port list*>

Configuring Multilink PPP Bundles

The Multilink PPP (MLP) standard defines a method for grouping multiple physical PPP links into a logical pipe, called an “MLP bundle”. Large packets are fragmented and transmitted over each physical link in an MLP bundle. At the destination, MLP reassembles the packets and places them in their correct sequence.

The following table describes the commands for configuring MLP:

Add PPP port(s) to an MLP bundle.	**ppp add-to-mlp** <*mlp* > **port** <*port list*>
Create MLP bundle(s).	**ppp create-mlp** <*mlp list*> **slot** <*number*>
Set MLP encapsulation format.	**ppp set mlp-encaps-format ports** <*port list*> [**format short-format**]
Set the size of frames that fragmented for transmission on an MLP bundle.	**ppp set mlp-frag-size ports** <*port list*> **size** <*size* >
Set the depth of the queue used to hold MLP packets for preserving the packet order.	**ppp set mlp-orderq-depth ports** <*port list*> **qdepth** <*number-of-packets*>
Set the depth of the queue used to hold packet fragments for reassembly.	**ppp set mlp-fragq-depth ports** <*port list*> **qdepth** <*number-of-packets*>

Compression on MLP Bundles or Links

Compression can be applied on either a bundle or link basis if MLP is enabled on PPP links. If compression is enabled on a bundle, the packets will be compressed *before* processing by MLP. If compression is enabled on a link, the packets will be compressed *after* the MLP processing.

In general, choose bundle compression over link compression whenever possible. Compressing packets before they are “split” by MLP is much more efficient for both the compression algorithm and the WAN card. Link compression is supported to provide the widest range of compatibility with other vendors’ equipment.

Monitoring PPP WAN Ports

Once you have configured your PPP WAN interface(s), you can use the CLI to monitor status and statistics for your WAN ports.

Note: WAN traffic received on a WAN port reflects on the first physical port of the module only.

The following table describes the monitoring commands for WAN interfaces, designed for use in Enable mode:

Display a particular PPP service profile.	**ppp show service** <*service name*>
Display all available PPP service profiles.	**ppp show service all**
Display bridge NCP statistics for specified PPP WAN port.	**ppp show stats port** <*port name*> **bridge-ncp**
Display IP NCP statistics for specified PPP WAN port.	**ppp show stats port** <*port name*> **ip-ncp**
Display link-status statistics for a specified PPP WAN port.	**ppp show stats port** <*port name*> **link-status**
Displays information for PPP ports that are added to MLP bundles.	**ppp show mlp** <*mlp list*>|**all-ports**

PPP Port Configuration

To configure PPP WAN ports, you must first define the type and location of the WAN interface, optionally “set up” a library of configuration settings, then apply those settings to the desired interface(s). The following examples are designed to give you a small model of the steps necessary for a typical PPP WAN interface specification.

To define the location and identity of a High-Speed Serial Interface (HSSI) PPP WAN port located at router slot 5, port 1 with a speed rating of 45 million bits per second:

```
xp(config)# port set hs.5.1 wan-encapsulation ppp speed 45000000
```

Suppose you wish to set up a service profile called “profile2” that includes the following characteristics:

- Bridging enabled
- Leave high-, low-, and medium-priority queue depths set to factory defaults
- IP and IPX enabled

- Sending of LCP Echo Requests disabled
- Use of LCP magic numbers disabled
- The maximum allowable number of unanswered requests set to 8
- The maximum allowable number of negative-acknowledgment transmissions set to 5
- The maximum allowable number of unanswered/improperly answered connection-termination requests before declaring the link to a peer lost set to 4
- Random Early Discard disabled
- The number of seconds between subsequent configuration request transmissions (the “retry interval”) set to 25
- RMON enabled

The command line necessary to set up a service profile with the above attributes would be as follows:

```
xp(config)# ppp define service profile2 bridging enable ip enable ipx enable lcp-echo off lcp-magic off
    max-configure 8 max-failure 5 max-terminate 4 red off retry-interval 25 rmon on
```

To assign the above service profile to the active PPP WAN port defined earlier (slot 5, port 1):

```
xp(config)# ppp apply service profile2 ports hs.5.1
```

WAN Configuration Examples

Simple Configuration File

The following is an example of a simple configuration file used to test frame relay and PPP WAN ports:

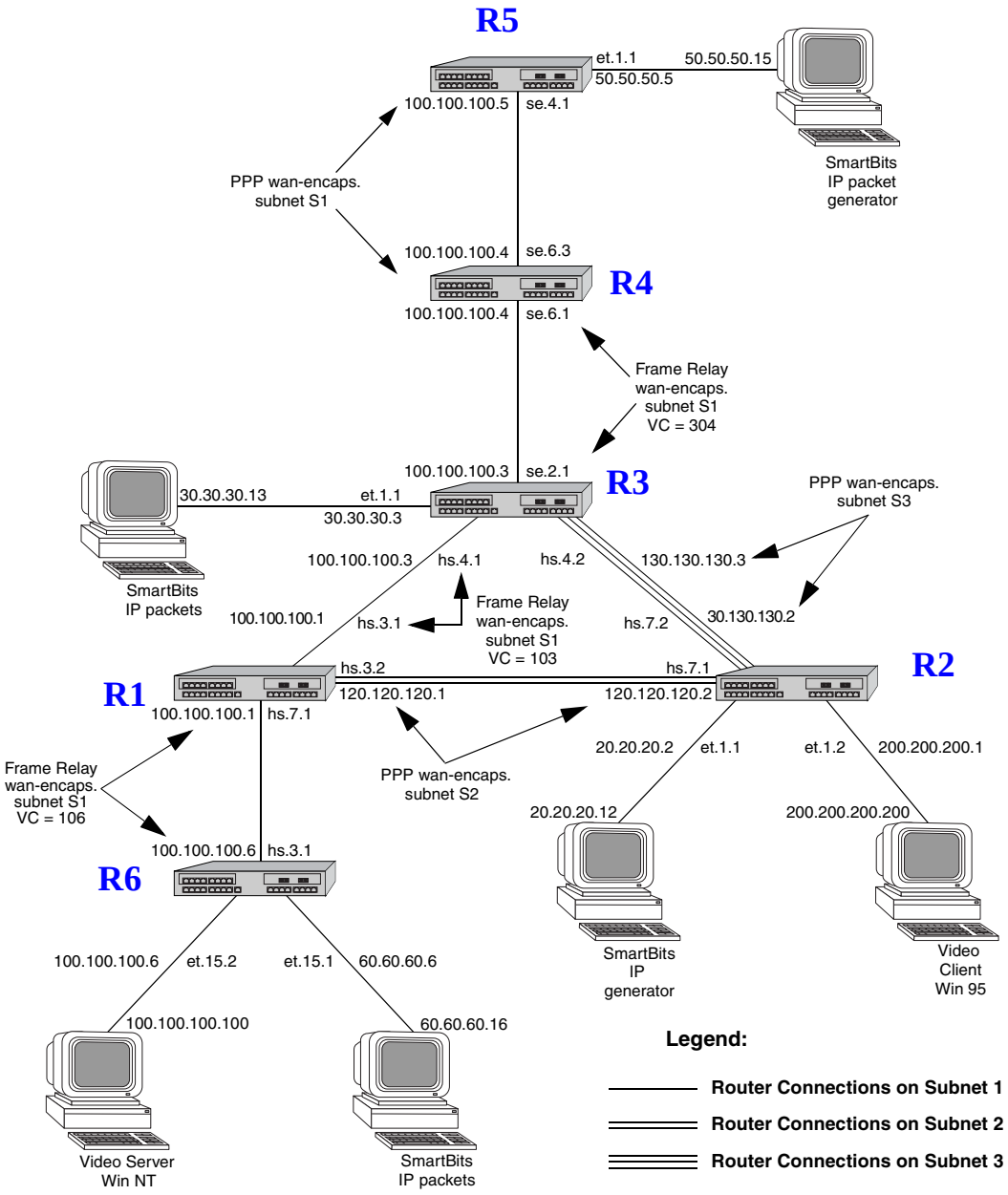
```
port set hs.5.1 wan-encapsulation frame-relay speed 45000000
port set hs.5.2 wan-encapsulation ppp speed 45000000
interface create ip fr1 address-netmask 10.1.1.1/16 port hs.5.1.100
interface create ip ppp2 address-netmask 10.2.1.1/16 port hs.5.2
interface create ip lan1 address-netmask 10.20.1.1/16 port et.1.1
interface create ip lan2 address-netmask 10.30.1.1/16 port et.1.2
ip add route 10.10.0.0/16 gateway 10.1.1.2
ip add route 10.40.0.0/16 gateway 10.2.1.2
```

For a broader, more application-oriented WAN configuration example, see *Multi-Router WAN Configuration* next.

Multi-Router WAN Configuration

Figure 46 is a diagram of a multi-router WAN configuration encompassing three subnets. From the diagram, you can see that R1 is part of both Subnets 1 and 2; R2 is part of both Subnets 2 and 3; and R3 is part of subnets 1 and 3. You can click on the router label (in blue) to jump to the actual text configuration file for that router:

Figure 46. Multi-Router WAN Configuration



Router R1 Configuration File

The following configuration file applies to Router R1.

```
------------------------------------------------------------------------
Configuration for ROUTER R1
------------------------------------------------------------------------
port set hs.7.1 wan-encapsulation frame-relay speed 45000000
port set hs.3.1 wan-encapsulation frame-relay speed 45000000
port set hs.3.2 wan-encapsulation ppp speed 45000000
port set et.1.* duplex full

frame-relay create vc port hs.7.1.106
frame-relay create vc port hs.3.1.103
frame-relay define service CIRforR1toR6 cir 45000000 bc 450000
frame-relay apply service CIRforR1toR6 ports hs.7.1.106

vlan create s1 id 200
vlan create s2 id 300
vlan add ports hs.3.1.103,hs.7.1.106 to s1
vlan add ports hs.3.2 to s2
interface create ip s1 address-netmask 100.100.100.1/16 vlan s1
interface create ip s2 address-netmask 120.120.120.1/16 vlan s2

rip add interface all
rip set interface all version 2
rip set interface all xmt-actual enable
rip set auto-summary enable
rip start

system set name R1
```

Router R2 Configuration File

The following configuration file applies to Router R2.

```
------------------------------------------------------------------------
Configuration for ROUTER R2
------------------------------------------------------------------------
port set hs.7.1 wan-encapsulation ppp speed 45000000
port set hs.7.2 wan-encapsulation ppp speed 45000000
port set et.1.* duplex full

vlan create s2 id 300
interface create ip PPPforR2toR3 address-netmask 130.130.130.2/16 peer-address 130.130.130.3 port hs.7.2
interface create ip SBitsLAN address-netmask 20.20.20.2/16 port et.1.1
vlan add ports hs.7.1 to s2
interface create ip s2 address-netmask 120.120.120.2/16 vlan s2
interface create ip VideoClient address-netmask 200.200.200.1/16 port et.1.2

qos set ip VideoFromNT high 100.100.100.100 200.200.200.200 any any
qos set ip VideoFrom95 high 200.200.200.200 100.100.100.100 any any

rip add interface all
```

```
rip set interface all version 2
rip set auto-summary enable
rip start

system set name R2
arp add 20.20.20.12 exit-port et.1.1 mac-addr 000202:020200
```

Router R3 Configuration File

The following configuration file applies to Router R3.

```
------------------------------------------------------------------------
Configuration for ROUTER R3
------------------------------------------------------------------------
port set se.2.1 wan-encapsulation frame-relay speed 1500000
port set et.1.* duplex full
port set hs.4.1 wan-encapsulation frame-relay speed 45000000
port set hs.4.2 wan-encapsulation ppp speed 45000000

frame-relay create vc port se.2.1.304
frame-relay create vc port hs.4.1.103

vlan create s1 id 200
interface create ip SBitsLAN address-netmask 30.30.30.3/16 port et.1.1
vlan add ports hs.4.1.103,se.2.1.304 to s1
interface create ip PPPforR2toR3 address-netmask 130.130.130.3/16 peer-address 130.130.130.2 port hs.4.2
interface create ip s1 address-netmask 100.100.100.3/16 vlan s1

rip add interface all
rip set interface all version 2
rip set interface all xmt-actual enable
rip set broadcast-state always
rip set auto-summary enable
rip start

system set name R3

arp add 30.30.30.13 exit-port et.1.1 mac-addr 000303:030300
```

Router R4 Configuration File

The following configuration file applies to Router R4.

```
--------------------------------------------------------------------------
Configuration for ROUTER R4
--------------------------------------------------------------------------
port set se.6.1 wan-encapsulation frame-relay speed 1500000
port set se.6.3 wan-encapsulation ppp speed 1500000
port set et.1.* duplex full

frame-relay create vc port se.6.1.304

vlan create s1 id 200
vlan add ports se.6.1.304,se.6.3 to s1
interface create ip s1 address-netmask 100.100.100.4/16 vlan s1

rip add interface all
rip set interface all version 2
rip set interface all xmt-actual enable
rip set broadcast-state always
rip set auto-summary enable
rip start

system set name R4
```

Router R5 Configuration File

The following configuration file applies to Router R5.

```
--------------------------------------------------------------------------
Configuration for ROUTER R5
--------------------------------------------------------------------------
port set se.4.1 wan-encapsulation ppp speed 1500000
port set et.1.* duplex full

vlan create s1 id 200

interface create ip lan1 address-netmask 50.50.50.5/16 port et.1.1
vlan add ports se.4.1 to s1
interface create ip s1 address-netmask 100.100.100.5/16 vlan s1

rip add interface all
rip set auto-summary enable
rip set interface all version 2
rip start

system set name R5

arp add 50.50.50.15 mac-addr 000505:050500 exit-port et.1.1
```

Router R6 Configuration File

The following configuration file applies to Router R6.

```
--------------------------------------------------------------------------
Configuration for ROUTER R6
--------------------------------------------------------------------------
port set et.15.* duplex full
port set hs.3.1 wan-encapsulation frame-relay speed 45000000

frame-relay create vc port hs.3.1.106
frame-relay define service CIRforR1toR6 cir 45000000 bc 450000
frame-relay apply service CIRforR1toR6 ports hs.3.1.106

vlan create BridgeforR1toR6 port-based id 106
interface create ip FRforR1toR6 address-netmask 100.100.100.6/16 vlan BridgeforR1toR6
interface create ip lan1 address-netmask 60.60.60.6/16 port et.15.1
vlan add ports hs.3.1.106 to BridgeforR1toR6
vlan add ports et.15.2 to BridgeforR1toR6

qos set ip VideoFromNT high 100.100.100.100 200.200.200.200 any any
qos set ip VideoFrom95 high 200.200.200.200 100.100.100.100 any any

rip add interface all
rip set interface all version 2
rip set auto-summary enable
rip start

system set name R6

arp add 60.60.60.16 mac-addr 000606:060600 exit-port et.15.1
```