GRF Configuration and Management - 1.4 Update

Ascend Communications, Inc.
Part Number: 7820-2030-001
For software version 1.4.6 and later

October, 1998

Ascend Customer Service

Ascend Customer Service provides a variety of options for obtaining technical assistance, information about Ascend products and services, and software upgrades.

Finding information and software on the Internet

Visit Ascend's Web site at `http://www.ascend.com` for technical information, product information, and descriptions of available services.

Visit Ascend's FTP site at `ftp.ascend.com` for software upgrades, release notes, and addenda to this manual.

Obtaining Technical Assistance

You can obtain technical assistance by telephone, email, fax, or modem, and over the Internet.

Enabling Ascend to assist you

If you need to contact Ascend for help with a problem, make sure that you have the following information when you call or that you include it in your correspondence:

- Product name and model.
- Software and hardware options.
- Software version.
- Service Profile Identifiers (SPIDs) associated with your product.
- Your local telephone company's switch type and operating mode, such as AT&T 5ESS Custom or Northern Telecom National ISDN-1.
- Whether you are routing or bridging with your Ascend product.
- Type of computer you are using.
- Description of the problem.

Calling Ascend from within the United States

In the U.S., you can take advantage of Priority Technical Assistance or an Ascend Advantage Pak service contract, or you can call to request assistance.

Priority Technical Assistance

If you need to talk to an engineer right away, call (900) 555-ASND (2763) to reach Ascend's Priority Call queue. The charge of $2.95 per minute does not begin to accrue until you are connected to an engineer. Average wait times are less than three minutes.

Ascend Advantage Pak

Ascend Advantage Pak is a one-year service contract that includes overnight advance replacement of failed products, technical support, software maintenance releases, and software update releases. For more information, call (800) ASCEND-4 (272-3634), or access Ascend's

Web site at `www.ascend.com` and select Services and Support, then Advantage Service Family.

Other telephone numbers

For a menu of Ascend's services, call (800) ASCEND-4 (272-3634). Or call (510) 769-6001 for an operator.

Calling Ascend from outside the United States

You can contact Ascend by telephone from outside the United States at one of the following numbers:

Telephone outside the United States	(510) 769-8027 or (800) 697-4772
Australia/Asia Pacific	(+61) 3 9656 7000 apac.support@ascend.com http://apac.ascend.com/contacts.html
Austria/Germany/Switzerland	(+33) 492 96 5672
Benelux	(+33) 492 96 5674
France	(+33) 492 96 5673
Italy	(+33) 492 96 5676
Japan	(+81) 3 5325 7397
Middle East/Africa	(+33) 492 96 5679
Scandinavia	(+33) 492 96 5677
Singapore	(+65) 738 7723
Spain/Portugal	(+33) 492 96 5675
UK	(+33) 492 96 5671

Obtaining assistance through correspondence

You can send questions to Ascend in one of the following ways:

- Email—support@ascend.com
- E-mail (in Asia Pacific)—apac.support@ascend.com
- E-mail (in Europe and the UK)—emeasupport@ascend.com
- Fax—(510) 814-2312
- Customer Support BBS (by modem)—(510) 814-2302
- Write to Ascend at the following address:

  Attn: Customer Service
  Ascend Communications, Inc.
  One Ascend Plaza
  1701 Harbor Bay Parkway
  Alameda, CA 94502-3002

Important safety instructions

The following safety instructions apply to the GRF router models GRF-4-AC, GRF-4-DC, GRF-16-AC, and GRF-16-DC except as noted:

1 Read and follow all warning notices and instructions marked on the product or included in the manual.

2 Do not attempt to service this product yourself, as opening or removing covers and/or components may expose you to dangerous high voltage points or other risks. Refer all servicing to qualified service personnel.

3 The maximum recommended ambient temperature for all GRF router models is 104˚ Fahrenheit (40˚ Celsius). Care should be given to allow sufficient air circulation or space between units when the GRF chassis is installed in a closed or multi-unit rack assembly because the operating ambient temperature of the rack environment might be greater than room ambient.

4 Slots and openings in the GRF cabinet are provided for ventilation. To ensure reliable operation of the product and to protect it from overheating, maintain a minimum of 4 inches clearance on the top and sides of the GRF 400 router, and a minimum of 6 inches on the top and sides of the GRF 1600 router.

5 Installation of the GRF 400 or 1600 in a rack without sufficient air flow can be unsafe.

6 If a GRF router is installed in a rack, the rack should safely support the combined weight of all equipment it supports.
- A fully loaded, redundant-power GRF 400 weighs 38.5 lbs (17.3 kg).
- A fully loaded, single-power GRF 400 weighs 32.5 lbs (14.6 kg).
- A four card, redundant-power GRF 1600 weighs 147 lbs (66.2 kg).
- A four card, single-power GRF 1600 weighs 127 lbs (57.2 kg).

7 The connections and equipment that supply power to GRF routers should be capable of operating safely with the maximum power requirements of the particular GRF model. In the event of a power overload, the supply circuits and supply wiring should not become hazardous.

8 Models with AC power inputs are intended to be used with a three-wire grounding type plug - a plug which has a grounding pin. This is a safety feature. Equipment grounding is vital to ensure safe operation. Do not defeat the purpose of the grounding type plug by modifying the plug or using an adapter.

9 Prior to installation, use an outlet tester or a voltmeter to check the AC receptacle for the presence of earth ground. If the receptacle is not properly grounded, the installation must not continue until a qualified electrician has corrected the problem. Similarly, in the case of DC input power, check the DC ground (s).

10 If a three-wire grounding type power source is not available, consult a qualified electrician to determine another method of grounding the equipment.

11 Models with DC power inputs must be connected to an earth ground through the terminal block Earth/Chassis Ground connectors. This is a safety feature. Equipment grounding is vital to ensure safe operation.

12 Install DC-equipped GRF 400 and 1600 routers only in restricted access areas in accordance with Articles 110-16, 110-17, and 110-18 of the National Electrical Code, ANSI/NFPA 70.

13 Do not allow anything to rest on the power cord and do not locate the product where persons will walk on the power cord.

14 Industry-standard cables are provided with this product. Special cables that may be required by the regulatory inspection authority for the installation site are the responsibility of the customer.

15 When installed in the final configuration, the product must comply with the applicable Safety Standards and regulatory requirements of the country in which it is installed. If necessary, consult with the appropriate regulatory agencies and inspection authorities to ensure compliance.

Wichtige Sicherheitshinweise

Die folgenden Sicherheitshinweise gelten für die GRF-Oberfräsenmodelle GRF-4AC, GRF-4-DC, GRF-16-AC und GRF-16-DC, außer wenn anderweitig angegeben:

1 Lesen und befolgen Sie alle am Produkt angebrachten und im Handbuch enthaltenen Warnhinweise und Anleitungen.

2 Versuchen Sie nicht, dieses Gerät selbst zu warten bzw. die Abdeckung zu öffnen oder Bauteile zu entfernen. Hochspannungsgefahr. Die Wartung muß durch qualifiziertes Fachpersonal ausgeführt werden.

3 Die empfohlene maximale Umgebungstemperatur für alle GRF-Oberfräsenmodelle liegt bei 40º C. Sorgen Sie für gute Belüftung bzw. ausreichenden Abstand zwischen einzelnen Geräten, wenn das GRF-Gehäuse in einem Einzel- oder Mehrfach-Einschubrahmen installiert werden soll, da die Betriebstemperatur in dem Einschubrahmen evtl. höher als die Raumtemperatur sein kann.

4 Schlitze und Öffnungen im GRF-Gehäuse dienen zur Belüftung. Um einen einwandfreien Betrieb des Produktes zu gewährleisten und um Überhitzung vorzubeugen, jeweils oben und an den Seiten der GRF-400-Oberfräse mindestens 10,16 cm und an der GRF-1600-Oberfräse mindesten 15,24 cm Freiraum vorsehen.

5 Bei unzureichender Belüftung ist die Installation eines GRF-400 oder 1600 in einem Einschubrahmen gefährlich.

6 Bei Installation einer GRF-Oberfräse in einem Einschubrahmen, muß dieser das Gesamtgewicht aller darin installierten Geräte sicher tragen können.

- Ein komplett bestückter Redundanzstrom-GRF-400 wiegt 17,3 kg.
- Ein komplett bestückter Einzelstrom-GRF-400 wiegt 14,9 kg.
- Ein mit vier Karten bestückter Redundanzstrom-GRF-1600 wiegt 66,2 kg.
- Ein mit vier Karten bestückter Einzelstrom-GRF-1600 wiegt 57,2 kg.

7 Die Adapter und Geräte, die die GRF-Oberfräsen mit Strom versorgen, sollten auch bei maximaler Stromanforderung des einzelnen GRF-Modells noch sicher laufen. Im Fall einer Stromüberlastung sollten die Versorgungskreise und kabel keine Gefahrenquelle darstellen.

8 Alle mit Netzeingängen versehenen Geräte müssen mit einem vorschriftsmäßigen Stecker bestückt sein. Der Stecker bietet die notwendige Erdung und darf in keiner Weise modifiziert oder mit einem Adapter verwendet werden.

9 Überprüfen Sie vor der Installation mit Hilfe eines Steckdosentestgerätes oder eines Voltmeters die Erdung der Netzsteckdose. Sollte die Steckdose nicht ordnungsgemäß geerdet sein, darf mit der Installation erst fortgefahren werden, wenn ein qualifizierter Elektriker dieses Problem behoben hat. Handelt es sich um einen Gleichstromeingang ist dieser in gleicher Weise auf ordnungsgemäße Erdung zu überprüfen.

10 Ist keine 3polige geerdete Stromquelle vorhanden, beauftragen Sie einen qualifizierten Elektriker damit, das Gerät auf andere Weise zu erden.

11 Bei Modellen mit Gleichstromeingängen muß ein Erdungsdraht entweder an der Klemmleiste oder an einer Gehäuseschraube angeschlossen werden. Hierbei handelt es sich um eine Sicherheitseinrichtung. Die Erdung des Gerätes ist eine wichtige Voraussetzung für den sicheren Betrieb.

12 Die gleichstromausgerüsteten Oberfräsenmodelle GRF-400- und GRF-1600-Oberfräse dürfen nur in Bereichen mit beschränktem Zugang, unter Berücksichtigung der anwendbaren Bestimmungen für Elektroinstallationen sowie der Standards ANSI/NFPA 70 installiert werden.

13 Keine Gegenstände auf das Netzkabel stellen. Das Kabel so verlegen, daß Personen nicht versehentlich darauf treten können.

14 Standardkabel sind im Lieferumfang des Produkts enthalten. Sonderkabel, die evtl. gemäß den örtlichen Bestimmungen für die Installation erforderlich sind, sind vom Kunden zu stellen.

15 Zur Installation in der endgültigen Konfiguration muß das Produkt den am Installationsort geltenden Sicherheitsstandards und bestimmungen entsprechen. Genauere Informationen erhalten Sie ggf. bei den zuständigen Behörden.

Contents

Chapter 3 Management Commands and Tools 3-1

Chapter 12 ATMP Configuration Guide 12-1

Chapter 15 Configuring Frame Relay 15-1

Figures

Tables

About This Guide

The *GRF Configuration and Management* manual provides information for configuring individual GRF system parameters, options, and services, and describes configuration options available for each type of media card.

Unless otherwise noted, the information in this guide applies to GRF 400 and 1600 systems as well as to GRF 400 and GR-II systems using an RMS node.

About the 1.4 Update

The 1.4 GRF manual set is updated to include new features added since software release 1.4.6. GateD information is now provided in a new document, the *GRF GateD Manual*.

How to use this guide

The *GRF Configuration Guide* contains these chapters:

- Chapter 1, “Working in the GRF User Interface,” describes the Command Line Interface (CLI) and UNIX shell environments available to users.
- Chapter 2, “Configuring System Parameters,” details configuration of basic system-level operating parameters and options.
- Chapter 3, “Management Commands and Tools” provides an overview of GRF administrative commands and usage information for diagnostic and data-gathering tools.
- Chapter 4, “Management Tasks” provides procedures for a range of GRF administrative and maintenance tasks.
- Chapter 5, “ATM OC-3c Configuration,” describes configuration options and monitoring/debug commands available on ATM/Q media cards.
- Chapter 6, “FDDI Configuration,” describes configuration options and monitoring/debug commands available on FDDI/Q media cards.
- Chapter 7, “HIPPI Configuration,” describes configuration options and monitoring/debug commands available on HIPPI media cards.
- Chapter 8, “HSSI Configuration,” describes configuration and protocol options and monitoring/debug commands available on HSSI media cards.
- Chapter 9, “Ethernet Configuration,” describes configuration options and monitoring/debug commands available on fast Ethernet media cards.

- Chapter 10, "SONET OC-3c Configuration," describes configuration and protocol options and monitoring/debug commands available on SONET OC-3c media cards.
- Chapter 11, "ATM OC-12c Configuration," describes configuration options and monitoring/debug commands available on ATM OC-12c media cards.
- Chapter 12, "ATMP Configuration Guide," explains the Ascend Tunnel Management Protocol implementation on HSSI and ATM media and provides configuration examples.
- Chapter 13, "Transparent Bridging," describes the features of GRF bridging and provides media card configuration information and examples.
- Chapter 14, "IP Packet Filtering," explains the application of packet filtering options.
- Chapter 15, "Configuring Frame Relay," describes the implementation of Frame Relay on HSSI and SONET media cards.
- Chapter 16, "Integrated Services:Controlled-Load," describes the initial GRF implementation of Integrated Services, the provision of Controlled-Load services on Ethernet, FDDI, SONET, and HSSI media cards.
- Appendix A, "Introduction to Subnetting," details the design of variable-length netmasks and subnets that support classless addressing.
- Appendix B, "Warranty," contains the product warranty information.
- Appendix C, "Configuration Tracking via CFMS," describes how to install and use the CFMS application. This enables the site to remotely monitor and manage GRF configurations.

This guide also includes an index.

Helpful things to know

Configuring and monitoring the GRF requires that a Network Administrator have experience with and an understanding of UNIX systems, and the ability to navigate in a UNIX environment. Knowledge of UNIX, its tools, utilities, and editors is useful, as is experience with administering and maintaining a UNIX system.

Configuring the GRF requires network experience and familiarity with:

- UNIX systems and commands
- IP protocol and routing operations
- IP internetworking

The Network Administrator must understand how TCP/IP internetworks are assembled; what interconnections represent legal topologies; how networks, hosts, and routers are assigned IP addresses and configured into operation; and how to determine and specify route table (routing) information about the constructed internetwork(s). Although not required, a high-level understanding of SNMP is useful.

Manual set

The GRF 1.4 Update documentation set consists of the following manuals:

- *GRF 400/1600 Getting Started - 1.4 Update*
- *GRF Configuration and Management - 1.4 Update* (this manual)
- *GRF Reference Guide - 1.4 Update*
- *GRF GateD Manual*

Documentation conventions

Ascend uses standard documentation conventions, which are as follows:

Convention	**Meaning**
`Monospace text`	Represents text that appears on your computer's screen, or that could appear on your computer's screen.
`Boldface monospace text`	Represents characters that you enter exactly as shown (unless the characters are also in ***`italics`***—see *Italics*, below). If you could enter the characters but are not specifically instructed to, they do not appear in boldface.
Italics	Represent variable information. Do not enter the words themselves in the command. Enter the information they represent. In ordinary text, italics are used for titles of publications, for some terms that would otherwise be in quotation marks, and to show emphasis.
[ ]	Square brackets indicate an optional argument you might add to a command. To include such an argument, type only the information inside the brackets. Do not type the brackets unless they appear in bold type.
|	Separates command choices that are mutually exclusive.
Key1-Key2	Represents a combination keystroke. To enter a combination keystroke, press the first key and hold it down while you press one or more other keys. Release all the keys at the same time. (For example, Ctrl-H means hold down the Control key and press the H key.)
Press Enter	Means press the Enter, or Return, key or its equivalent on your computer.
Note:	Introduces important additional information.
Caution:	Warns that a failure to follow the recommended procedure could result in loss of data or damage to equipment.
Warning:	Warns that a failure to take appropriate safety precautions could result in physical injury.

Related publications

Here are some related publications that you may find useful:

- *Internetworking with TCP/IP*, Volume 1 and 2, by Douglas E. Comer, and David L. Stevens. Prentice-Hall,
- *TCP/IP Illustrated*, Volumes 1 and 2, by W. Richard Stevens. Addison-Wesley, 1994.
- *Interconnections*, Radia Perlman. Addison-Wesley, 1992. Recommended for information about routers and bridging.
- *Routing in the Internet*, by Christian Huitema. Prentice Hall PTR, 1995. Recommended for information about IP, OSPF, CIDR, IP multicast, and mobile IP.
- *TCP/IP Network Administration*, by Craig Hunt. O'Reilly & Associates, Inc. 1994. Recommended for network management information.
- *Essential System Administration*, Æleen Frisch. O'Reilly & Associates, Inc. 1991. Recommended for network management information.

Working in the GRF User Interface

1

Chapter 1 describes the three interface components you use when working with the GRF:

– a UNIX shell is available, with standard UNIX commands and some GRF-UNIX commands modified for the GRF

– a Command Line Interface (CLI), with TNT-like profiles for system, card, load files, and dump parameters. The UNIX shell is created with the CLI **sh** command, and many of the UNIX and GRF-UNIX commands used frequently in GRF manuals can be entered at the CLI prompt, currently there are more than 80.

– a set of **maint** commands that report low-level media card statistics

The *GRF Reference Guide* contains descriptions and usage information for each CLI and GRF-UNIX command.

Chapter 1 includes these topics:

Overview of GRF user interface components

This overview is intended to help you understand what is happening in the procedures that make up the *GRF Configuration and Managemen*t manual. The procedures use both the Command Line Interface and the UNIX shell. As you configure and manage the GRF, you will use CLI commands along with UNIX commands to specify parameters, monitor the interfaces, and edit configuration files.

The GRF user environment consists of two main components, the Command Line Interface (CL), and the UNIX shell. There is also a third component, a set of low-level **maint** commands. It is not possible to nest CLI and shell sessions.

Command Line Interface (CLI)
The Command Line Interface, usually called CLI, supports a set of profiles and a large set of GRF and UNIX-like commands. CLI profiles replace the `/etc/grinchd.conf` file. Variables from `/etc/grinchd.conf` are now fields in Card, User, Dump, Load, and System profiles. You use the command-line interface (CLI) to access the profiles and assign values to the fields they contain. Data is stored in `/etc/prof` files.

There are five profiles that contain parameters for five areas:

- System profile, includes IP address, host name, hardware characteristics
- Card profile, includes media type, protocol, port parameters, ICMP settings
- User profile(s), a profile for each defined user, includes access, password
- Dump profile, defines dump events and storage
- Load profile, defines which are the running binaries

Profiles are described in detail in this chapter.

The GRF UNIX-like system management and configuration commands are available in the CLI. At the CLI prompt (usually `super>`) enter a "?" to retrieve the list of available commands. Refer to the *GRF Reference Guide* for a description of each CLI command.

When you log in to a GRF as `root`, you automatically get the CLI shell. In the CLI, `root` is super user, hence the `super>` prompt.

UNIX shell
While in the CLI you use the **sh** command to open a UNIX shell. The shell supports standard UNIX commands and the GRF UNIX-like commands. You can manage the GRF using the UNIX network and configuration management commands. The GRF also has a number of configuration files that you edit with a UNIX editor, you must be in the shell to edit GRF configuration files.

To configure and manage the GRF, you will use both the CLI command set and the UNIX shell. Type **exit** to leave the shell. When you exit the UNIX shell, you can execute CLI-only commands.

maint commands
A third component is the set of **maint** commands specific to each media card. The **maint** commands display low-level card-specific statistics and counts. These commands are documented within each media card's configuration information in the*GRF Configuration and Management* manual. Ethernet **maint** commands are in the Ethernet chapter, for example.

CLI command list

To see a list of CLI commands, type:

```
super> ?
```

or

```
super> help
```

You see the list of supported commands and permission levels.

If a permission is not enabled in your User profile, you will not be able to see or use the commands at that level. By default, every user can execute the "`user`" commands. `system` and `update` commands require higher-level permissions. If you have logged in using the Default profile, these are the commands available to you: **?**, **auth**, **clear**, **exit**, **help**, **quit**, **sh**, and **whoami**.

This manual describes logging in as `root` because it automatically puts you in the Super profile and you are pre-assigned `system` and `update` command permissions.

Note that the CLI includes GRF-specific commands such as **grarp** (ARP) and **gratm** (ATM) as well as commands from UNIX such as **netstat** and **traceroute**. For details about each command, see the *GRF Reference Guide*.

Table 1-1. CLI command list and descriptions

Command name	Permission level	Description
?	user	Help, or its alias, ?, returns a list of all registered commands authorized by user's security profile. Provides description/usage when followed by a specific command.
auth	user	Changes permissions levels in a User profile.
biosver	system	Prints the BIOS version installed on a GRF system.
bredit	system	Accesses and edits `/etc/bridged.conf` configuration file.
brinfo	user	Displays bridging port information.
brstat	system	Returns **bridged** state information.
cd	user	Lists the fields of the current profile. (same as list command)
clear	user	Clears screen, moves prompt to top line of screen.
date	update	Returns current time and date.
delete	update	Deletes a profile or field member.
dir	user	Returns a list of the main-line profiles.
exit	user	Exits a user from the command-line interface (CLI).

Table 1-1. CLI command list and descriptions (continued)

Command name	Permission level	Description
fastboot	system	Reboots or halts system without checking disks.
flashcmd	system	Copies a file or directory to or from a flash device.
gdc	system	Monitors and manages the **gated** routing daemon.
get	user	Displays contents of a profile read into local memory, same as ls.
getver	system	Reports version number of current software or next one to boot.
grarp	system	Manages Internet-to-physical address resolution tables (ARP).
grass	system	Manages services linked to internal operating system port.
gratm	system	Manages ATM configuration parameters.
grbist	system	Establishes card connection for field-run diagnostics.
grburn	system	Reprograms ATM, FDDI, and HIPPI flash, see grflash.
grc	system	RMS node system only - archives config, system files.
grcard	system	Displays type and status of media card in each card slot.
grclean	system	Program that compresses and manages dumps and logs.
grconslog	system	Accesses the system console log, `gr.console.`
grdebug	system	Provides options to monitor system functionality.
grdump	system	Captures memory dump images.
gredit	system	Opens `filterd.conf`, `gated.conf` in UNIX vi editor.
grfddi	system	Sets SAS and DAS attachments on FDDI media cards.
grfins	system	Installs software on GRF systems with the new control board.
grflash	system	Reprograms HSSI, SONET, CDDI, Ethernet flash, see grburn
grinch	system	Assigns, displays **grinch** configuration variables and values.
grlamap	system	Assigns logical addresses to HIPPI media card interfaces.
grmaint	system	Sends a hand-coded maintenance packet to a port.
grmem	system	Accesses media card memory for debug purposes.
grmrflash	system	Dangerous - completely initializes flash memory prior to an install. Use only under direction of Support staff.

Table 1-1. CLI command list and descriptions (continued)

Command name	**Permission level**	**Description**
grms	system	Halts, reboots, or shuts down operating system from local login.
grppp	system	Configures point-to-Point Protocol on HSSI and SONET.
grreset	system	Resets media cards, performs media card dumps.
grrmb	system	Switches to the GR#> prompt for maintenance commands.
grroute	system	Initializes and manipulates static routes on those systems not running dynamic routing / GateD.
grrt	system	Displays media card route table, can also configure a route, but this method is not recommended.
grsavecore	system	Copy and format GRF kernel panic information.
grsite	system	Manages custom or other special operating system or media software files on GRF systems with the new control board.
grsnapshot	system	Archives configuration files or flash device image for storage on other flash memory on GRF systems with the new control board.
grwrite	system	Loads configuration and other files from system RAM to flash memory on GRF systems with the new control board.
gsm	system	
help	user	Help, or its alias, ?, returns a list of all registered commands authorized by user's security profile. Provides description/usage when followed by a specific command.
ifconfig	system	Assigns addresses, mask, other values to a logical interface.
iflash	system	Initializes (formats) specified flash memory.
list	user	Lists the fields of the current profile. (same as cd command)
load	update	Restores (loads) previous configuration script into current use.
ls	user	Displays contents of the current working profile. (same as get)
man	system	Returns a man page for the specified command.
mem	user	Displays amount of control board RAM, max is 512 (bytes).
mountf	system	Mounts a flash device as an available file system on GRF systems with the new control board.
netstat	system	Displays interface routing, protocol, and connection statistics.

Table 1-1. CLI command list and descriptions (continued)

Command name	Permission level	Description
new	system	Creates a new instance of a profile or field member.
ping	system	Sends, receives ICMP/IP echo, reply packets to/from interfaces.
pwd	user	Shows current user location in the profile tree (context), same as whereami.
quit	user	Terminates current CLI session. If the session originated from a remote device, an associated connection is terminated (telnet or modem connection). If the session is on a local console and system-wide authentication is in use, the login prompt is issued.
read	user	Reads a profile into local memory for user view, access.
route	system	Adds or deletes static routes if dynamic routing is not running.
save	update	Saves configuration information in `/etc/prof` directory.
set	system	Sets a field value or returns help text about a field, needs write.
setver	system	Specifies version of software to run after the next system boot.
sh	user	Creates a UNIX shell in the CLI.
shutdown	system	Halts, reboots, shuts down operating system from remote login.
traceroute	system	Prints the packet route to a specified destination host/network.
umountf	system	Unmounts a flash device on GRF systems with the new control board
vpurge	system	Removes a specified release or configuration version from a flash device on a GRF system with the new control board.
whatami	system	Tells if system is RMS node (`irms`) or control board (`cb`) based.
whereami	user	Tells user level of CLI or profile, same as pwd.
whoami	user	Returns the user profile name associated with this session
write	update	Permanently saves contents of a profile.

On-line help options

To obtain on-line information about a specific command, enter one of these commands:

```
super> ? <command-name>
super> help <command-name>
```

To obtain on-line information about a profile field, use:

```
super> set <field-name> ?
```

CLI prompts

At the first log on, whether by serial or telnet connection, you see the initial CLI prompt:

```
super>
```

Use **dir user** to see the set of standard User profiles:

```
super> dir user
   103  09/28/98 13:54:18  admin
    92  09/28/98 13:54:18  default
   106  09/28/98 20:54:18  super
```

View the fields and their default values in User super:

```
super> read user super
USER/super read
super> list
name* = super
password = Ascend
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace}}
active-enabled = yes
allow-system = yes
allo-update = yes
allow-password = yes
allow-debug = yes
prompt = *
log-display-level = none
```

If you have not changed the standard password, you see the password you have used, `password = Ascend`.

If you have changed the password, you will see that new password in the `password =` field. In the `prompt =` field, the * represents the name or index of this specific User profile. In User super, the user's prompt will be `super>` unless you change the * setting.

View the fields and their default values in User default. This is the profile you use to create new User profiles for site users, and to set their password and permissions:

```
super> read user default
USER/default read
super> list user default
name* = default
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace}}
```

```
active-enabled = yes
allow-system = no
allow-update = no
allow-password = no
allow-debug = no
prompt = GRF=>
log-display-level = none
```

The value in the `prompt` field is GRF. When you modify the standard User profiles or create profiles (accounts) for site users, you can specify unique prompts. Because prompts can be unique, examples in this manual use the `super>` throughout.

Line-editing commands

Table 1-2 lists the CLI line-editing commands. An arrow key indicates arrow keys on your keyboard or VT-100 arrow key escape sequences.

Table 1-2. CLI line-editing commands

Control Sequence	Effect
Control-H, DEL	Erase the previous character. Moves the cursor one position to the left, erasing the character at that position. Has no effect if placed at the beginning of a line and does not erase any prompt (beginning of line is to the right of the prompt).
Control-W	Erase the previous (space-delimited) word. A word is delineated by a space character. Characters to the right of the cursor, if any, are shifted left over the erased word. Has no effect if placed at the beginning of a line.
Control-U	Erase entire line, but not the prompt. The cursor is placed immediately to the right of the prompt.
Control-K	Erase the line to the right of the cursor position.
Control-C	Use Control-C only to terminate paged output. Other uses can have unexpected results.
Control-M or Control-J	Terminate the line. A carriage-return and line-feed are written to the output device, but are not stored in the buffer returned to caller. The input is added to the end of the input history buffer.
Control-P or ↑	Replace the current line with the previous line from the line history. The current position within the history is kept so subsequent ^P on the same line will select earlier lines from the history. The command is ignored when the current line is the oldest line of the history.
Control-N or ↓	Replace the current line with the next line from the line history. This sequence is valid only if Control-P or ↑ was used to select a previous line. The command is ignored when the current line is the beyond the end of line history.
Control-B or ←	Back up cursor to the previous position without deleting that character. If you now type non-control characters, they are inserted in the line (ignored at the beginning of a line).
Control-F or →	Move the cursor one character to the right, unless at the end of a line, in which case the character is ignored.

Using the command-history buffer

The command history buffer contains the last 10 command lines. If the buffer is full, the oldest command line is deleted when you enter a new command.

To repeat the previous line or to redisplay it so you can modify it, press the up-arrow key, or Control-P. Modify and then press Enter to execute the new command. The cursor can be positioned anywhere within the command line when you press Enter.

To replace the current line with the next line from the line history, press the down-arrow key, or Control-N. This sequence is valid only if Control-P or ↑ was used to select a previous line. The command is ignored when the current line is the beyond the end of line history.

Command-line shortcuts

Use the period (.)

You can use a period (.) to substitute for the last profile name or field name you entered. In this example with the **get** and the **list** commands, a period represents "card 8":

```
super> read card 8
CARD/8 read
super> get .
card-num* = 8
media-type = atm-oc3-v2
    •
    •
    •
super> read card 8
CARD/8 read
super> list .
card-num* = 8
media-type = atm-oc3-v2
```

Abbreviate field names

While you cannot use abbreviated profile names such as "user s" for User super, you can specify a field name by typing enough characters to specify a unique string. The CLI automatically fills in the rest of the name.

To examine the `icmp-throttling` field from the Card 8 profile above, enter: **get . icmp**

```
super> read card 8
CARD/8 read
super> get . icmp
echo-reply = 10
unreachable = 10
    •
    •
    •
```

A single letter works if it is a unique string:

```
super> read card 8
CARD/8 read
super> get . i
echo-reply = 10
unreachable = 10
    •
    •
    •
```

In the Card profile, two fields begin with "c", `card-num*` and `config`. If you do not specify a unique string, you get this message:

```
super> get . c
error: field name "c" ambiguous
```

All other control characters

Except for the control character usage described in Table 1-2, control characters (Control-D, Control-C) are not used on the command line.

Printable characters

Printable characters are inserted at the current cursor position. Characters to the right of the cursor are shifted to the right within the input buffer. If character insertion would cause the buffer size to be exceeded, the rightmost character in the buffer is removed, truncating the input to the buffer size.

Use of asterisks

- In a profile, an asterisk following a field name tells you that field contains the profile's index.

  For example, this entry in a User profile tells you it is the User default profile:
  `name* = default`

- In a profile, an asterisk following the = sign tells you that field will use the profile's index as its assigned value.

  For example, this entry in a User profile tells you the prompt will use the value assigned as the index:
  `prompt = *`

Paged line output

Paged output breaks up multiple lines of output at screen boundaries and lets the user press a key to get more information or cancel the listing.

At page boundaries, the following prompt is displayed:

```
stdin
```

The user has the option of entering:

- a space to display the next page of data
  (causes current data to scroll off the top of screen)
- a return to display one more line
- a Control- C to cancel paged output (“`Stopped`” is displayed)

Introduction to profiles

The command-line interface (CLI) supports five types of profiles.

In the CLI, profiles are referenced (called) by their profile type – Card, User, Dump, Load, System. Some profile types are one-of-a-kind, and are referenced just by their type name.

If more than one exists of a certain type of profile, it must be referenced by an index. An index specifies a particular profile in a group of the same type. In the case of User admin, User is the profile type and admin is the profile name or index. The index used for CLI access to a profile is identical to the index used for SNMP access to the profile.

Each of the five profile types is identified by a unique type name:

Card

– describes a media card in a specified slot

– multiple Card profiles, referenced by slot number indexes 0–15 or 1–4, depending upon the number of slots in the router chassis: `read card 6`

Dump

– specifies system dump parameters

– only one Dump profile, referenced by its name: `read dump`

Load

– names the run-time binary code for each media card type

– only one Load profile, referenced by its name: `read load`

System

– contains system information

– only one System profile, referenced by its name: `read system`

User

– provides a user account with security, access privileges, and permissions

– multiple User profiles, referenced by the user name: `read user bob`

Profile fields

The parameters listed in a profile are called fields. Fields are expressed in many forms: numeric, text, IP address, boolean, enumerated, or hexadecimal. A field name is unique within the profile, but the name can appear in and be shared among different profiles. When you reference a field name, case is ignored and you need to enter only enough leading characters to uniquely identify the name.

The User profile has 10 fields on its top or main level:

```
name* = bob
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace}}
active-enabled = yes
allow-system = yes
allow-update = yes
allow-password = yes
allow-debug = yes
prompt = *
log-display-level = none
```

Fields can contain their own set of fields. Such a field is called a complex structure. Fields that contain a subset of fields are referenced by the names of those fields.

In the User profile, the `auth-method` field is the only complex structure. Here are the three fields listed within `auth-method`:

```
super> list auth-method
auth-type = PASSWD
rad-auth-client = { "" 1645 udp "" }
securid-auth-client = { 5500 udp 5510 tcp /var/ace }
```

Here is how the subset of fields in `auth-method` field is referenced, abbreviations are used:

```
super> list auth-type
auth-type = PASSWD

super> cd ..
auth-type = PASSWD
rad-auth-client = { "" 1645 udp "" }
securid-auth-client = { 5500 udp 5510 tcp /var/ace }

super> list rad
auth-server = ""
auth-port = 1645
auth-protocol = udp
auth-key = ""

super> cd ..
auth-type = PASSWD
rad-auth-client = { "" 1645 udp "" }
securid-auth-client = { 5500 udp 5510 tcp /var/ace }
```

```
super> list sec
auth-port = 5500
auth-protocol = udp
auth-slave-port = 5510
auth-slave-protocol = tcp
auth-server-conf-path = /var/ace
super>
```

Complex structure

A field that is a complex structure contain curly brackets {}. The `hw-table` and `dump-vector-table` fields in the Dump profile are complex structures:

```
super> read dump
DUMP read

super> get dump
hw-table = <{hippi 20 /var/portcards/grdump 0}{rmb 20 /var/portcards/gr+
dump-vector-table = <{0 no-media "" < > } { 3 rmb "RMB default dump vect+
config-spontaneous = off
keep-count = 2

super> list hw
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }
```

Inside the curly brackets of a complex structure are the contents of each field separated by a space. If a field is a list of complex structures, it is delimited by angle brackets < > (when the list is not empty).

Inside the angle brackets are the contents of each complex structure in the list delimited by curly brackets { }. If a list field is empty, a pair of quotes `""` appear in place of the field contents.

When the contents of a field are longer than the maximum length of the line, the contents are truncated to fit on the line and a + (plus) is appended to indicate there is more data.
Look at the ports = line in this example:

```
super> read card 4
CARD/4 read

super> list
card-num* = 4
media-type = sonet-v1
debug-level = 0
hssi-frame-protocol = Frame-Relay
```

```
sonet-frame-protocol = Cisco-HDLC
ether-verbose = 0
ports =<{0 {off on 10 3} {single off} {"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

To read beyond the +, use the **list** or **cd** command on the field:

```
super> list ports 3
port_num = 3
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 207 }
hssi = { 0 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disab+
super>
```

Card profile components

Figure 1-1 shows the fields in each level of the Card profile. The fields are described on the pages following.

Figure 1-1. Diagram of Card profile levels

Card profile field descriptions

1st-level fields

Here are the fields at the first level.

`card-num* = 8`

- read only, chassis slot number (0-3, GRF 400) (0-15, GRF 1600)

`media-type = fddi`

- read only, names are specified the same as in Load profile list

`debug-level = 0`

- 0 or non-zero, 1 = lower level of debugging

`hssi-frame-protocol =`

- cisco-hdlc, ppp, frame-relay (default)

`sonet-frame-protocol =`

- ppp (default), frame-relay

`ether-verbose = 0`

- specifies flow of card event messages, 1..9 = more messages

`ports =`

- contains fields for individual physical port configuration

`load =`

- holds configuration parameters for the card's own load procedure, you can specify a custom binary to be loaded at boot in this section

`dump = 0`

- holds configuration parameters for the card's own dump procedure, you can specify custom dump requirements in this section

`icmp-throttling =`

- contains fields for ICMP configuration options

2nd-level fields

Here are the fields at the second level, defaults are shown.

load:

Settings in its fields are card-specific, override system-wide settings in Load profile.

`config = 0`

- field not available

`hw-table = < >`

- empty field, use to specify name of special executable to load

`boot-seq-index = 1`

- 0 or non-zero, current index into boot sequence

`boot-seq-state = 0`

- 0 or non-zero, current state of binary running `boot-seq-diagcode = 0`

- 0 or non-zero, exit code of last binary executed

config:

Settings in its fields define an alternative boot binary.

`word = 0`

- 0 or non-zero, use default

`ping = 1`

- 0 or non-zero, use default

`reset = 1`

- 0 or non-zero, use default

`init = 4`

- 0 or non-zero, use default

`panic-reset = 0`

- 0 or non-zero, use default

`speed-tx-thresh = 0`

- 0 to 100%, the transmit threshold % set for selective packet discard

dump:

Settings in its fields are card-specific and override the system-wide settings in the Dump profile for a particular card.

`config = 0`

Settings are:
0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The `config =` value is the sum of any number of settings, expressed in hex. You may sum multiple settings, but you must always use hex.

To dump during panic, card reset, and power up, sum 0004, 0008, and 0020 to obtain 20.

`hw-table = < >`

- empty field, specify name for special dump file

`config-spontaneous = off`

- off or on, use default

`dump-on-boot = off`

- off or on, enables or disables automatic dump at each card boot

icmp-throttling:

Fields specify how fast different ICMP messages are generated from media cards, a setting of 0 disables. Specified in number per one-tenth second.

`echo-reply = 10`

- number of replies to echo requests

`unreachable = 10`

- number of “cannot deliver packet” replies

`redirect = 2147483647`

- redirect messages are not limited

`TTL-timeout = 10`

- number of time-to-live replies

`param-problem = 10`

- number of parameter problem (packet discard) messages

`time-stamp-reply = 10`

- number of time of day time stamp replies

3rd-level port fields

Here are the port fields at the third level, defaults are shown.

cisco-hdlc:

`debug = off`

- Cisco HDLC debug off (disabled) or on (enabled)

`keepalive-enabled = on`

- Keepalive messages set off (disabled) or on (enabled)

`keepalive-interval = 10`

- Number of milliseconds before next keepalive message is sent

`keepalive-error-thresh = 3`

- Number messages received before marking link down

fddi:

`single-dual = single`

- single (SAS) or dual (DAS) connection/port

`optical-bypass = off`

- off (disabled) or on (enabled)

sonet:

`path_trace_message = ""` - not in use

`section_trace_message = ""` - not in use

`aps-command = 1`

Specifies the APS command value, 1 through 6:

1 - clear out all other switch commands, default is 1, use before changing a setting,
2 - do not allow a protection channel
3 - forced switch of working, overrides hardware switch
4 - forced switch of protection, overrides hardware switch
5 - manually switch the working channel
6 - manually switch the protection channel

`mode = sonet`

- set media mode to SONET (sonet) or SDH (sdh)

`clock = internal-oscillator`

- set to internal-oscillator or to recovered-clock

`aps-status = 0`

- returns a code that reflects the actual state of the active line

`payload-identifier = 207`

- specifies SONET payload-identifier, numeric value ranges from 0 to 255, default is 207

hssi:

`source-clock = 0`

- set to 1 if using null modem cable, set to 0 if not

`CRC-type = 16-bit`

- either 0, 16-bit (Frame Relay, PPP), or 32-bit (HDLC)

ether:

`if-config = autonegotiate`

Use to specify Ethernet transfer rate/connection mode:

autonegotiate - autonegotiate, default
10-half - 10 BaseT half duplex
10-full - 10 BaseT full duplex
100-half - 100 BaseT half duplex
100-full - 100 BaseT full duplex

hippi:

`debug-level = 1`

- 0–3, number of messages sent to logger

`ready-count = 32`

- 1–63, HIPPI ready count

`testmode = no-mode`

Settings for test mode:

no-mode: no test running, default
hippi-source: sourcing HIPPI data
loopback: loopback, a single board mimics a cable
switch-test: switch test
agency: agency test mode
abort: test aborted HIPPI connection
ip-packet: spit out one IP packet over HIPPI
immunity: like agency but with error checking

`test-duration = 999999`

- test duration in seconds, 0 or non-zero

`test-pkt-size = 4`

- size of test packet in HIPPI bursts

`test-pattern = incremental`

Sets HIPPI test pattern, options are:

alt-walking - alternates walking 1 bit and walking 0 bits
all-ones - all 1 bits

repeat - repeat a pattern of 00000000 01010101 02020202 03030303
incremental - incremental pattern of 01010101 02020202 to ffffffff
alternate - alternate buffers of random pattern and aaaaaaaa/55555555

`test-check-data = 5`

Sets rate of test packets to be verified, every nth packet, 1 – 10

`max-Ifields = 300`

- currently not used

`out-timeout = 10`

- sets number of tenths of a second until output time-out, 0 or non-zero

`switch-timeout = 10`

- sets number of tenths of a second until switch time-out, 0 or non-zero

`IBM-HO-mode = disabled`

- enables/disables IBM H0 mode

`hold-connection = disabled`

- enables/disables HIPPI hold connection

When disabled, a new connection per IP packet is needed. When enabled, a connection is held until an error occurs.

`out-timeout-mode = 0`

- settings are 0 or 1:

0 = default time-out checks for output buffer fed to FIFO,

1 = default check for non-decreasing number of buffers queued for output to FIFO

`disable-raw = disabled`

- enables/disables HIPPI raw mode transfers, if disabled, only IP mode is valid

`mcast-addr = 03:00:0f:e0`

- sets switch address of the HIPPI multicast server, this is the I-field HIPPI uses to send multicast packets, a 4-byte hex field

`tunnel-table =`

- option not supported

`local-sw-addr = 00`

- sets HIPPI switch address when utilizing a HIPPI ARP server, a 1-byte hex field

`arp-addr = 00`

- HIPPI switch address of the ARP server, 1-byte hex field

Dump profile components

Figure 1-2 and Figure 1-3 show the `hw-table` and `dump-vector-table` fields in the Dump profile. The fields are described on the pages following.

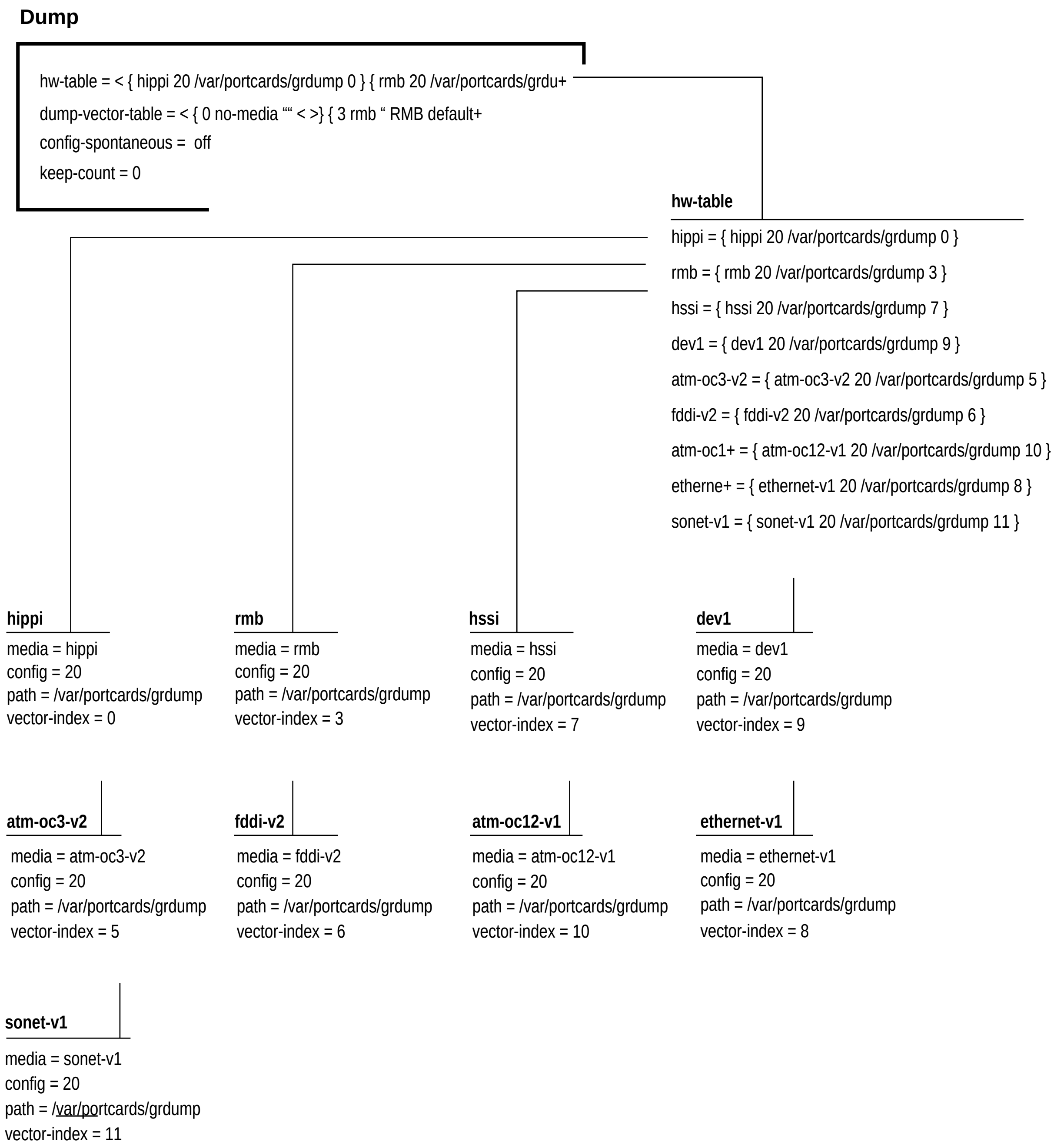


Figure 1-2. Dump profile: hw-table fields

dump-vector-table (all are read-only)

3 = { 3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = { 5 atm-oc3-v2 "ATM/Q default dump vectors" < { 1 "atm inst memory" 1677721+
6 = { 6 fddi-v2 "FDDI/Q default dump vectors" < { 1 "fddi/Q CPU0 core memory" 2+
7 = { 7 hssi "HSSI default dump vectors" < { 1 "hssi rx SRAM memory" 2097152 41+
8 = { 8 ethernet-v1 "ETHERNET default dump vectors" < { 1 "Ethernet rx SRAM mem+
9 = { 9 dev1 "DEV1 default dump vectors" < { 1 "dev1 rx SRAM memory" 2097152 20+
10 = { 10 atm-oc12-v1 "ATM OC-12 default dump vectors" < { 1 "ATM-12 SDRAM memo+
11 = { 11 sonet-v1 "SONET default dump vectors" < { 1 "SONET rx SRAM memory" 20+

dump-vector-table 3

index = 3
hw-type = rmb
description = "RMB default dump vectors"
segment-table = < { 1 SRAM 262144 524288 } >

segment-table

1 = { 1 SRAM 262144 524288 }

dump-vector-table 5

index = 5
hw-type = atm-oc3-v2
description = "ATM/Q default dump vectors"
segment-table = < { 1 "atm inst memory" 167772 . . . +

segment-table

1 = { 1 "atm inst memory" 16777216 4194304 }
2 = { 2 "SAR0-TX control memory" 50462720 131072 }
3 = { 3 "SAR0-RX control memory" 50593792 131072 }
4 = { 4 "SAR1-TX control memory" 50724864 131072 }
5 = { 5 "SAR1-RX control memory" 50855936 131072 }
6 = { 6 "dual port memory" 33554432 32768 }
7 = { 7 "shared memory" 50331648 131072 }

dump-vector-table 6

index = 6
hw-type = fddi-v2
description = "FDDI/Q default dump vectors"
segment-table = < { 1 "fddi/Q CPU0 core memory" . . . +

segment-table

1 = { 1 "fddi/Q CPU0 core memory" 2097152 2097152 }
2 = { 2 "fddi/Q IPC memory" 8388608 32800 }
3 = { 3 "fddi/Q shared descriptor memory" 16777216 131072 }
4 = { 4 "fddi/Q CPU1 core memory" 4194304 2097152 }

dump-vector-table 7

index = 7
hw-type = hssi
description = "HSSI default dump vectors"
segment-table = < { 1 "hssi rx SRAM memory" . . .+

segment-table

1 = { 1 "hssi rx SRAM memory" 2097152 4194304 }
2 = { 2 "hssi shared SRAM memory" 131072 32768 }
3 = { 3 "hssi tx SRAM memory" 69206016 2097152 }

dump-vector-table 8

index = 8
hw-type = ethernet-v1
description = "ETHERNET default dump vectors"
segment-table = < { 1 "Ethernet rx SRAM memory" . . . +

segment-table

1 = { 1 "Ethernet rx SRAM memory" 2097152 4194304 }
2 = { 2 "Ethernet shared SRAM memory" 131072 32768 }
3 = { 3 "Ethernet tx SRAM memory" 69206016 2097152 }

dump-vector-table 9

index = 9
hw-type = dev1
description = "DEV1 default dump vectors"
segment-table = < { 1 "dev1 rx SRAM memory" . . . +

segment-table

1 = { 1 "dev1 rx SRAM memory" 2097152 2097152 }
2 = { 2 "dev1 shared SRAM memory" 131072 32768 }

dump-vector-table 10

index = 10
hw-type = atm-oc12-v1
description = "ATM OC-12 default dump vectors"
segment-table = < { 1 "ATM-12 SDRAM memory" 167772 . . . +

segment-table

1 = { 1 "ATM-12 SDRAM memory" 16777216 4096 }
2 = { 2 "ATM-12 SSRAM memory" 25165824 1048576 }
3 = { 3 "SUNI Registers" 6291456 1024 }
4 = { 4 "Tx CTRL Regs" 33554432 16 }
5 = { 5 "Tx SAR Regs" 34603008 128 }
6 = { 6 "Tx DESC Memory" 35651584 131072 }
7 = { 7 "Rx CTRL Regs" 50331648 16 }
8 = { 8 "Rx SAR Regs" 34603008 128 }
9 = { 9 "Rx DESC Memory" 52428800 131072 }

Figure 1-3. Dump profile: dump vector tables

Dump profile field descriptions

Dump profile fields set system-wide values not usually changed. To change values on a specific card, change settings in the `dump` field in the Card profile.

1st-level fields

The `hw-table` and `dump-vector-table` fields at the first level are complex structures.

`hw-table = < {hippi 20 /var/portcards/grdump 0} {rmb 20 /var/portcards/grdu+`

`dump-vector-table = <{3 rmb "RMB default dump vectors" <{1 SRAM 262144 5242+`

`config-spontaneous = off,` set to off or on, use default

`keep-count = 2,` sets number of dumps to keep plus the current and the first of the day, set to 0 or non-zero, default 2 saves 2 additional dumps daily

2nd-level fields

Except as noted, the `hw-table` fields are common across cards.

`media =` - specific hardware type

`config =` - dump configuration settings are:

```
0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up
```

The default (20) dumps at card panics and when cards hang.

`path =` - file location of dump for this hardware type

`vector-index =` - the index into internal dump vector table for hardware type

The `dump-vector-table` fields define memory areas to be dumped. These are read-only fields and cannot be changed. Except as noted, fields are common across cards, FDDI/Q defaults are shown in this example.

`dump-vector-table 6`

`index = 6` - the vector table index, set to 0 or non-zero

`hw-type = fddi-v2` - the hardware type of this interface

`description = "FDDI/Q default dump vectors"` - vector description, 128 characters

```
segment-table = < { 1 "fddi/Q CPU0 core memory" 2097152 2097152 }
                  { 2 "fddi/Q I+
```

3rd-level fields

To view the list of segment tables for this card

```
super> get . dump 6 segment-table
1 = { 1 "fddi/Q CPU0 core memory" 2097152 2097152 }
2 = { 2 "fddi/Q IPC memory" 8388608 32800 }
3 = { 3 "fddi/Q shared descriptor memory" 16777216 131072 }
4 = { 4 "fddi/Q CPU1 core memory" 4194304 2097152 }
```

Here are representative `dump-vector-table` segment table fields at the third level, values for FDDI/Q cards are shown.

`segment-table 1`

`index = 1` - index of hardware type

`description = "fddi/Q CPU0 core memory"`
- object name (`sys_vector_seg_desc`), 128 characters

`start = 2097152` - object name (`sys_vector_seg_start`), set to 0 or non-zero

`length = 1048576` - object name (`sys_vector_seg_length`), set to 0 or non-zero

Here are the other FDDI/Q segment tables 2–4:

```
index = 2
description = "fddi/Q IPC memory"
start = 8388608
length = 32800

index = 3
description = "fddi/Q shared descriptor memory"
start = 16777216
length = 131072

index = 4
description = "fddi/Q CPU1 core memory"
start = 4194304
length = 2097152
```

Load profile components

Figure 1-4 shows the fields in each level of the Load profile. The fields are described on the pages following.

Load

```
hippi = { "" "" on 0 0 < { 1 usr libexec } > }
rmb = { /usr/libexec/portcards/rm.run N/A off 0 0 < { 1 "" N } > }
hssi = { /usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_tx.run +
dev1 = { /usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_tx.run +
atm-oc3-v2 = { /usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards/atmq_t+
fddi-v2 = { /usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fddiq-1.r+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 0 < { 1 "" N } > }
ethernet-v1 = { /usr/libexec/portcards/ether_rx.run /usr/libexec/portcards/ethe+
sonet-v1 = { /usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/sonet_t+
```

hippi

```
type = hippi
rx-config = 0
rx-path = ""
tx-config = 0
tx-path = ""
enable-boot-seq = on
mode = 0
iterations = 1
boot-seq-table = < { 1 usr libexec } >
```

rmb

```
type = rmb      (control board)
rx-config = 0
rx-path = /usr/libexec/portcards/rm.run
tx-config = 0
tx-path = N/A
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

hssi

```
type = hssi
rx-config = 0
rx-path = /usr/libexec/portcards/hssi_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/hssi_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

dev1

```
type = dev1
rx-config = 0
rx-path = /usr/libexec/portcards/dev1_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/dev1_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

atm-oc3-v2

```
type = atm-oc3-v2
rx-config = 0
rx-path = /usr/libexec/portcards/atmq_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/atmq_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

fddi-v2

```
type = fddi-v2
rx-config = 0
rx-path = /usr/libexec/portcards/fddiq-0.run
tx-config = 0
tx-path = /usr/libexec/portcards/fddiq-1.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

atm-oc12-v1

```
type = atm-oc12-v1
rx-config = 0
rx-path = /usr/libexec/portcards/atm-12.run
tx-config = 0
tx-path = N/A
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

ethernet-v1

```
type = ethernet-v1
rx-config = 0
rx-path = /usr/libexec/portcards/ether_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/ether_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

sonet-v1

```
type = sonet-v1
rx-config = 0
rx-path = /usr/libexec/portcards/sonet_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/sonet_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table =  <  >
```

Figure 1-4. Diagram of Load profile levels

Load profile field descriptions

Load profile fields set system-wide values not usually changed. To change values on a specific card, change settings in the `load` field in the Card profile.

1st-level fields

The Load fields at the first level are complex structures.

2nd-level fields

Here is a representative Load hardware field at the second level, FDDI/Q defaults are shown.

fddi:

`type = fddi-v2` - specific media type

`rx-config = 0` - hardware configuration for receive CPU, set to 0 or non-zero

`rx-path = /usr/libexec/portcards/fddiq-0.run`

- default receive binary for this card

`tx-config = 0` - hardware configuration for transmit CPU (if dual CPU)

`tx-path = /usr/libexec/portcards/fddiq-1.run` default transmit binary for this port card (if dual CPU), set to NA or leave empty for single-processor card

`enable-boot-seq = off` - turn the use of boot sequences on or off

`mode = 0` - set mode for boot sequences, set to 0 or non-zero

`iterations = 1` - number of iterations for a binary to execute, set to 0 or non-zero

`boot-seq-table = < >` - an empty field, use **new** to create a configurable image

You can enter a boot sequence for running diagnostics prior to executing run-time code, or loading successive runtime binaries into an cards. In the case of diagnostics, **grbootd** will halt the load sequence if a diagnostic fails.

System profile components

Figure 1-6 shows the fields in the System profile for a GRF 400.

System profile - GRF 400

```
os-level = 1.4.12
hostname =  grf.site.com
chassis =  GRF 400
ip-address = xxx.xxx.xxx.xxx
netmask =  0.0.0.0
default-route =  0.0.0.0
hippi-ifield-shift = 5
enable-congest = disabled
num-slots =  4
rmb-load-path = /usr/libexec/portcards/rm.run
rmb-dump-config = 4
physical-memory = 512
hardware-revision = "Not Available"
chassis-revision = "Not Available"
xilinx-revision = 8
num-fans = 3
num-pwr-supply = 1
Forward_Directed_Bcast_Pkts = disabled
```

Figure 1-5. Diagram of GRF 400 System profile level (single level)

Figure 1-6 shows the fields in the System profile for a GRF 1600.

System profile GRF 1600

```
os-lev- el = 1.4.12
hostname =  grf.site.com
chassis =  GRF 1600
ip-address = xxx.xxx.xxx.xxx
netmask =  0.0.0.0
default-route =  0.0.0.0
hippi-ifield-shift = 5
enable-congest = disabled
num-slots =  16
rmb-load-path = /usr/libexec/portcards/rm.run
rmb-dump-config = 4
physical-memory = 512
hardware-revision = "Not Available"
chassis-revision = 1
xilinx-revision = 8
num-fans = 2
num-pwr-supply = 1
Forward_Directed_Bcast_Pkts = disabled
```

Figure 1-6. Diagram of GRF 1600 System profile level (single level)

System profile field descriptions

Here are the System profile fields, read-only values are read from the `etc/netstart` file:

`os-level = 1.4.12` - read-only, Ascend Embedded/OS release level

`hostname = grf.site.com` - read-only, host name of this GRF

`ip-address = x.x.x.x` - read-only, host IP address

`netmask = 0.0.0.0` - read-only, system netmask field

`default-route = 0.0.0.0` - read-only, system IP address or netmask field

`hippi-ifield-shift = 5` - number of bit positions to shift an I-field, can be 4 or 5

`enable-congest = disabled` - congestion management enabled or disabled

`num-slots = 4` - read-only, number of slots in the chassis

`rmb-load-path = /usr/libexec/portcards/rm.run` - control board (RMB) load path

`rmb-dump-config = 4`

Default setting dumps control board (RMB) when it panics, other setting options are:
0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

`physical-memory = 256` - read-only, system memory in MB

`hardware-revision = “Not Available"` - not currently used on GRF 400 or 1600

`chassis-revision = 1` - read-only, GRF 1600 chassis revision level, not currently used on GRF 400

`xilinx-revision = 8` - read-only, revision of hardware XILINX

`num-fans = 3` - read-only, number of cooling fans, GRF 400 has 3, GRF 1600 has 2

`num-pwr-supply = 1` - read-only, number of power supplies, 2 = redundant unit

`Forward_Directed_Bcast_Pkts = disabled` - enables/disables the forwarding of directed broadcast packets, disabled by default

User profile components

Figure 1-7 shows the fields in each level of the User profile.

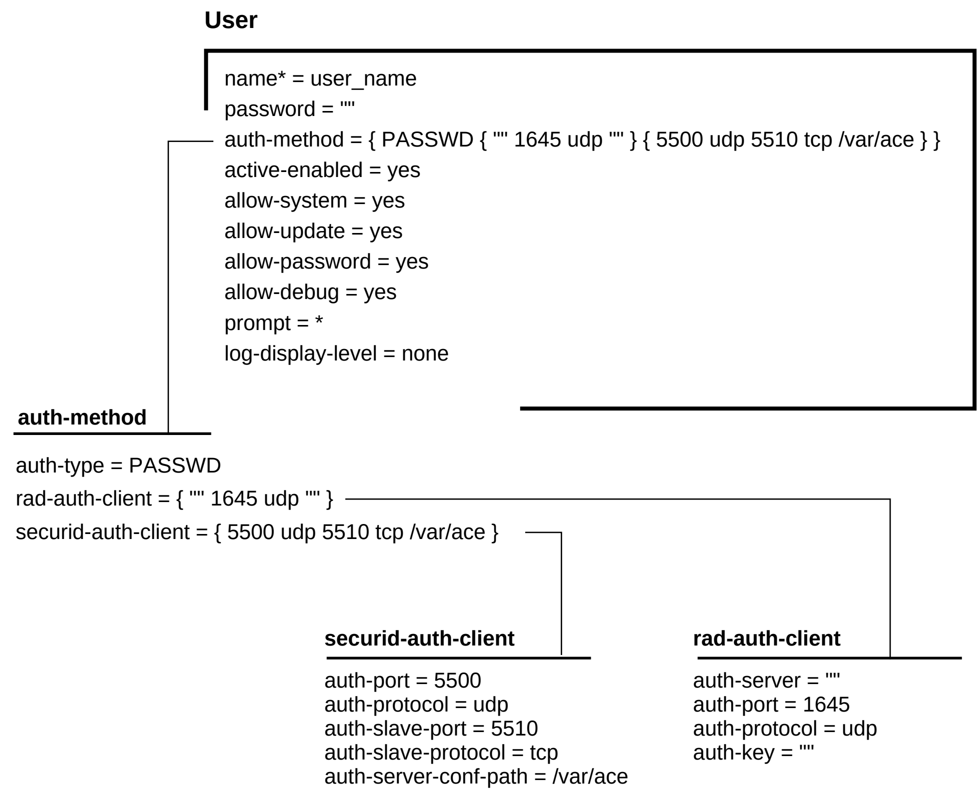


Figure 1-7. Diagram of User profile levels

User profiles are not UNIX accounts. They only allow local router configuration access to users with an assigned profile. You cannot telnet into the GRF using a profile name.

CLI and UNIX passwords

The UNIX `root` and `netstar` log ins have passwords associated with UNIX user accounts. These passwords are different from the CLI user profile password. The CLI **auth** command controls permissions for CLI user logins. A user can log in to the CLI by using the **auth** command to return a password prompt. If the user supplies the correct password, then the CLI permissions specified for the user are granted.

When you log into the GRF, it uses UNIX authentication. When you enter the CLI, it automatically assigns you a CLI authentication as "super" if you log in as root, otherwise, it assigns "default". You can display your CLI authentication with the **whoami** command. Your CLI authentication affects which fields you can see and which commands you can invoke within the CLI.

User profile field descriptions

1st-level fields

Here are the fields at the first level.

`name* =` - name associated with specific User profile, up to 24 characters

`password =` - password of the user asking for validation, up to 21 characters

`auth-method = {PASSWD { "" 1645 udp ""} { 550 udp 5510 tcp /var/ace} }`
- sets method to use for login validation

`active-enabled =` - yes if this user account is enabled for use, no if not

`allow-system =` - yes if this user may use system commands, no if not

`allow-update =` - yes if this user may use update commands, no if not

`allow-password =` - yes if this user may user may view password fields, no if not

`allow-debug =` - yes if this user may use debug commands, no if not

`prompt = *` - the prompt displayed to the user, the value '*' is substituted with the user's name

`log-display-level =`

Sets level of log message to display immediately to the user

none - no log messages are saved/displayed

emergency - bad event occurs, normal operation is doubtful

alert - bad event occurs, but normal operation likely

critical - an interface goes down, also used for security errors

error - something that should not occur has occurred

warning - message for unusual event in otherwise normal operation, for example, a login failure due to entry of bad user name or password

notice - things of interest in normal operation, for example, a link comes up or goes down

info - state and status changes that are normally not of general interest

debug - messages of interest when debugging unit configuration

2nd-level fields

`auth-type = PASSWD`- sets the type of authentication to perform, values are:

```
none
PASSWD
TACACS
RADIOUS
SECURID
```

Fields for `rad-auth-client`:

`auth-server = ""` - read-only, IP address of the RADIUS authentication server

`auth-port = 1645` - read-only, UDP port to use for RADIUS authentication

`auth-protocol = udp` - read-only, RADIUS port protocol to communicate with the RADIUS server `auth-key = ""` - read-only, RADIUS authentication access key shared with the RADIUS server

Fields for `securid-auth-client`:

`auth-port = 5500` - read-only, the port to use to communicate with SecurID server

`auth-protocol = udp` - read-only, protocol of the port to use to communicate with the SecurID server

`auth-slave-port = 5510` - read-only, TCP port to use to communicate with the SecurID server

`auth-slave-protocol = tcp` - read-only, protocol of the slave port used to communicate with the SecurID server

`auth-server-conf-path = /var/ace` - read-only, location of the `sdconf.rec` file produced by the SecurID/ACE server

Working with profiles

All data is stored in profiles. Profiles are complex structures that contain one or more fields. A field may be one of several data types:

- a number
- a boolean value
- an enumerated value
- a hexadecimal number
- an IP address
- a text string
- a complex structure
- a list of complex structures.

To look at or change a particular piece of data, you need to access the profile in which that data is stored. You use profile management commands to retrieve, read, and write in a profile.

Profile management commands

- **dir**
  List the types of profiles and their indexes, a "directory" of management information.
- **read**
  Read a profile in preparation for looking at or changing individual fields.
- **get**
  Show the value of a specific field in a specific profile.
- **ls**
  An alias for get.
- **list**
  Change the current context to the specified field or list the fields in the current profile.
- **cd**
  An alias for list.
- **set**
  Change the value of a specific field in a specific profile.
- **write**
  Validate the profile and apply any changes made.
- **new**
  Create a new instance of the specified profile type.
- **delete**
  Remove a profile instance from local storage.
- **save**
  Saves the specified profile configuration to a script that can be loaded at a later time. If no profile is specified, all savable profiles are saved.

– **load**
  Load the profile configuration script stored in the specified file.

– **pwd**
  Shows the current location (context) in the tree.

Access the profile set

Use the **dir** command to look at the list of profile types:

```
super> dir
CARD     Card info
DUMP     System dump information
LOAD     System load information
SYSTEM   System info
USER     Administrative user accounts
super>
```

Use the **dir <*profile_type*>** command to look at all the profiles of a specific type.

The output is in four columns:

Size in bytes	Modification date	Modification time	Index
92	9/19/98	11:12:31	default

This example looks at a single-instance profile, System:

```
super> dir system
27  9/01/98 13:11:51  .
```

This example looks at a profile type with multiple instances, User:

```
super> dir user
92  9/19/98 11:12:31  default
103 8/8/98  09:09:31  admin
106 6/16/98 11:19:31  super
88  11/22/98 16:03:31  bob
```

Profiles that exist in external databases are not listed.

Read profile into local memory

To look at the data in a profile, use the **read** command to put the profile into local memory. Once the profile is in local memory, you can view or change the data, and then save changes to make them permanent. After a profile is read, you receive a response indicating that the read was successful.

Note that you can work with only one profile at a time. When you read another profile, it replaces the previous profile in local memory.

The **read** command syntax is:

`read` *`profile-type`* `[` *`profile-index`* `]`

Here are examples of **read** for single-instance and multiple-instance profile types:

```
super> read system
SYSTEM read

super> read user bob
USER/bob read
```

Viewing the contents of a profile

After you read a profile into local memory, the command you use to view the contents of the profile depends upon whether or not you intend to make changes to the profile.

If you plan to make changes, use **list** or **cd**— use the “elevator” to move fields into memory so you can write and save changes.

If you only want to look, use **get** or **ls** — stay on the “scenic overlook”.
Important: while you are in one profile, **get** and **ls** let you look horizontally across to another profile without leaving where you are.

Viewing to change the contents of a profile

If you plan to make changes, use **list** or **cd** to bring fields into local memory.

The **list** command has the syntax:

```
list [field-name] [field-index] [...]
```

cd is an alias for the **list** command. A simple **list** command results in a paged output of a list of fields and their values in the current profile. Each field is displayed using the format:

```
field-name = field-value
```

Here is the output of **read** and **list** commands on the profile User default:

```
super> read user default
USER/default read
super> cd
name* = default
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp +
active-enabled = yes
allow-system = no
allow-update = no
allow-password = no
allow-debug = no
prompt = GRF=>
log-display-level = none
```

Use **list** or **cd** `field-name` to look at what's missing in the + truncated `auth-method` field:

```
super> list auth-method
auth-type = PASSWD
rad-auth-client = { "" 1645 udp "" }
securid-auth-client = { 5500 udp 5510 tcp /var/ace }
super>
```

Use **set** `field-name` **?** to get more information about a specific parameter in the field:

```
super> set securid-auth-client ?
securid-auth-client:
   SECURID authentication client information
   Complex field, cannot be set directly
super>
```

Checking another profile from a profile

In this example, the goal is to change the CRC setting on interface 0 of the HSSI card in slot 3 to match that of the HSSI card in slot 1. This example first shows how you move down in the Card 3 profile to the CRC field using the **list** command. Then it shows how you use **get** and a field-name path to look at the setting in another profile, the profile for the HSSI card in slot 1:

```
super> read card 3
CARD/3 read
super> list .
card-num* = 3
media-type = hssi
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < { 0{ off on 10 3}{single off}{ " "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }

super> list ports 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 207 }
hssi = { 0 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 increment 5 300 10 10 03:00:0f:c0 disable+

super> list hs
source-clock = 0
CRC-type = 16-bit
```

To show that you are at a location in the Card 3 profile and have read fields into local memory in order to make a change, this example does a **whereami** here:

```
super> where
CARD 3/ports 0/hssi
```

Here is where you use **get** to look into another profile, at the settings in the HSSI card in slot 1:

```
super> get card 1 ports 0 hssi
source-clock = 0
CRC-type = 32-bit
super>
```

Do **whereami** again just to show you are still where you want to be to set and write the change!

```
super> where
CARD 3/ports 0/hssi

super> set CRC-type = 32-bit
super> write
CARD 3/written
```

Moving up and down in a profile

The **list** `field-name` command changes the current location in the tree "down" one level to that field. Two dots (..) signify the level above the current location in the tree.

Here are examples of how **cd** and **list** commands are used with .. to change levels in a profile. Notice that `ports` is a list of complex fields and `load`, `dump`, and `config` are all complex fields.

First, read the profile for card number 2

```
super> read card 2
CARD/2 read
```

List the contents of the `hssi-frame-protocol` field (down one level):

```
super> list hssi-frame-protocol
hssi-frame-protocol = Frame-Relay
```

Move back (up) one level and list the contents at that location, card 2:

```
super> cd ..
card-num* = 2
media-type = hssi
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < { 0{ off on 10 3}{single off}{ " "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
super>
```

List the ports without an index specified (down one level):

```
super> cd ports
0 = {0{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
1 = {1{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
2 = {2{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
3 = {3{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
4 = {4{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
5 = {5{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
6 = {6{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
7 = {7{ off on 10 3}{single off} {"" "" 1 sonet internal-oscillator 0 20+
super>
```

List the fields in port 1 (down one more level):

```
super> list 1
port_num = 1
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 207 }
hssi = { 0 16-bit }
ether = { autonegotiate }
hippi = { 1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disa+
super>
```

Remember, the **get** and **ls** commands do not move you up and down, or change your level in a profile.

Getting field information

The **get** command retrieves the names and contents of fields within profiles without changing the user's location within the tree or affecting the last profile read. **ls** is an alias of **get**.

The syntax is:

```
get [. | profile-type [ profile-index ] ] [ field-name field-index ... ]
```

The **get** command operates on the last profile read if a dot (.) is used in place of the profile type and profile index. Providing a profile-type profile-index does not effect the last profile read. If no field-name is present upon the command line, every field in the requested profile is listed using paged output.

This example gets the fields from the last profile read:

```
super> get .
name* = default
passwd = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp +
active-enabled = yes
allow-system = no
allow-update = no
allow-password = no
allow-debug = no
prompt = GRF_400>
log-display-level = none
super>
```

This example gets a specific field from the last profile read:

```
super> ls . name
name* = default
```

This example gets a specific field from a profile other than the one last read:

```
super> get system os-level
os-level = 1.4
```

Checking where you are

Check where you are with the **pwd** command or its alias, **whereami,** or a shortcut such as **whe**.

The **pwd** command returns the user's current location or level in the tree. The output is similar to a path in a file system. Each level in the tree is separated by forward slashes (/). Each level in the tree is typically another profile. If a profile is indexed (a member of a list), the index follows the profile name. (**whereami** and **pwd** are used interchangeably in the examples here.)

Here is the output from the very top, when no profiles have been read:

```
super> pwd
/
```

Here is the output after a read of a single-instance profile:

```
super> read system
SYSTEM read
super> pwd
SYSTEM
```

Here is the output after a read of a multiple-instance profile and a coffee break:

```
super> read card 2
CARD/2 read
. . .
super> whereami
CARD 2
```

Here is the output after moving deeper into the tree:

```
super> cd ports 1
port_num = 1
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 207 }
hssi = { 0 16-bit }
ether = { autonegotiate }
hippi = { 1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disa+

super> whe
CARD 2/ports 1
super> cd ..
```

To access another first-level field, use **cd ..** to move back up to the Card level.

Changing a profile

The **set** command modifies fields in the last profile read. Modifications do not take effect until the profile is written. **set** has two formats:

```
set field-name = field-value
```

```
set field-name ?
```

The first format changes a field, the second format gets help on the type of values to which the field can be set. In both cases, the `field-name` must match a name in the last profile read. When changing a field, the `field-value` is everything between the white space following the = and the end of the line.

This example changes Bob's prompt and writes the change:

```
super> read user bob
USER/bob read
super> get user bob
name* = bob
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace }}
active-enabled = yes
allow-system = yes
allow-update = yes
allow-password = yes
allow-debug = yes
prompt = *
log-display-level = none

super> set prompt = support1
super> write
USER/bob written
```

Multiple set commands

You can enter several **set** commands while you are in the same field group before you have to do a `write` to save changes. This is useful in the User profile where there are many fields at the same level:

```
super> read user bob
USER/bob read

super> set allow-system = no
super> set allow-update = no
super> set allow-password = no
super> set log-display-level = debug
super> write
USER/bob written
```

Writing changes

The **write** command validates and stores the current profile into memory. **write** does not require a `profile-type` or `profile-index` since it always writes the current, active profile.

If you do a **set** but do not follow with a **write**, you will get a message warning that your changes will be lost.

```
WARNING:  You are about to discard your changes.
If you wish to save these changes, please write the current
profile before reading or creating a new profile.
Do you wish to continue without saving changes?  [y/n]
```

Deleting a profile

The **delete** command removes an instance of a profile from local storage. The format of the command is:

`delete profile-type profile-index` (to delete a main-level profile instance)
`delete field-name field-index` (to delete a profile list member)

The `field-name` is the name of the field that is the profile list.

The `profile-index` or `field-index` is not optional because only indexed profiles can be deleted.

Warning: Memory used by the profile is immediately made available to the system. Once deleted, a profile is gone forever – it cannot be “undeleted”.

The **delete** command always responds with a query for confirmation of the deletion:

```
super> delete user operator
Delete profile USER/operator? [y/n] y
USER/operator deleted
```

To delete a member of a profile list, read the main-line profile first:

```
super> read card 1
CARD/1 read
super> delete port 1
Delete ports/1? [y/n] y
ports/1 deleted
```

If you change your mind when queried, here is what you see:

```
super> delete ports 1
Delete ports/1? [y/n] n
super>
```

If you attempt to delete an instance that does not exist, here are examples of what you see:

```
super> delete ports 2
Delete ports/2? [y/n] y
ports/2 does not exist
super> delete user tom
error: specified profile not found
```

Saving and loading alternate profiles

Use the **save** and **load** commands to save off main-line profiles to a file that you can restore again later. All files are saved in the `/etc/prof` directory.

Using the save command

The **save** command saves the current profile configuration in a script form to permanent storage in *`filename`*. This file can be loaded at a later date to restore the previous configuration.

Note:
The CLI remembers which profile *`filename`* was saved as and automatically restores it as the active version of that profile type.

To save the configuration to a file, the syntax of the **save** command is:

```
save [-a] [-m] filename [profile-type [profile-index]]
```

To write the configuration to the screen, the syntax is:

```
save [-a] [-m] console [profile-type [profile-index]]
```

If a `profile-type` is not specified, all savable profiles are saved. If a `profile-type` is specified, but a `profile-index` is not specified (and it is a multiple-instance profile), all profiles of that type are saved.

If the `-a` option is specified, all fields are explicitly saved. Otherwise, only those fields whose contents differ from the default values are saved.

If the `-m` option is specified, all fields are saved by their field numbers rather than by their field names.

If the current user does not have password accessibility, a message appears warning the user not to save any profiles that contain passwords. Without password accessibility, all passwords are written as strings of stars.

All files are saved in the `/etc/prof` directory. If the specified profile already exists, a message appears, warning the user that this file already exists, and asking the user if s/he wants to overwrite this file.

You can “save” the current profile to the console, this type of **save** just displays the script on the screen, reflecting system activity. To save the current profile to the console:

```
super> save console system
; saving profiles of type SYSTEM
; profile saved Mon Feb  3 17:27:40 1997
new SYSTEM
set rmb-load-path = /some/alternative/load/path
write -f
```

To save the System profile using the `-a` option:

```
super> save -a console system
new SYSTEM
set hippi-ifield-shift = 0
set enable-congest = disabled
```

```
set num-slots = 0
set rmb-load-path = /some/alternative/load/path
set rmb-dump-config = 4
write -f
super>
```

To save the System profile using the `-m` option:

```
super> save -m console system
; saving profiles of type SYSTEM
; profile saved Mon Aug 11 15:56:26 1997
new SYSTEM
set 9 = /some/alternative/load/path
write -f
;
```

To save all savable profiles to a specified file:

```
super> save all.conf
```

To save all User profiles to a specified file:

```
super> save user.conf user
super>
```

This example saves the User admin profile to a specified file:

```
super> save admin.conf user admin
super>
```

This is what you see if you attempt to save to a file that already exists:

```
super> save all.conf card

WARNING: all.conf already exists.  If you choose to save to this file,
all configuration information that now exists in all.conf will be over-
written. Continue? [y/n] n
save aborted
super>
```

This is what you see if you attempt to save a user profile without password access:

```
super> save default.conf user default
WARNING: the current user has insufficient rights to view password
fields.  A configuration saved under this circumstance should not be used
to restore profiles containing passwords.
Save anyway? [y/n] n
super>
```

Using the load command

The **load** command runs a previously-saved configuration script to restore (load) a previous configuration.

The syntax of the **load** command is:

`load` *`filename`*

where *`filename`* is the name of the file in which the configuration script is saved. These files should be located in `/etc/prof`.

To load a previous configuration saved as `system.conf`:

```
super> load system.conf
SYSTEM read
SYSTEM written
super>
```

Here is what you see when you attempt to load a file that does not exist:

```
super> load special.conf
error:  special.conf does not exist
super>
```

Creating a new profile

A new profile can be created in two ways.

- use the **read**, **set**, and **write** commands together
- use the **new** command

A profile cannot be copied and then renamed because two profiles of a given type cannot have the same index (name). A combination of **read** and **write** commands can be used to copy everything except the index.

In this example, a new user profile for George is created. Additional user profiles are created

```
super> dir user
92  01/31/98 10:16:08  default
103  01/31/98 10:16:09  admin
106  01/31/98 10:16:10  super
99  02/03/98 15:27:00  frank

super> read user frank
USER/frank read

super> set name = george
super> write
USER/george written

super> dir user
 92  01/31/98 10:16:08  default
103  01/31/98 10:16:09  admin
106  01/31/98 10:16:10  super
 99  02/03/98 15:27:00  frank
100  02/03/98 16:00:00  george
```

Note:
If user profile "george" already exists, the **write** command replaces it with a copy of the factory profile:

```
super> write
USER/george written
```

Adding user profiles

In the CLI, you create additional User profiles with the **new** command. User profiles are not UNIX accounts. They only allow local router configuration access to users with an assigned profile. You cannot telnet into the GRF using a profile name.

To create an account for Bob, enter:

```
super> new user bob
USER/bob written
```

The response tells you that a basic account exists for Bob. Edit it to set access permissions, prompt text (bob>), and password.

CLI auth passwords

The UNIX `root` and `netstar` log ins have passwords associated with UNIX user accounts. These passwords are different from the CLI user profile password. The CLI **auth** command controls permissions for CLI user logins. A user can log in to the CLI by using the **auth** command to return a password prompt. If the user supplies the correct password, then the CLI permissions specified for the user are granted.

When you log into the GRF, it uses UNIX authentication. When you enter the CLI, it automatically assigns you a CLI authentication as "super" if you log in as root, otherwise, it assigns "default". You can display your CLI authentication with the **whoami** command. Your CLI authentication affects which fields you can see and which commands you can invoke within the CLI.

Using the new command

The **new** command creates a new instance in local memory of a main-level profile (Card, System, User, Load, Dump), or a new instance of a member of a profile list in a list field. The new instance is not permanent until the main-level profile is written.

The syntax of **new** is:

`new profile-type [profile-index]`	(to create a main-level profile instance)
`new field-name [field-index]`	(to create a new profile list member)

If a `profile-index` or `field-index` is not specified, the default index is used.

In this example, a new main-level User profile is created for Fred (note that the default index is used):

```
super> new user
USER/default read
```

If you specify a new `profile-index`, the correctly-named User profile is created:

```
super> new user fred
USER/fred read
```

If you try to create a main-level profile instance that already exists, it just reads the existing profile:

```
super> new user admin
USER/admin read
```

In this example, a new member of the Ports profile list is created on Card 1. If you specify a new `field-index`, the correctly-named Port profile is created:

```
super> new port 10
ports/10 created
```

If port 10 already exists, you see this:

```
super> new port 10
error: profile already exists
```

Configuring System Parameters

2

Chapter 2 describes how to set up GRF system parameters and enable network services such as SNMP and IP routing options.

Unless otherwise noted, the information in this manual applies to the GRF 400, GRF 1600, and to GRF and GR-II systems using RMS nodes. Examples use slot numbers 0–3 or 0–15, particular slot numbers are not significant in examples.

Chapter 2 covers these topics:

A note about grinchd.conf

GRF profiles replace the `/etc/grinchd.conf` file. Variables from `grinchd.conf` are now configuration fields in the Card, User, Dump, Load, and System profiles. Use the command-line interface (CLI) described in Chapter 1 to access profiles and assign field values.

Overview of system configuration

The configuration procedures described in this chapter are performed after the automatic configuration script has successfully run and you have logged in.

The following tasks are described:

– root and administrative logins

– configuring system logging to a PCMCIA device

– setting up IP addressing, the loopback alias, alias addressing, ISO addresses

– changing the GRF hostname

– creating host telnet access sites (`/etc/ttys`)

– enabling IP routing options, static and source routing, directed broadcast forwarding

– ARP processing on the GRF (**grarp**)

– configuring and disabling SNMP, community names, system information, traps

– starting dynamic routing, configuring GateD

– static and dynamic options for Equal Cost Multi-path routing

– setting the GRF as client for TACACS+, RADIUS, and securID authentication

Configuration files and their uses

`aitmd.conf`	- defining parameters for ATMP (VPN tunneling protocol)
`bridged.conf`	- defining system bridging services
`filterd.conf`	- defining system filtering services
`gated.conf`	- enabling dynamic routing functions
`grarp.conf`	- mapping IP addresses to physical hardware addresses
`gratm.conf`	- configuring ATM PVCs and SVCs
`grfr.conf`	- configuring Frame Relay on HSSI and SONET cards
`grifconfig.conf`	- identifying each logical interface on a media card
`grlamap.conf`	- mapping HIPPI logical addresses to media cards
`grppp.conf`	- assigning PPP to HSSI and SONET interfaces
`grroute.conf`	- setting static routes
`snmpd.conf`	- enabling SNMP capabilities
`syslog.conf`	- configures remote logging of log files via **syslogd**

Use grconslog to monitor the GRF

It is common practice to telnet into the GRF, enter **grconslog -vf,** and keep the window open to monitor ongoing system events as they are reported. Use the abort or equivalent key to quit the log. The `gr.console.log` displays all types of events include card resets and panics, user log ons, and configuration changes. Refer to the GRF Reference Guide for a description of **grconslog** options.

Logging on to a GRF

This section describes the root and administrative log ins to the GRF. The root log in is the UNIX-equivalent super user. The first log in while you are connected using the VT-100 terminal must be a root log in.

After you connect the GRF to the local Ethernet, you telnet to the GRF and use the administrative log in. Certain tasks on the GRF can be done only while you are logged in as root. Generally, you will log in to the GRF as an administrative user. Both are explained here.

root log on

As described in the *GRF 400/1600 Getting Started* manual, the GRF begins to boot as soon as it is powered on. The startup configuration script runs, and prompts you to enter the IP address and other information that enables the GRF to connect to your local Ethernet. When the script finishes, it displays this message:

```
Press <Enter> to continue.
```

Pressing the Enter key brings up the `User` prompt. This is the first log in to the GRF, it should be a `root` log in. The GRF is shipped with Ascend (capital A) as the root password.

```
User: root
Password:
```

At the Password: prompt, enter Ascend. When a password is entered, it is not echoed (displayed) on the screen.

The super> prompt appears. The super> prompt indicates you are in the CLI:

```
super>
```

When you log in to a GRF as `root`, you automatically get the CLI shell. In the CLI, `root` is super user, hence the `super>` prompt.

Change root password

Ascend recommends that you change this preset password now, before you begin system configuration. If you are at the `super>` prompt, you are in the CLI. You need to be in the UNIX shell to change the password. Use **sh** to invoke the UNIX shell. Each time you start the shell, you see the Ascend copyright and version notice.

```
super> sh
#
```

At the # prompt, use the **passwd** command to start the process and enter `NetStar` at the Old password prompt:

```
#  passwd
Old password: • • • • • • • •
New password:
```

Enter the new password twice, as prompted: (use at least eight alphanumeric characters)

```
New password: • • • • • • • • • • •
Retype new password: • • • • • • • • • • •
#
```

Type **exit** to leave the shell.

Administrative log on

After you attach the Ethernet connection from an administrative LAN to the control board, `telnet` from your administrative station to the GRF using the IP address assigned in the configuration script. This is where you set up the user

At the User: prompt, enter: `netstar` *(all lowercase)*
At the Password: prompt, enter: `NetStar` *(capital N, capital S)*

```
User: netstar
Password: • • • • • • •
#
```

At the # UNIX prompt, enter: `su` *(all lowercase)*
At the Password: prompt, enter: `Ascend` *(capital A)*

Change to superuser.

```
#  su -
```

At the Password: prompt, enter the new root password you created or enter `Ascend` if you have not changed the preset password:

```
Password: • • • • • • • • • •
super>
```

The `super>` prompt indicates you are in the command-line interface (CLI).

CLI and UNIX passwords

The UNIX logins described here have passwords associated with UNIX user accounts. These passwords are different from the CLI user profile password. The CLI **auth** command controls permissions for CLI user logins. A user can log in to the CLI by using the **auth** command to return a password prompt. If the user supplies the correct password, then the CLI permissions specified for the user are granted.

When you log into the GRF, it uses UNIX authentication. You cannot telnet using a user profile name or password. When you enter the CLI, it automatically assigns you a CLI authentication as "super" if you log in as root, otherwise, it assigns "default". You can display your CLI authentication with the **whoami** command. Your CLI authentication affects which fields you can see and which commands you can invoke within the CLI.

Configuration tasks - shell or CLI ?

To configure and manage the GRF, you will use both the CLI command set and the UNIX shell. You switch between them, there are no nested levels of shell and CLI.

You must be in the UNIX shell to edit the `/etc/xxx.conf` configuration files. The `/etc` directory contains the configuration files.

Otherwise, many system management and configuration commands are available in the CLI as well as the UNIX shell. Enter a "?" to retrieve a list of CLI commands. All system commands are described in the *GRF Reference Guide*.

Configure system logging

A portion of this section appears in the *GRF 400/1600 Getting Starte*d manual because it is helpful to have logging configured during initial installation and set up, prior to bringing up the media cards.

When the GRF first boots and loads, logging is not enabled. System memory restrictions on the GRF control board require that logging be to external storage. If target storage is not specified to receive log entries, log entries are not saved.

There are three options for logging:

- "local" logging and dumping to an external PCMCIA flash device inserted in the control board
- remote logging (network) to a **syslog** server
- remote logging (network) to an NFS-mounted file system

This section includes only the procedure to set up local logging to a PCMCIA device in the control board. Refer to the *Getting Started* manual for the two other procedures.

Configure logging to a PCMCIA device

Do not remove a mounted PCMCIA device from the slot, unmount it first. You will get an error message if you remove a mounted device.

Three commands enable remote management of PCMCIA slots. The **csconfig** ***slot_number*** command returns status of each PCMCIA slot. The **csconfig** ***slot_number*** **up** and **csconfig** ***slot_number*** **down** commands mark the specified PCMCIA slot up or down, respectively. An example is included in the device installation procedure which follows.

The procedure formats and initializes an external device (flash or disk), temporarily mounts it on `/mnt`, creates subdirectories and symbolic links, and creates a permanent site file for storing the symbolic links. The example installs a flash device into slot A and specifies that logs and dumps be sent to directories on the device. You do not need to specify whether the device is a flash or spinning disk.

The PCMCIA device is named according to the slot it occupies:

- `/dev/wd2a`, the device residing in slot A (often a spinning disk)
- `/dev/wd3a` , the device residing in slot B (a flash disk or modem)

1 Insert the PCMCIA device into slot A on the GRF control board.

The thickness of a spinning disk device requires it be installed in slot A. A flash disk or modem can be installed in either A or B.

2 Log in as `root` to the GRF, start the UNIX shell, and execute these commands from the shell:

```
super> sh
# cd /
# iflash -A
# mountf -A -w -m /mnt
```

```
# mkdir /mnt/crash
# mkdir /mnt/portcards
# cd /var
# mv crash crash.orig
# mv portcards portcards.orig
# ln -s /var/log/portcards /var/portcards
# ln -s /var/log/crash /var/crash
# grsite --perm portcards crash
# cd /var/log
# pax -rw -pe -v . /mnt
# umountf -A
```

3 Edit the file `/etc/fstab`:

```
# cd /etc
# vi fstab
```

Use the UNIX editor to add this line as shown at the bottom of the excerpt:
`/dev/wd2a  /var/log  ufs   rw  0 2  #PCMCIA slot A, use wd3a for B`

Here is the portion of the file where you will add the specified line:

```
# Filesystem mount table information.  See the fstab(5) man page
# and the /etc/fstab.sample file for more information and examples.
#
# Each line is of the form:
# device    mount_point    type    flags   dump    fsck_pass
#
# Note that multiple flags (when used) are specified as a
# comma separated list without spaces.
#
# Blank lines and lines beginning with `#' are comments.
#
/dev/rd0a     /      ufs   rw  0 0
/dev/wd2a  /var/log  ufs   rw  0 2  #PCMCIA slot A, use wd3a for B
```

4 Edit the file `/etc/syslog.conf` to specify the location where the logs will be kept. After you edit `/etc/syslog.conf`, you need to send a -HUP signal to **syslogd**.

When installing a spinning disk or flash disk

Uncomment the local log configuration lines in the "`Log messages to Disk`" section by removing `#disk#` from each line and specify `/var/log` as the directory for each log:

These are the first four lines in the section:

```
#disk#*.err;*.notice;kern.debug;lpr,auth.info;mail.crit
          /var/log/messages
#disk# cron.info                            /var/log/cron
#disk# local0.info                          /var/log/gritd.packets
#disk# local1.info                          /var/log/gr.console
```

The entries should now look like the following:

```
*.err;*.notice;kern.debug;lpr,auth.info;mail.crit     /var/log/messages
cron.info                                          /var/log/cron
local0.info                                        /var/log/gritd.packets
local1.info                                        /var/log/gr.console
local2.*                                           /var/log/gr.boot
local3.*                                           /var/log/grinchd.log
local4.*                                           /var/log/gr.conferrs
local5.*                                           /var/log/mib2d.log
```

Touch the files and restart syslogd

Touch each file to create it, here is an example:

```
#cd /var/log
# touch gritd.packets gr.console gr.boot grinchd.log gr.conferrs
mib2d.log
```

Determine the PID (process ID) for the **syslog** daemon and restart it:

```
# ps -ax | grep syslogd

# kill -HUP <PID>
```

Optional task:
Step 5 is only for sites that had previously configured logging to **syslogd** or NFS.
New installations can go onto Step 6.

5 If you had previously configured your GRF to log messages to a directory other than `/var/log`, you changed settings in `/etc/grclean.conf` and `/etc/grclean.logs.conf` files. Go back into those files now and change the log directory.

Modify `/etc/grclean.conf` and `/etc/grclean.logs.conf` to reflect the new log directory.

The `/etc/grclean.conf` file specifies which log and dump files the **grclean** program compresses, archives, and deletes.

The `/etc/grclean.conf` file entries should look like the following:

```
###################################################################
# port card dump files.
###################################################################
hold=4
size=1
remove=y
local=y
logfile=/var/portcards/grdump.*
###################################################################
# cleanup our own log file, if necessary.
###################################################################
DEFAULTS
hold=2
local=y
size=10000
```

```
logfile=/var/log/grclean.log
```

The `/etc/grclean.logs.conf` file is used to set size limits on log files. Here are some sample entries:

```
*************************************************************************
* Log files that used to be archived by the /etc/{daily|weekly|monthly}
* scripts.
*************************************************************************
size=150000
logfile=/var/log/gr.console
size=11000
logfile=/var/log/gr.boot
```

6 Save all changes and reboot:

```
# grwrite -v
# reboot -i
```

7 Verify that the PCMCIA interface and device are up. The **csconfig** command returns information about both external ports:

```
# csconfig -a
             Slot 0: flags=0x3<UP,RUNNING>
               Attached device: wdc2
               Manufacturer Name: "Kingston Technology"
               Product Name: "DataPak 520"
               Function ID: 4 (PC card ATA)
               Assigned IRQ: 11
               Assigned I/O port1: 0x3d0-0x3df
             Slot 1: flags=0x5<UP,EMPTY>
```

If you specify the slot number (0 or 1), **csconfig** returns information about the specified slot. This example looks at the flash disk in slot B:

```
# csconfig 1
      Slot 1: flags=0x3<UP,RUNNING>
        Attached device: wdc1
        Manufacturer Name: "SunDisk"
        Product Name: "SDP"
        Additional Info1: "5/3 0.6"
        Function ID: 4 (PC card ATA)
        Assigned IRQ: 11
        Assigned I/O port1: 0x3d0-0x3df
```

8 To run a quick test, execute the **grconslog** command. This command verifies that logging is correctly configured, so if it runs, the PCMCIA installation has been performed correctly:

```
# grconslog
```

iflash command - caution !

The PCMCIA procedure uses the **iflash** command. The **iflash** command determines the geometry of an installed PCMCIA ATA Type-II or Type-III storage device, and can format the device for use in a GRF.

Caution: By default, **iflash** does not initialize a flash that already has a file system installed. The **-f** option overrides the check and forces any data on the target device to be overwritten. Use this option only if you wish to erase the existing contents of a flash device.

When you use **iflash** without **-f**, you are informed when there is a file system already on the device and reminded that you must use the **-f** option to overwrite it. Because of its “force” capability, use the **iflash -f** command with caution.

Update changes to grclean.logs.conf *(if needed)*

grclean is an internal program that compresses, archives, and manages dumps and log files, and saves them to a specified file name.

grclean manages and archives other system log files that had previously been maintained by the daily, weekly, and monthly scripts.including `acct`, `maillog`, `messages`, `daemon.log`, `cron`, `xferlog`, `access_log`, `error_log`, `ftp.log`, `kerberos.log`, `cli.log`, and `lpd-errs`.

Software release 1.3.11 changed the contents of the `/etc/grclean.logs.conf` file.

If you are upgrading to 1.4 from a 1.3.9 or earlier software release, the previous `/etc/grclean.logs.conf` file is renamed to `/etc/grclean.logs.conf.old`, and a new copy of `/etc/grclean.logs.conf` is installed. You may see the message describing this transfer if you are watching the console as **grfins** operates.

If you have never made any changes to `/etc/grclean.logs.conf`, the upgrade has no effect. However, if you did change `/etc/grclean.logs.conf` in the past, then you must now make those changes again as soon as possible after the 1.4 update procedure is finished. Cut and paste just your changes into the 1.4 version of the file, take care not to overwrite the new sections in the file.

During software upgrades, **grclean** appends `.old` to the `/etc` files which are replaced in the upgrade. This saves the current configuration so it can be applied to the new version of the file.

Assign system IP addresses - grifconfig.conf

IP routing requires IP addressing information for the interfaces on media cards as well as directly attached interfaces such as `de0` or `ef0`, the maintenance Ethernet interfaces, or `lo0`, the software loopback. Each interface configured on the GRF must be on a different subnet.

Use a UNIX editor to open the `/etc/grifconfig.conf` file. Here is a new file. The startup configuration script has already written the IP address, and netmask if you entered one, for `de0` to the file:

```
#
# name  address                netmask         broad_dest      arguments
#
de0    192.0.2.1           255.255.255.0   192.0.2.255     mtu 1024
lo0    127.0.0.1           255.0.0.0
#gl000  127.0.1.1
# configuration for iso addresses
#gf0xx  49.0000.80.3260.3260.3260.00 49.0000.80  -  iso

de0 204.122.132.33
```

You must remove the comment character at the beginning of the loopback `lo0` line if one is there, this interface must be active. Use commented lines to identify the interfaces you configure:

```
# Card 0 - HSSI
gs010  10.202.1.133    255.255.255.0
gs011  10.202.2.133    255.255.255.0
gs012  10.20.2.226     255.255.255.252
#
# Card 1 - ATM3
ga020  205.2.1.133
ga026  10.20.2.234      255.255.255.252
ga029  10.20.2.214      255.255.255.252
```

File format

The format for an entry in the `grifconfig.conf` file is:

```
name   address                 netmask       broad_dest    arguments
```

Interface name

The interface name is required. An interface name describes an interface in terms of media type, chassis number, chassis slot, and logical interface number.

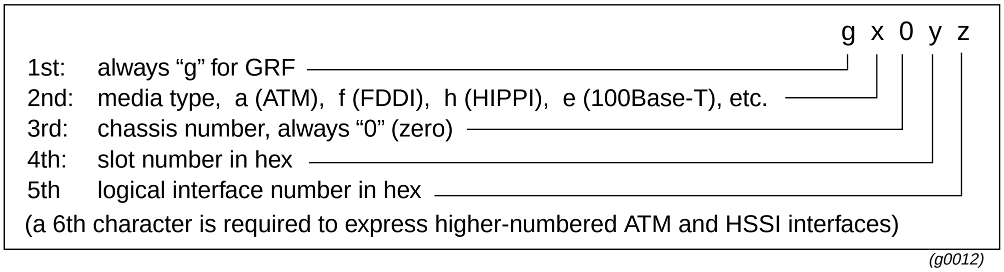


Figure 2-1. Components in the GRF interface name

Here are examples of the interface names for a card in slot 3:

- ATM: `ga030 through ga03ff`
- FDDI: `gf030, gf021, gf012, and gf003`
- HIPPI: `gh030` (only the slot # changes)
- HSSI: `gs030, gs021, gs0180`
- SONET OC-3c: `go030` (only the slot # changes)
- Ethernet: `ge030, ge031 .. ge037`

Note: All interface names are case sensitive ! Always use lower case letters when defining interface names.

IP address

An Internet Protocol (IP) address is required. The Internet address is the 32-bit IP address for the logical interface being specified. The address is entered in standard dotted-decimal (octet) notation: `xxx.xxx.xxx.xxx`.

Netmask (optional)

A netmask determines which part of an IP address represents the network and which part represent the host machine. The default netmask is 255.255.255.0.

The netmask is the 32-bit address for the logical IP network on the physical network to which the specific GRF or media card physical interface is attached. The netmask is entered in standard dotted-decimal (octet) notation. If no broadcast/destination address is supplied, a netmask is required. If a broadcast address is supplied, enter a dash (-) as a placeholder for the netmask column.

Broadcast / destination address (optional)

This address identifies a broadcast IP address for an Ethernet or FDDI interface or a destination address for a point-to-point ATM or HIPPI interface. Enter the broadcast or destination address in standard dotted-decimal (octet) notation. When a broadcast IP address is assigned to a logical interface, the netmask value is ignored. A dash (-) can be entered in the netmask column as a placeholder. When you configure a logical interface on a point-to-point media, an entry in the broadcast/destination field creates a point-to-point connection to that address. If you do not specify a broadcast address, you create a non-broadcast, multi access (NBMA) interface.

Argument field (optional)

The arguments field is currently used to specify MTU values on a per interface basis when the default MTU will not be used. Also, the field specifies ISO when an ISO address is being added to an interface's IP address. If you want to use the arguments field and are not using a broadcast or destination address, enter dashes as the address placeholder. Refer to the **ifconfig** man page for a description of argument options.

You can specify a different MTU for each logical interface. In `/etc/grifconfig.conf`, specify the MTU value as `mtu` *xyz*.

Create a loopback alias

You must create an alias IP address for the loopback (`lo0`) interface. Specify the IP addresses for the `lo0` interface in `/etc/grifconfig.conf` as shown in the following example:

```
#name  address    netmask
lo0   127.0.0.1   255.0.0.0
lo0   x.x.x.x     255.255.255.255
```

where `x.x.x.x` corresponds to the IP address that you want to be aliased to `lo0`.

Default MTUs

The default MTUs for GRF media are:

HIPPI:	65280 bytes
FDDI:	4352 bytes
ATM OC-3c:	9180 bytes
10/100Base-T:	1500 bytes

The default MTUs for framing protocols on HSSI, and SONET cards are:

Frame Relay	4352 bytes
HDLC	4352 bytes
Point-to-Point Protocol	1500 bytes

MTU discovery facility

GRF software supports MTU Discovery, which dynamically sets the MTU size per TCP connection (Path MTU Discovery, RFC 1191). MTU sizes are generally selected at the host end of the route. This is accomplished by turning on the host's MTU discovery facility and allowing the host to send packets.

In effect, the discovery facility tells the router not to fragment, but to advise the host when the packet size is larger than the given path can handle. This allows the host to discover the largest packet which the most restrictive of the media components within the same path can handle. Once "discovered", the host then sends only packets in sizes matching the reported maximum, and packets are not fragmented.

Define an alias or secondary address

An alias or "secondary" address can be assigned to a logical interface by specifying two entries, each with a different IP address, in the `/etc/grifconfig.conf` file. An alias enables the same interface to be in more than one logical IP subnet. This may be useful for some dynamic routing protocols to make a network appear as a full mesh.

This example assigns an alias to a SONET interface:

```
#name  address    netmask
go070  192.0.2.1  255.255.255.0
go070  192.0.3.1  255.255.255.0
```

The first entry is the primary IP address (`192.0.2.1`), the second is the alias or secondary address (`192.0.3.1`).

This is an example for an ATM interface:

```
#name  address    netmask
ga060 192.0.4.1 255.255.255.0
ga060 192.0.5.1 255.255.255.0
```

Note: For ATM, a unique VPI/VCI must also be added in the `/etc/gratm.conf` and `/etc/grarp.conf` files.

Defining an ISO address (IS-IS)

IS-IS is supported on ATM OC-3c, Ethernet, FDDI, HSSI, and SONET media cards. Configuration information and IS-IS examples are in the *GRF GateD Manual*.

To configure the IS-IS protocol, you must also assign each interface an ISO address in `/etc/grifconfig.conf`. The ISO address entry is in addition to the IP address entry.

The dash before "`iso`" is required, here is the entry syntax and an example:

```
#interface_name <iso_address> <iso_area> - iso
gf030 xxx.xxx.xxx.xxx 255.255.255.0 - mtu 4100
gf030 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

Change IP address without card reset

You can change the IP address of a logical interface without needing to reset the media card. First, edit the `/etc/grifconfig.conf` file and change the interface's IP address. Then use an **ifconfig** command to create an interface with the new IP address. Given:

```
#/etc/grifconfig.conf
#name  address        netmask
ge030  200.200.200.1  255.255.255.0
```

Edit the file to change interface `ge030`'s IP address to 200.1.1.5:

```
# vi grifconfig.conf

#name  address        netmask
ge030  200.1.1.5  255.255.255.0
```

Use the **ifconfig** command to create the interface:

```
# ifconfig ge030 200.1.1.5
```

Check if the new address is configured and test its connection with these commands:

```
# netstat -rn
# ping 200.1.1.5
```

Install the configuration

To install a new `/etc/grifconfig.conf` file, either re-run the **grifconfig** program:

```
# grifconfig gx0yz
```

Or use the **grreset** ***slot_number*** command to reset the media card:

```
# grreset slot_number
```

Change GRF hostname

The factory-preset hostname (`grf.ascend.com`) must be changed on each GRF.

When the GRF is first booted, you should have entered a new host name when prompted by the startup configuration script. If you did not, you can re-run this script using **config_netstart**. How to re-run the script is described in the “Management Tasks” chapter.

Alternatively, you can change the hostname in the `/etc/netstart` and `/etc/hosts` files. Start the UNIX shell and use an editor to open each file.

`Change`

```
super> sh
# cd /etc
# vi  /etc/netstart
```

Edit the line: hostname=grf.ascend.com

to read: hostname=new.host.com

Save the file and open `/etc/hosts`.

```
# vi /etc/hosts
```

Edit the line: ###.###.###.### grf.ascend.com

to read: ###.###.###.### new.host.com

Save the file and execute the **hostname** command:

```
# hostname new.host.com
```

Enable host telnet access - /etc/ttys

Each instance of `ttypX` allows one remote telnet session. Out of the nine entries available, update the number of entries in the `/etc/ttys` file that your site will need.

Use a UNIX editor to change the `ttypX` settings in the `/etc/ttys` file. The file lines look like this:

```
ttyp0      none   network
ttyp1      none   network
ttyp2      none   network
ttyp3      none   network
   .
   .
   .
```

They should be changed to this:

```
ttyp0      none   network secure
ttyp1      none   network secure
ttyp2      none   network secure
ttyp3      none   network secure
   .
   .
   .
```

Note that user “`netstar`” can always log in to the GRF.

Link0 and link1 flags

LINK0 and LINK1 flags are reported in **ifconfig -a** output. That **ifconfig** command verifies the connection status of individual logical interfaces. LINK0 and LINK1 are different from other “links” such as links seen in **netstat** output.

The kernel asserts the LINK0 flag on a logical interface when the card detects continuity out to the attached device. When LMI protocol is running, Link0 also indicates that LMI is up.

The kernel asserts the LINK1 flag after **filterd** has initialized and conducted a handshake with the interface. Filters may or may not be assigned the interface.

You should always see LINK1. If you do not see LINK0, the interface may or may not be able to send packets. Using an **ifconfig** command, it is possible to manually set LINK0. However, a manual set is not recommended because an underlying problem usually prevents LINKO from being asserted.

IP routing options

Static-only routing

When a GRF router is configured for static-only IP routing, no dynamic routing protocols are being run and all routes are configured manually.

Static routes are configured by either:

– editing the `/etc/grroute.conf` file and saving its contents with the **grwrite** command (permanent, changes are preserved across reboots)

– using the **route add** command (changes are lost at reboot)

If a GRF will do dynamic routing (GateD), *but* will also connect to a router using static routing, you must enter routes for the static router with a GateD Static statement. GateD only recognizes those routes it collects. Manually-entered static routes added via **route add** or in `/etc/grroute.conf` are lost when GateD removes them during its normal maintenance of the master route table. The GateD Static statement preserves static routes.

grroute.conf file

The `/etc/grroute.conf` file provides the GRF with static routing information.

A specific route's data is entered on one line in three columns. All address and mask entries are in standard dotted-decimal (octet) notation. The format of the file is:

```
#Destination    Destination      Gateway /
#address        netmask          next hop
```

- Destination address is the IP address of the target destination host or network
- Destination netmask holds the netmask for the network route destination (the destination netmask for host routes must be 255.255.255.255)
- Gateway/next hop holds the next-hop address to which data is forwarded on its way to the target destination address

Default route

The default route is specified as 0.0.0.0 (or `default`), with a netmask of 0.0.0.0.

Error checking

No check is made to ensure that the next-hop address is actually reachable via an attached network.

Putting grroute changes into effect

Changes to `/etc/grroute.conf` take effect only after the file is reloaded and the media card(s) reset. Use the **grreset** command.

route command

The UNIX **route add** command adds a route to a destination IP address, but the changes are lost at system reboot. The basic format of a **route** command is:

```
# route add destination
```

Refer to the **route** man page for information and other options.

Static route example

In the example below, Host A wants to ping Hosts B and X, and Host B wants to ping Host A:

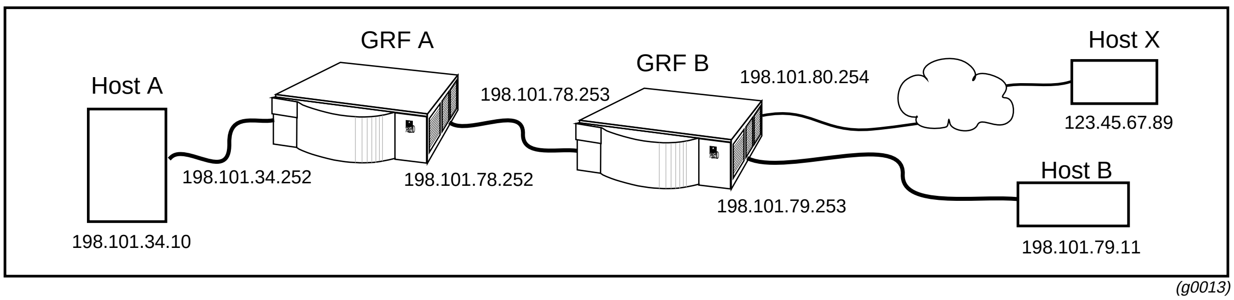


Figure 2-2. Illustration for static routing configuration

These are the configuration options when a network is using only static routing, and no dynamic routing. Configure routes using commands or use GateD.

For GRF A, these entries in a `grroute.conf` file:

```
198.101.79.0  255.255.255.0 198.101.78.253      (to B)
123.45.67.89  255.255.255.255 198.101.78.253    (to X)
```

are equivalent to the following **route add** commands:

```
# route add 198.101.79.0 -netmask 255.255.255.0 198.101.78.253
# route add -host 123.45.67.89 198.101.78.253
```

For GRF B, this entry in a `grroute.conf` file:

```
198.101.34.0 255.255.255.0 198.101.78.252      (to A)
```

is equivalent to the following **route add** command:

```
# route add 198.101.34.0 -netmask 255.255.255.0 198.101.78.252
```

See the Static Statements subsection in the *GRF GateD Manual* for more information about configuring static routes.

Displaying static route tables

Use this command to check the number of entries in the route table, the number is returned:

```
# netstat -rn | wc -l
     1866
```

The recommended way to view static route tables per media card is to use the **grrt** command. To view the routing table for the media card in slot 1, enter:

```
#  grrt -p 1 -S
```

Here is an example of the type of information returned:

```
Route            Netmask     Metric  NextHop     Interface    Type
default                         0    0.0.0.0       inx 0    UNREACH
0.0.0.0       255.255.255.255  1    0.0.0.0       inx 0      DROP
10.20.1.0     255.255.255.0    17   0.0.0.0       ge034      FWD
10.20.1.133   255.255.255.255  16   0.0.0.0       ge034      LOCAL
10.20.1.255   255.255.255.255  15   0.0.0.0       ge034      BCAST
10.20.2.0     255.255.255.0    21   10.205.1.150  ga00f0     FWD
10.205.1.0    255.255.255.0    20   0.0.0.0       ga00f0     FWD
10.205.1.133  255.255.255.255  19   0.0.0.0       ga00f0     LOCAL
10.205.1.255  255.255.255.255  18   0.0.0.0       ga00f0     BCAST
10.205.3.0    255.255.255.0    24   0.0.0.0       ga00f1     FWD
10.205.3.133  255.255.255.255  23   0.0.0.0       ga00f1     LOCAL
```

IP source routing

In IP source routing, the source specifies (via addresses in the IP header) the path a datagram will take to the destination address. There are two kinds of source routing, strict and loose.

Strict source routing specifies an exact route. The path can only be through the router addresses provided by the source.

In loose source routing, a source specifies intermediate hops on the route. This option lets intermediate routers determine the best connection between the intermediate hops with the option of avoiding specific addresses. Note that loose source routing can result in a slow path, and routing performance can be affected.

By default, source routing is disabled on the GRF.

To enable source routing on, enter this command at the UNIX prompt:

```
# sysctl -w net.inet.ip.forwsrcrt = 1
```

To disable source routing, change the value of the setting to zero (0):

```
# sysctl -w net.inet.ip.forwsrcrt = 0
```

Directed broadcast forwarding

By default, the GRF does not forward directed broadcast datagrams. There are two ways to configure directed broadcast so that it is permanently enabled:

– edit the `/etc/rc.local` file and insert the `sysctl` directed broadcast line

– enable the `Forward_Directed_Bcast_Pkts` field in the System profile

Use sysctl entry in /etc/rc.local

Open the `/etc/rc.local` file with a UNIX editor and go to the end of the file.

```
# cd /etc
# vi rc.local
```

Above the “exit 0” line, enter a new line:

```
        sysctl -w net.inet.ip.fwdirbcast=1
        exit 0
```

Leave no space on either side of the = sign. With the option line set as `=1`, directed broadcast is enabled at each system boot. Save the file and exit. If later you want to disable directed broadcast, change the line so that set `=0`.

Note: you can temporarily enable directed broadcast by executing a **sysctl** command. This method does not survive system reboot:

```
# sysctl -w net.inet.ip.fwdirbcast=1
```

Verify the current setting, use this command:

```
# sysctl -w net.inet.ip.fwdirbcast
     net.inet.ip.fwdirbcast = 1
```

Enable field in System profile

You can enable the GRF to forward directed broadcast packets using the field in the System profile. Here is the process:

```
super> read system
SYSTEM read
super> list
os-level = 1.4.12R.4
hostname = box1.anysite.com
chassis = GRF1600
ip-address = 206.146.160.186
netmask = 0.0.0.0
default-route = 0.0.0.0
hippi-ifield-shift = 5
enable-congest = disabled
num-slots = 16
rmb-load-path = /usr/libexec/portcards/rm.run
rmb-dump-config = 4
physical-memory = 320
hardware-revision = "Not Available"
chassis-revision = 1
xilinx-revision = 8
num-fans = 2
num-pwr-supply = 1
Forward_Directed_Bcast_Pkts = disabled
super> set Forward = enabled
super> write
SYSTEM/ written
```

IP multicast

IP multicast is supported on the GRF Ethernet and FDDI media cards.

Route table lookup

The GRF performs a hardware-assisted full route table lookup that can be accomplished in less than 3 microseconds, even when the route table contains 150,000 routes. For most networks, the next hop is found in less than 1 microsecond. This is 100 times faster than software-driven route table lookups.

Selective packet discard (SPD)

Selective packet discard (SPD) can be enabled on the ATM OC-3c (ATM/Q), FDDI/Q, Ethernet, HSSI and SONET media cards to ensure that dynamic routing packets are transmitted on the media in the presence of a sustained high volume of data packets. During high traffic volumes, data packets are discarded in a rate that favors dynamic routing packets. Specifying a congestion and discard threshold is described in the media card configuration guides in this manual.

Packet discard is regulated by reserving buffers for dynamic routing packets. This gives the operator control over the point at which congestion management begins to discard data packets. A user-configured threshold defines the percentage of buffers to reserve for dynamic routing packets. The selective packet discard threshold is configured in the Card profile.

Each of the media card configuration chapters has a section discussing useful thresholds and how to configure the threshold in the Card profile. Media cards that support SPD also support Controlled-Load class filtering. Refer to the “Integrated Services” chapter in this manual for more information.

Precedence handling

Precedence handling prioritizes delivery of dynamic routing update packets, even when the transmitting media card is congested. The GRF dynamic routing agent sets a precedence value in the internal packet header of the dynamic routing update packets it generates, which communicates to the media card a high-priority status for the packet. The media card maintains a user-configurable threshold of transmit buffers that always remain available for high-priority traffic, ensuring that dynamic routing update packets are forwarded during congested conditions.

Precedence field

With selective packet discard enabled, the available buffer pool is managed as two pools, one for those with the "precedence field" set (high priority) and one for low priority data. Therefore, as the packets are taken off the switch, the buffer pools can be set up so that high priority packets will always find a buffer available, and the low priority packets will be dropped.

The precedence field is set in the IP packet header by GateD on dynamic routing packets or by filters configured to set this field on incoming data that matches any filter definition

Most dynamic routing packets sourced by the GRF have the precedence field set. This results in priority handling on the outbound (transmit) side of the media card in that a buffer is always made available for these packets as the data is read off the switch or communications bus. The media card starts discarding "low priority" packets before it completely runs out of buffers.

ARP on the GRF

This is a brief overview of ARP implementation on the GRF.

ARP processing on media cards

The media card processes and sends the ARP requests, not the control board. The control board is not involved in ARP for any of the cards.

Ping opposite interface to invoke ARP

Given that two GRF routers are connected across an Ethernet hub with ports 0 and 1, respectively, configured on the connecting Ethernet cards. A ping is sent from port 0 to port 1.

ARP is resolved on both routers. If port 1 is IDLE for 600 seconds, the TTL expires and the ARP cache times out. A second ARP request should not automatically go out.

Ping to a broadcast address

Pinging to a broadcast address does not place an ARP entry in cache. This is normal. Since you are broadcasting, the hardware address is automatically `ff:ff:ff:ff:ff:ff`, hence, no ARP request. There is no need to get a specific hardware address, everyone should receive it.

Proxy ARP support

Proxy ARP is supported on GRF broadcast media, the FDDI and Ethernet cards.

Proxy ARP enables a router to answer an ARP request on one of its networks that is actually destined for a host on another of the router's networks. This leads the sender of the ARP request into thinking that the router is the destination host, when in fact the destination host is "on the other side" of the router. The router acts as a proxy agent for the destination host, relaying packets to it from the other hosts.

Use grarp -f to process ARP entries

Use the **grarp -f /etc/grarp.conf** command to have **grarp** re-read the configuration file, process entries, and send ARP information to media cards.

Use grarp -i to display ARP information

To display ARP information, use the **grarp -i *interface hostname*** command. Other **grarp** options are described in the *GRF Reference Guide*.

tcpdump does not display ARP

The **tcpdump** utility does not display ARP information. This is normal. **tcpdump** acts only on packets that are routed. ARP packets are not routed.

Configure SNMP (option)

By default, SNMP is configured to process only GET requests for the “public” community. All configuration of SNMP is done via the `/etc/snmpd.conf` file. Instructions for configuring the more common portions are described here.

Configure SNMP subagents

The SNMP agent can be configured to be used with multiple subagents. By default, the agent is configured to operate with only one subagent. This subagent is used to provide support for MIB-II as defined by RFC 1213. To configure a subagent, add an ALLOW entry that specifies the subagent identifier to the `/etc/snmpd.conf` file:

```
# Subagent for handling MIB-II (RFC 1213) information
ALLOW        SUBAGENT 1.3.6.1.4.1.1080.1.1.1
             WITH OTHER PASSWORD
             USE 15 SECOND TIMEOUT
```

The ALLOW statement specifies that the SNMP agent will wait up to 15 seconds for a response from the MIB-II subagent before attempting to make a new connection.

Configure community names

The SNMP agent can be configured to use community names for various types of operations. By default, the agent is configured to handle only GET requests using the “public” community name. Normally, a separate community name is used to allow a network manager to set any objects that can be written via SNMP. This is illustrated in the following examples from `/etc/snmpd.conf`:

```
# Default community name
COMMUNITY    public
             ALLOW GET OPERATIONS
             USE NO ENCRYPTION

# Network manager community name
COMMUNITY    netman
             ALLOW SET OPERATIONS
             USE NO ENCRYPTION
```

In this example, all network managers using the “public” community name are allowed to request the MIB information. However, only the network managers using the “netman” community name are allowed to change the MIB information.

If you replace the SET keyword with the ALL keyword, all network managers can perform both GET and SET operations via SNMP.

Configure system contact information

The sysContact object is defined within the system group of RFC 1213. By default, the sysContact object returns a NULL string. To configure the system contact information, add an INITIAL entry to the `/etc/snmpd.conf` file in which up to 256 bytes of information are specified to describe the system contact person. Here is an example:

```
# Define the system contact person
    INITIAL    sysContact        "Site Guru
    email: <site.guru@site.com>
    Phone: (xxx) xxx-xxxx"
```

Configure system name information

The sysName object is defined within the system group of RFC 1213. The sysName object always returns the information given by the **hostname** command. This information is configured by adding an entry to the `/etc/hosts` file as shown below:

```
# Host Database
    206.146.164.20  workstationX.site.com
```

Configure system location information

The sysLocation object is defined within the system group of RFC 1213. By default, the sysLocation object returns a NULL string. To configure the system location information, add an INITIAL entry to the `/etc/snmpd.conf` file in which up to 256 bytes of information are specified to describe the system location. An example is given below:

```
# Define the system location
    INITIAL    sysLocation    "Main Computer Room
    10250 Valley View Road
    Minneapolis, MN  55344"
```

Configure trap management

A trap is an SNMP message sent from a managed system to a management station when a particular event occurs. The message indicates the type of event, and can also contain the values of certain variables in the MIB.

The SNMP daemon can be configured to send trap information to one or more network management stations. By default, the trap information is not sent to any network management stations. To configure the SNMP agent to send traps to a network management station, add a MANAGER entry to the `/etc/snmpd.conf` file that specifies either the name or IP address of a network management station to which the trap information should be sent. Here is an example:

```
MANAGER     workstationX
SEND ALL TRAPS
```

Put configuration changes into effect

The SNMP agent and all subagents must re-read the be notified of any configuration changes before they can be put into effect. The notification process requires the operator to issue the -HUP signal to the SNMP agent and each of the subagents. To do so, execute the following commands from the UNIX shell:

1 Determine the process identifier (*`process id`*) for the current **snmpd** process, enter:

```
# ps -ax|grep snmpd
```

The process identifier is returned:

```
26053 p2  S+     0:00.05 grep snmpd
127 co- S      1:59.55 snmpd /etc/snmpd.conf /var/run/snmpd.NOV
```

2 Send the -HUP signal to the current **snmpd** process to cause **snmpd** to re-read the `/etc/snmpd.conf` file and restart. Enter:

```
# kill -HUP 127
```

3 Determine the process identifier (*`process id`*) for the current **mib2d** process, enter:

```
# ps -ax|grep mib2d
```

The process identifier is returned:

```
28053 p2  S+     0:00.09 grep mib2d
142 co- S      1:59.55 mib2d /etc/mib2d.conf /var/run/mib2d.NOV
```

4 Send the -HUP signal to the current **mib2d** process which will cause **mib2d** to re-read the `/etc/snmpd.conf` file and restart. Enter:

```
# kill -HUP <process id>
```

15 second time-out entry

The `snmpd.conf` file template contains the ALLOW entry. This entry should not be removed because it gives **snmpd** a 15 second time-out for responses coming from **mib2d** and keeps **snmpd** active should **mib2d** hang.

Alternatives to SNMP gets of route tables

When a GRF is maintaining a large route table (50K entries), and an SNMP Management Station sends a “return all known routes” request, **mib2d** consumes major memory resources trying to process the request. If GateD is running, please view route tables using the GateD State Monitor (GSM) tool. Establish a GSM session and use this command:

```
gsm> show ip all
```

Otherwise, look at the route table for a media card in a specified slot by entering:

```
# grrt -p slot -S
```

Disabling SNMP and mib2d daemons

You can disable the SNMP and mib2d daemons by renaming them to a user-defined filename and then using the **grsite *filename*** command to archive the renamed files.

```
# cd /usr/sbin
# move snmpd test1
# grsite test1
# move mib2d test2
# grsite test2
```

After the files are archived, you must reboot the GRF.

```
# shutdown -r now
```

SNMP support

This section describes the areas of SNMP support currently provided on the GRF.

TCP/IP Network Management Support (RFC 1213)

No direct support is provided for setting any of the read-write objects defined by RFC 1213 via SNMP. However, each of the following read-write objects defined by the system group can be set at the site through the normal GRF configuration operations:

- `sysContact` - contact person for this node.
- `sysName` - administratively-assigned name for this node
- `sysLocation` - physical location of this node

The GRF provides read-only support for the following MIB information defined by RFC 1213:

- `system`
- `interfaces`
- `ip, icmp`
- `tcp`
- `udp`
- `snmp`

The GRF provides read-only support for the following MIB information under the transmission group defined by RFC 1213:

- `Frame Relay DTE MIB` (RFC 1315)
- `PPP/LCP MIB` (RFC 1471)
- `PPP/IP MIB` (RFC 1473)
- `FDDI MIB` (RFC 1512)
- `HIPPI MIB` (HIPPI end-point MIB)
- `HIPPISW` (experimental MIB for HIPPI switch)

The GRF does not currently support the following groups:

- `address translation` (deprecated in MIB-II)
- `egp`
- `oim`
- `transmission.frame-relay.frDlcmiTable`
- `transmission.frame-relay.frErrTable`

Enterprise MIB support

The GRF provides read-only support for each of the groups defined by the enterprise MIB located at `/usr/share/mibfiles/netstar.mib`.

- `grChassis`
- `grFDDI4`
- `grATMV1`
- `grATmUNI`
- `grHIPPI: HIPPI-MIB`

(HIPPI end-point MIB) is similar to an internet draft MIB definition. `HIPPISW-MIB` (experimental MIB for a HIPPI switch) contains `hippiswShiftCount`, `hippiswPortNumber`, `hippiswPortTable`, `hippiswLATable`.

Enterprise TRAP support

The GRF support for TRAPs includes both generic traps and enterprise-specific traps. Generic TRAPs are defined by `/usr/share/mibfiles/netstar.mib` and are listed below:

- `coldStart`
- `warmStart`
- `linkDown`
- `linkUp`
- `snmpEnableAuthenTraps`

Enterprise-specific TRAPs are defined by `/usr/share/mibfiles/netstar.mib` and are listed below:

- `grPowerSupplyFailure`
- `grOverTemp`
- `grFanFailure`
- `grCardDown`
- `grCardUp`
- `grSONETLossOfFrame`
- `grSONETLossOfSignal`
- `grSONETPathLossOfPointer`
- `grSONETLossOfPointer`
- `grSONETLineAlarmIndicationSignal`
- `grSONETSTSPathAlarmIndicationSignal`
- `grSONETPathAlarmIndicationSignal`
- `grSONETLineRemoteDefectIndication`
- `grSONETVTPathAlarmIndicationSignal`
- `grSONETLineRemoteDefectIndication`
- `grSONETVTPathRemoteDefectIndication`
- `grSONETTCLossOfCellDelineation`
- `grSONETLineRemoteDefectIndication`
- `grSONETVTPathRemoteDefectIndication`
- `grSONETTCLossOfCellDelineation`
- `grAtmPVCUp`
- `grAtmPVCDown`

MIB locations

All MIBs supported by the GRF agent are installed in `/usr/share/mibfiles`. They are:

- `rfc1213.smi`	MIB-II
- `rfc1227.smi`	SMUX MIB
- `rfc1315.smi`	Frame Relay DTE MIB
- `rfc1471.smi`	PPP MIB
- `rfc1473.smi`	PPP MIB
- `rfc1512.smi`	FDDI MIB
- `rfc1573.smi`	Extended Interface MIB
- `netstar.smi`	Enterprise MIB
- `hippimib.smi`	HIPPI End-point MIB
- `hswmib.smi`	HIPPI Switch MIB

Enable GateD (option)

Ascend’s GateD handles dynamic routing with a routing database built from information exchanged by routing protocols. GateD supports the use of the following dynamic routing protocols:

- Border Gateway Protocol (BGP)
- Routing Information Protocol (RIP)
- Open Shortest Path First (OSPF) protocol
- Intermediate System-to-Intermediate System (IS-IS) protocol.

GateD is a modular software program consisting of core services, a routing database, and protocol modules that support the multiple routing protocols listed above. GateD allows the network administrator to configure routing policy on the GRF through import/export statements that control learning and advertising (or redistributing) of routing information by individual protocol, source and destination autonomous system (AS), source and destination interface, previous hop router, and specific destination address.

On the GRF, trace and log files generated by GateD and saved on a local file system must be limited to a total of 500,000 bytes. Please see the `/etc/gated.conf` template for recommended trace file sizes and options.

Create and edit gated.conf

Configure GateD for dynamic routing:

1 Log in as `root`.

2 Use the `/etc/gated.conf` section in the *GRF GateD Manual* to help build your `/etc/gated.conf` file, then edit for site needs.

There are many options and parameters to specify for each type of GateD statement. Remember that the configuration statements must appear in the specified order. An out of order statement causes an error when the file is parsed.

Start the dynamic routing daemon

Changes to `/etc/gated.conf` after first-time installation take effect only after GateD rereads its configuration file. Use the **gdc** command to ensure GateD rereads its configuration file, **gdc** is documented with the GateD information in the *GRF GateD Manua*l.

```
# gdc reconfig
```

Use this command to start GateD:

```
# gdc start
```

If GateD has not been started, you will get a message to that effect.

Equal Cost Multi-path (ECMP)

The Equal Cost Multi-path (ECMP) feature provides an ability to efficiently modulate traffic to destination networks. With ECMP enabled, multiple gateways for destination network or host prefixes (addresses) can be legally installed in the GRF route table. This release supports a maximum of eight gateways per ECMP group. Rather than use a single "best" route, ECMP routes packets toward a destination network by splitting the packet load between different, but similar, paths.

The diagram below shows a simplified ECMP group example. The GRF1 router is running ECMP. If ECMP is statically configured, ECMP routes are created by configuring entries in `/etc/grroute.conf`. If ECMP is turned on in the `/etc/gated.conf` file, GateD learns the routes and allows multiple gateways to be assigned to a single source address. The R300 router is the gateway for an available but unequal path.

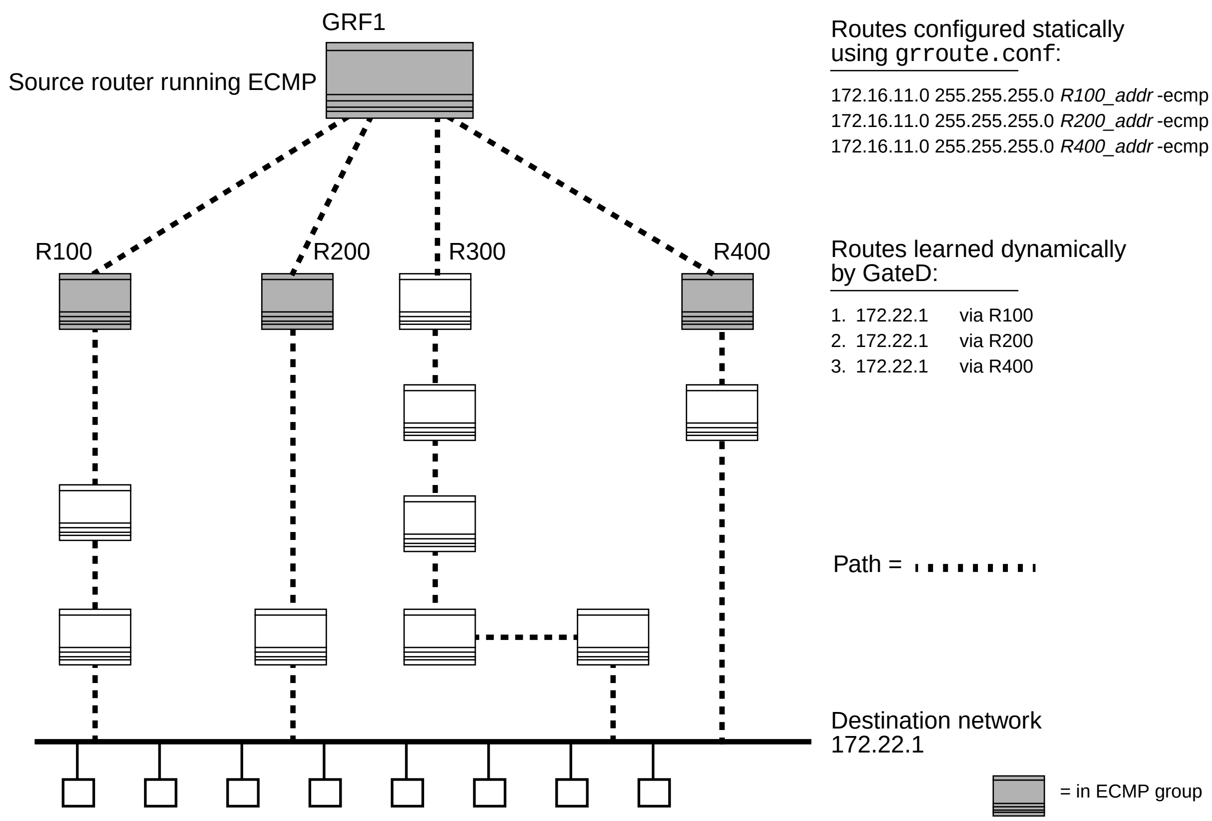


Figure 2-3. Example of alternate ECMP routes

For each packet, a determination mechanism selects the nexthop gateway from the ECMP group. The current determination mechanism selects from the group based on a source/destination hash.

The hash is fixed, and, for most cases, provides a reasonable, equal distribution of traffic. The hash system uses a core application to provide the fastest processing possible on a per-packet basis. This method ensures packets from a given source arrive in order to a given destination.

GateD support

GateD provides these services for ECMP:

– a configuration option to enable/disable dynamic ECMP (default is disabled)

– the ability to insert multiple equal routes into the kernel

– internal OSPF protocol support

Support includes equal cost multipath prefixes learned via the OSPF and OSPF_ASE protocols. Also, I/CBGP resolves prefixes to multiple gateways if the nexthop resolving protocol is OSPF or OSPF_ASE.

The ECMP parameter is available in a Definition Statement:

```
multipath { on | yes | off | no }  ;
```

The parameter enables/disables the installation of multiple gateways for network or host prefixes into the kernel route table. The default is off.

The **on** option is the same as **yes**, and enables GateD to install multiple routes for a single source with the same destination but different nexthops.

The **off** option is the same as **no**, and means that each route GateD installs in the kernel will have a unique destination and nexthop.

Please refer to the "*GRF GateD Manual*." More information is available in the Definition Statement description.

Dynamic ECMP configuration

Enable dynamic creation of ECMP routes in `/etc/gated.conf` by including either of these statements:

```
multipath on ;
```

or

```
multipath yes ;
```

Static ECMP configuration

Enter one line for each destination nexthop (gateway) in the `/etc/grroute.conf` file.

The `-ecmp` parameter is optional only for the source router's first entry, it is required for the rest of the entries:

```
# source_addr  netmask  next_hop_addr -ecmp
172.16.11.0 255.255.255.0 R100_addr [-ecmp]
172.16.11.0 255.255.255.0 R200_addr -ecmp
172.16.11.0 255.255.255.0 R400_addr -ecmp
```

The source GRF router running ECMP is 172.16.11.0.

Checking ECMP routes

To verify that routes have been installed in the kernel as ECMP routes, use the **netstat -rn** command. The “E” in the flags field is the ECMP identifier. Note that destination addresses have been assigned multiple gateways:

```
# netstat -rn
Routing tables


Internet:
Destination        Gateway            Flags     Refs      Use  Interface
10.0.0.82          10.0.0.82          UH          0         0  lo0
10.0.0.110         10.8.1.110         UGHE        0         0  go0a0
10.0.0.110         10.8.2.110         UGHE        0         0  go0b0
                        Method:CRC16
10.0.0.176         10.8.1.110         UGHE      693       692  go0a0
10.0.0.176         10.8.2.110         UGHE        0         0  go0b0
10.0.0.176         10.8.3.177         UGHE        0         0  go0d0
                        Method:CRC16
10.0.0.177         10.8.1.110         UGHE        0         0  go0a0
10.0.0.177         10.8.2.110         UGHE        0         0  go0b0
10.0.0.177         10.8.3.177         UGHE        0         0  go0d0
                        Method:CRC16
10.1.5/24          10.8.1.110         UGE         0         0  go0a0
10.1.5             10.8.2.110         UGE         0         0  go0b0
10.1.5             10.8.3.177         UGE         0         0  go0d0
                        Method:CRC16
10.1.6/24          10.8.1.110         UGE         0         0  go0a0
10.1.6             10.8.2.110         UGE         0         0  go0b0
10.1.6             10.8.3.177         UGE         0         0  go0d0
```

Authentication options

You can set the GRF as a client of an authentication program running on a remote server. The current options are:

– TACACS+
– RADIUS
– securID

The next sections describe the functionality of these authentication systems and the steps needed to configure the GRF as a client.

TACACS+ (option)

The Terminal Access Controller Access System (TACACS) runs on a remote machine and is used to validate logins on the GRF.

GRF client-side implementation

This section briefly explains what happens on the client side of the model when a user logs into the router.

Logging into the GRF requires the user to enter a user name. The user name is used to look up the password file entry for that user. If no password file entry is found, the user is denied access. When an entry is found, the authentication method specified for this user is now retrieved.

This authentication method maps to a class in the `/etc/login.conf` file.

A class definition must exist for TACACS+ in `/etc/login.conf`. If such a class does not exist, the user is denied access. If no authentication method is specified for the user in the password file, the `default` class is used for authentication.

If the authentication class is defined in `/etc/login.conf`, then the appropriate `/usr/libexec/login_xxx`, where `xxx` is the value of the `auth=` field defined for the class.

For example, if TACACS+ is the authentication method defined for a user in the password file, then there must exist a `tacacsPlus` class in the `/etc/login.conf` file. The specifics of the class definition are described below. If the `tacacsPlus` class has the `auth=` field set to `tacacsplus`, then `/usr/libexec/login_tacacsplus` is executed to validate the user.

Configuration steps on the GRF client

The example described here configures an account for the user “admin1”.

1 Configure the `admin1` user account to use the special TACACS+ class for authorization.
Use **vipw** to edit the user account to look similar to the line below :

```
admin1:<encrypted passwd>:<uid>:<gid>:tacacsPlus:0:0:
<clear text name>:<home dir>:<shell>
```

All entries enclosed in < > must be filled in appropriately. You can leave the `<encrypted passwd>` field empty until the first login. Set a password using the **passwd** command after the first login.

2 Open the `/etc/login.conf` file and make sure the class `tacacsPlus` exists with tacacsPlus defined as the authorization protocol in the `auth=` field.

Enter the remote server's IP address in the `tacacs-server=` field.

```
tacacsPlus:\
    :auth=tacacsplus:\
    :tacacs-server=x.x.x.x:\
    :tc=default:
```

3 Check to make sure the following lines are in the GRF `/etc/services` file:

```
# TACACS+ server
tacacs         49/udp        # Tacacs Server
```

Note: When you modify the `/etc/services` file, those changes will get saved to flash memory when you do a **grwrite**.

On RMS node systems, use the **grc** command to save and archive your changes to `/etc/services`.

However, subsequent software upgrades will install the new release version of `/etc/services`, overwriting any changes you may have made. Please be sure to record your changes to that file as you will need to add them again when you upgrade.

4 Finally, configure the remote TACACS server:

– make sure the server knows the client's IP address.

– make sure each GRF user is entered in the server's configuration file.

Check that the remote TACACS+ server is up and running.

Set RADIUS authentication (option)

The current software release supports the client side of RADIUS (Remote Authentication Dial In User Service) as described in the IETF Draft of February, 1996. GRF 400 and GRF 1600 systems and GRF and GR-II systems using RMS nodes can establish a RADIUS client.

Once configured, the client GRF sends authentication requests to a RADIUS server and allows access to the GRF based on the server response.

How RADIUS works

This section briefly explains what happens on the client side of the model when a user logs into the GRF router.

In a system without RADIUS, logging in to the GRF requires the user to first enter a user name. This user name is used to look up the password file entry for that user. If no entry is found for the user, the user is denied access. If a password entry is found, the user is prompted to supply a password.

In a system using RADIUS, the user name is again used to look up an entry in the password file. In this case, the entry for a valid user has an assigned authentication method field rather than solely a password.The authentication method is retrieved from the password file and is required in a second level of validation to map to a class definition in the `/etc/login.conf` file.

If a class definition does not exist for a particular authentication method, the user is denied access. (In a system using RADIUS, if no authentication method is specified for the user in the password file, the default class is used for authentication.)

If RADIUS is the authentication method defined for a user in the password file, then there must exist a RADIUS class in the `/etc/login.conf` file. The specifics of the class definition are described below. If the RADIUS class has the `auth=` field set to `radius`, then the `/usr/libexec/login_radius` program executes to validate the user. This program communicates with the server and prompts the user for a passcode.

Configure the GRF RADIUS client

To configure an account for user `bob` on the GRF client, follow these steps:

1 Edit `/etc/login.conf` to define the RADIUS server:

```
radius:\
    :auth=radius:\
    :auth-ftp=reject:\
    :radius-server=radius-server.domain.com:\
    :tc=default:\
```

where `radius-server.domain.com` is the domain name of the RADIUS server.

2 Use **vipw** to set the fifth field of the account line to radius:

bob:<*encrypted passwd*>:<*uid*>:<*gid*>:radius:0:0:
<*clear text name*>:<*home dir*>:<*shell*>

All entries enclosed in < > must be filled in appropriately. You can leave the <*encrypted passwd*> field empty until the first login. Set a password using the **passwd** command after the first login.

3 Create the `/etc/raddb` directory and install the appropriate users, clients, servers, and dictionary files. An entry must be made for each user who will be validated using RADIUS.

The server's file must have the name of the radius server and the secret key:

radius-server.domain.com *secret-key*

Specify the *secret-key* in up to but less than 128 characters. You can use numbers, upper case letters, and meta characters.

Fields in User profile

You can view the RADIUS configuration values at the User profile in the `auth-method` field.

Here is the path:

```
super> read user bob
USER/bob read

super> get .
name* = bob
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace }}
active-enabled = yes
allow-system = yes
allow-update = yes
allow-password = yes
allow-debug = yes
prompt = *
log-display-level = none

super> get . auth-method
auth-type = PASSWD
rad-auth-client = { "" 1645 udp "" }
securid-auth-client = { 5500 udp 5510 tcp /var/ace }

super> get . auth-method  rad-auth-client
auth-server = ""
auth-port = 1645
auth-protocol = udp
auth-key = ""
```

The four `rad-auth-client` fields are read-only.

The value for `auth-server` will be the same as *radius-server.domain.com* specified above. The value for `auth-key` will be the same as the *secret-key* specified above.

Remember that the `""` represent null values.

Set securID (option)

The GRF router supports the client side of securID. securID replaces user validation by password with validation by a randomly-generated passcode.

In the securID system, a user receives a card similar to a bank card with a LED panel. The panel displays a 6-digit passcode which is regenerated every 60 seconds. Initially, the administrator must synchronize the card with the securID server's clock. This server usually resides on an administrative network node. At login, the user enters his or her user name and is prompted for a passcode. The user enters their unique 4-digit pin number and the 6-digit code currently displayed on the card LEDs.That code must match that which the securID server recognizes as the current code.

To enable the securID client feature, sites upgrading from a NetStar 5.x release to current Ascend releases must edit the `/etc/login.conf` file. First-time installations of a current Ascend release do not need to modify `/etc/login.conf`.

How securID works

This section briefly explains what happens on the client side of the model when a user logs into the GRF router.

In a system without securID, logging in to the GRF requires the user to first enter a user name. This user name is used to look up the password file entry for that user. If no entry is found for the user, the user is denied access. If a password entry is found, the user is prompted to supply a password.

In a system using securID, the user name is again used to look up an entry in the password file. In this case, the entry for a valid user has an assigned authentication method field rather than solely a password.The authentication method is retrieved from the password file and is required in a second level of validation to map to a class definition in the `/etc/login.conf` file.

If a class definition does not exist for a particular authentication method, the user is denied access. (In a system using securID, if no authentication method is specified for the user in the password file, the default class is used for authentication.)

If securID is the authentication method defined for a user in the password file, then there must exist a securID class in the `/etc/login.conf` file. The specifics of the class definition are described below. If the securID class has the `auth=` field set to `securid`, then the `/usr/libexec/login_securid` program executes to validate the user. This program communicates with the server and prompts the user for a passcode.

Configure the GRF securID client

1 Make sure these lines appear at the end of `/etc/login.conf`:

```
securID:\
      :auth=securid:\
      :tc=default:
```

2 Create a user account on the client side.
Use **vipw** to add `securID:` to each user's password file account. Entries enclosed in `< >` must be filled in appropriately.as shown in this example:

```
userA:<encrypted passwd>:<uid>:<gid>:securID:0:0:
<clear text name>:<home dir>:<shell>
```

All entries enclosed in < > must be filled in appropriately.

You can leave the `<encrypted passwd>` field empty until the first login. Set a password using the **passwd** command after the first login.

3 Make sure the following lines are in the `/etc/services` file:

```
# ACE authentication server
    securid         5500/udp        # ACE server
    securidprop     5510/tcp        # ACE server slave
```

Note: When you modify the `/etc/services` file, do a **grwrite** to save those changes to flash memory.

On RMS node systems, use the **grc** command to save and archive your changes to `/etc/services`.

However, subsequent software upgrades will install the new release version of `/etc/services`, overwriting any changes you may have made. Please be sure to record your changes to that file as you will need to add them again when you upgrade.

4 The `/sdconf.rec` file is created by the securID server. Copy this file from the server and install it in `/var/ace`:

```
# mkdir /var/ace
# cp sdconf.rec /var/ace
```

5 Make sure the securID server has the client router's IP address and name in its configuration file. Ping the router from the securID server.

6 Test your installation

– Set up user accounts and token cards with and without the securID requirements.

– Log in as tester from any machine on the network.

– Follow the directions to enter the username and passcode when prompted.

7 On the GRF, use the **grsite** command to save the new file (`sdconf.rec`) copied to `/var/ace`:

```
# grsite /var/ace
```

securID fields in User profile

You can view securID configuration values at the User profile in the `auth-method` fields, the fields are read-only.

Here is the path:

```
super> read user bob
USER/bob read

super> get .
name* = bob
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace }}
active-enabled = yes
allow-system = yes
allow-update = yes
allow-password = yes
allow-debug = yes
prompt = *
log-display-level = none

super> get . auth-method
auth-type = PASSWD
rad-auth-client = { "" 1645 udp "" }
securid-auth-client = { 5500 udp 5510 tcp /var/ace }

super> get . auth-method securid-auth-client
auth-port = 5500
auth-protocol = udp
auth-slave-port = 5510
auth-slave-protocol = tcp
auth-server-conf-path = /var/ace
```

Save configuration files and reboot

To install the system configuration files, first save the files and then reboot the system. Save the files after you complete the system parameters and again after you configure the media cards and the network services.

GRF 400 and GRF 1600

Use the **grwrite -v** command to save the `/etc` configuration directory from RAM to a flash device.This preserves the configuration files over a reboot.

```
# grwrite -v
```

To save an alternate configuration on the internal flash based upon the currently-running configuration on the internal flash device:

```
# grsnapshot -sP -dP=revision,version
```

Any changes you make to the `/etc/services` file are overwritten when you install a new software release. Record these changes and add them back after the upgrade.

RMS node systems

Use the **grc** command to save a copy of the `/etc` configuration directory

To use **grc** to archive the default set of GRF configuration files to a specified directory on a diskette, enter:

```
# grc save -F -d directory_name
```

Any changes you make to the `/etc/services` file are overwritten when you install a new software release. Record these changes and add them back after the upgrade.

Reboot using shutdown (root login)

To cleanly stop and reboot the system from `root` login, use the UNIX **shutdown** command. The **shutdown** command performs an orderly shutdown, saving memory and allowing any transfers to complete. When the reboot option is specified, the system is rebooted and all media cards are reset.

```
# shutdown -r now
```

Resetting cards during traffic

When a significant amount of traffic is flowing from card A to card B and you reset card B, this does not cause a problem for card A. However, if you remove card B from the chassis, this can cause card A to hang or reboot.

Management Commands and Tools

3

This chapter provides an overview of frequently-used GRF and UNIX commands that are needed for administrative and management tasks. It describes the use and configuration of diagnostic and information-gathering utilities including **grdiag**, **grdinfo**, **threshpoll**, **pinglog**, and CFMS. Much of this information is used in the next chapter, “*Management Tasks*.”

Chapter 3 covers these topics:

Management commands – an overview

This section provides a brief overview of frequently-used management commands. These are administrative and configuration commands, most are prefixed with **gr** and most are GRF-only because they operate on the GRF internal flash.

These commands manage memory and support multiple configuration versions. These include: **flashcmd**, **getver**, **grfins**, **grsite**, **grsnapshot**, **grwrite**, **mountf**, **setver**, **umountf**, and **vpurge**.

Refer to the *GRF Reference Guide* for command syntax and examples. Man pages are available for most of these commands. An asterisk (*) indicates a command used only on the GR-II (GigaRouter) RMS node systems.

Many of the commands read/write the internal flash device. Those commands will mount the flash (**mountf -w**), perform their function, and then unmount the flash. Mounting takes several seconds. If you are doing several commands in a row, mount the flash yourself to avoid the repeated mount/unmount delay. The commands do not mount flash if it is already mounted and do not unmount it if they did not mount it.

This diagram of the control board memory structure provides a reference point as you review the memory commands.

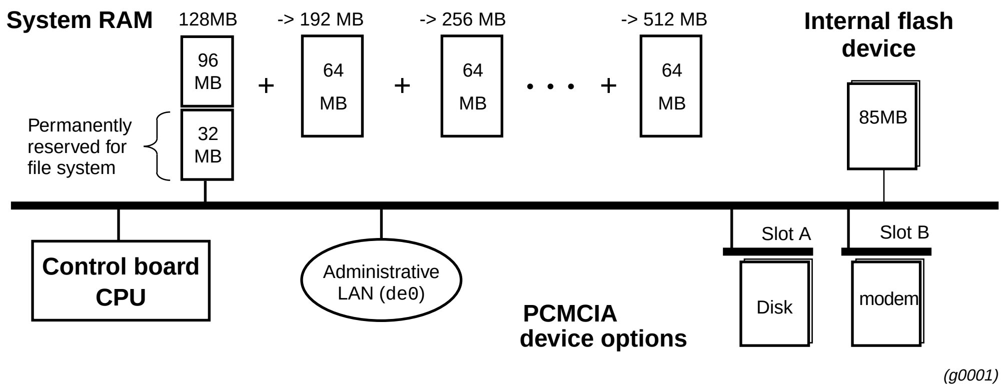


Figure 3-1. GRF control board memory components

csconfig

csconfig sets a PCMCIA slot interface on (up) or off (down), and reports general interface and device status. This command is useful for remote management of PCMCIA devices to verify the status of device and slot interface readiness.

flashcmd

This GRF command mounts the specified flash device, executes a command (such as **write**, **read**, **ls**, **df**) on the device, and then unmounts the device. For example, to use the **df** command to determine device capacity, use **flashcmd df**.

getver

This GRF command tells you the version of the operating system that is currently running. It can also report which release version will be run the next time the system is booted. In this case, **getver** is used in conjunction with **setver**. The **setver** command specifies which release will be run at the next system boot.

grarp

grarp builds tables on media cards that map IP addresses to physical addresses. For HIPPI, the physical address is an I-field; for FDDI and 10/100Base-T, a 48-bit MAC address; and for ATM, an option is the VPI/VCI value.

*grc

On GRF and GR-II systems using an RMS node, the **grc** script archives media configuration files and certain internal OS configuration files such as `/etc/passwd`. This command is replaced on the new GRF control board by **grsnapshot**.

grcard

grcard displays slot number, media type, and current operating status of installed media cards.

grfddi

This command is a utility to set dual and single attachment connections for FDDI interfaces.

grfins

The GRF **grfins** command installs a release onto the internal flash device. In the process, it installs all the new files and converts the system configuration files as required.

As an example, if a release has a new `/etc/gratm.conf` file, **grfins** does not write over your current version. Instead, it installs a new version of the `/etc/gratm.conf.template` file. In this way, you can copy the current configuration information into the new `.template` file, make any changes, and then save that file as the new `/etc/gratm.conf`.

This command is the GRF 400 and GRF 1600.version of the GR-II **grinstall** command.

*grinstall

On GRF and GR-II systems using an RMS node, **grinstall** installs the specified version of the operating software. This command is replaced by the **grfins** command on the GRF 400 and GRF 1600.

grlamap

This command builds a table that sets logical addresses to slot number mappings for HIPPI-SC (switch mode) addressing.

grreset

This command resets one or more specified media cards. Options can direct that memory be dumped when the media card comes back up **(grreset -D**) or that the media card be held in reset **(grreset -h**).

grrmb

This command enables you to use a set of control board status commands. These commands require the `GR ##>` screen prompt which is invoked by executing the **grrmb** command.

When **grrmb** is entered, the screen prompt changes to:

```
# grrmb
GR ##>
```

where ## is the number of a chassis slot. The default is 66, specifying that the command will act on slot 66, the control board. **grrmb** commands (at GR 66> prompt) include:

?	-	lists **grrmb** command set
bignore	-	displays broadcast ignore status, on or off
fan	-	displays RPMs for the chassis fans
maint *number*	-	these commands return media card statistics
power	-	displays on/off status of GRF 1600 power supplies
port *number*	-	sets specific slot for GR##> prompt
temp	-	returns internal temperature readings

grroute

This command adds the routes specified in the `/etc/grroute.conf` configuration file. This file maintains the set of static routes to remote nodes. If you are running GateD, do not use **grroute**, you must use the GateD Static Statement to create static routes.

grrt

The **grrt -p *slot* -S** command displays the route table for an individual media card. Other options delete table entries, display the route to a specified address, and so on.

grsite

The GRF **grsite** command enables you to manage and install individual files after the main release is loaded onto RAM. The file could be a new media card binary to be used for debug or testing. A **grsite *filename*** command overwrites the current version of ***filename***, but archives the original so you can go back to it if necessary. **grsite** has options to add, delete, or list files in the current release, the next boot release, or an arbitrary release set. Note that **grsite** does not work with files in the `/etc` configuration directory, **grwrite** saves those files.

grsnapshot

This GRF command runs a script that can be specified to copy configuration files (or release images) to a target flash device under a new or the current version name. For example, **grsnapshot** can be set up to initialize an external (PCMCIA) flash device, copy the entire contents of the internal flash device to it, and rename the image as a backup.

On the GRF 400 and GRF 1600, **grsnapshot** replaces **grc**, the archival command used on the RMS node.

grstat

This command returns Layer 3 media card statistics for all except HIPPI media cards and Layer 2 statistics for ATM OC-3c (ATM/Q), HSSI, Ethernet, and SONET cards.

grwrite

The GRF **grwrite** command is crucial on the GRF because it saves configuration changes made in the `/etc` directory to internal flash. This saves the changes across system boots. By default, **grwrite** saves a copy of those files with a newer timestamp than the last boot.

mountf

This GRF command mounts any flash device so that the device looks like a file system to the operating system. The flash device is mapped as `/flash` into working RAM as part of the available file system. Mounting a flash device enables various processes to be applied to the device. A device is mounted as read only (default) or writable. See the **umountf** command.

*pwrfaild

This command is used on GRF and GR-II systems using an RMS node but is not needed on the GRF 400 or GRF 1600. **pwrfaild** is the power failure monitoring daemon. It responds to power failure signals sent by a UPS connected to the RMS node, and initiates a clean shutdown of the router manager system.

setver

This GRF command specifies the software version that will load during the next system reboot. The general form of the command is `setver` *`release_name`*. When **setver** executes, it verifies that the specified *`release_name`* can actually be loaded by checking to see that the appropriate release files, startup scripts, and configuration entities are in place. You see a message if these release components are incomplete. See also **getver**.

umountf

This GRF command unmounts a flash device previously mounted by the **mountf** command. See also **mountf**.

vpurge

This GRF command removes a specified release or configuration version from a specified flash device.

UNIX tools

The system provides standard UNIX debugging tools for monitoring and debugging. Please access a UNIX system for standard UNIX man pages. GRF-specific man pages are available.

ping

This standard tool generates and receives ICMP/IP echo request and reply messages. It is used to test connectivity to a specific interface or host.

When **ping** is directed from the kernel out to a system external to the GRF, the command behaves in the standard manner. When **ping** is directed from the kernel to one of the router's interface addresses, the echo request is sent to the appropriate media card and the status of the network interconnection is checked.

When **ping** is directed from an external system to any GRF address, the echo request is sent to the appropriate media card and the status of the network interconnection is checked.

Refer to the *GRF Reference Guide* for **ping** examples.

route

Static routing can be configured by using either the UNIX **route** command or the GRF **grroute** command. Routing is the primary function of a router that allows IP traffic from one network to reach another network. The GRF supports both static and dynamic IP routing.

The UNIX **route** command can be used to manually add or delete routes. When **route** is used, no media card or system reset is needed to install the new routes, the new routes are updated in the kernel and downloaded into each media card automatically.

The GRF **grrt** command can also be used to examine the routing table on a specific media card but it is not recommended for large routing table configuration because it does not ensure that routing tables are synchronized among the various media cards.

Static routes can also be set by editing the `/etc/grroute.conf` configuration file. Changes made via this configuration file do not take effect until the affected media card is reset or the GRF system is reset.

Dynamic routing can be configured by editing the `/etc/gated.conf` configuration file and running the **gated** daemon in the Ascend router. GateD implements complex routing protocols. Please refer to the *GRF GateD Manual* for information about using GateD on the GRF.

Note: If you plan to run GateD, set up your static routes in `/etc/gated.conf` by using the Static statement. If you add routes using the **route** command when GateD is active, those routes are removed by GateD.

tcpdump

This standard UNIX media examination tool is modified for use with media card protocols. **tcpdump** also works on the router's Ethernet LAN interface located on the GRF control board (`de0`), and the communications bus (**rmb0**).

tcpdump prints out all packet headers or a specified type of header transmitting on the target network. Note that **tcpdump** can interfere with network operations and performance.

When using a **tcpdump** on a router interface, local pings are reported as two ICMP requests instead of a request-response pair. This is an artifact resulting from the way in which filtering and local ping sequencing is handled on the media cards. No other effects on packet filtering or other operations of **tcpdump** have been observed as a result of this artifact.

traceroute

This standard command prints the route that packets must take to a destination network host. **traceroute** uses the ICMP/IP parameters time-to-live and time-exceeded to trace a route between two IP entities and provide IP destination statistics. **traceroute** is available from the CLI and the UNIX shell.

You can use the **traceroute** command to determine if packets from an external host are actually being routed through the GRF to get to the target destination address.

ifconfig

ifconfig is used to assign an address to a logical GRF interface and/or to configure interface parameters. The command has been modified for use in the GRF.

One modification includes the use of three special interface names, **-a**, **-ad** and **-au**. The names are reserved and specify a group of logical system interfaces. When one of these interface names is used, the commands following it apply to the specified group:

– **a** applies the command(s) to all interfaces in the system

– **ad** applies the command(s) to all interfaces marked “down”

– **au** applies the command(s) to all interfaces marked “up”

You can also configure GRF interfaces by editing the `/etc/grifconfig.conf` file. Information in `/etc/grifconfig.conf` is eventually turned into **ifconfig** commands.

Using the netstat command

The UNIX **netstat** command reports status and information about media card physical interfaces. **netstat** is available from the CLI and the UNIX shell.

- **netstat -r -s** prints routing statistics
- **netstat -i -n** shows all configured interfaces
- **netstat -a -n** prints a list of all active connections
- **netstat -g -n** prints the multicast route table
- **netstat -r -n** prints the current table of installed routes

  In the output from **netstat -r -n**, the => symbol next to a route means it is a duplicate key, but with a different netmask.
- **netstat -rn | wc -l** returns the number of entries in the routing table, here is an example of a 50-entry table:

  ```
  # netstat  -rn | wc -l
       50
  ```
- **netstat -s** prints comprehensive statistics for protocols, including: IP, ICMP, TCP, and UDP, and GRIT, GRIEF, and GRID for GRF entities.

Refer to the man page for a complete list of **netstat** options. Examples of **netstat** usage follow.

netstat -r -n

Use this **netstat** command to determine that a media card has the correct routing table entries. **netstat -rn** shows current routing tables, **-n** prints out numeric IP addresses:

```
# netstat -rn
Routing tables

Internet:
Destination      Gateway            Flags     Refs      Use  Interface
192.168.20       192.168.20.11      U           0         0  gf010
198.174.11       link#1             UC          0         0  de0
198.174.11.2     8:0:20:1b:24:d2    UHL         0         3  de0
198.174.11.38    8:0:7:bc:d:b1      UHL         2        83  de0
198.174.11.155   8:0:20:78:9c:60    UHL         1       255  de0
198.174.11.156   8:0:20:7a:e0:63    UHL         1      1673  de0
198.174.11.239   8:0:20:74:1a:a8    UHL        16      5640  de0
198.174.11.249   0:c0:80:b:30:53    UHL         4       170  lo0
198.174.11.250   0:60:2f:3:45:42    UHL         0         0  de0
204.221.156      204.221.156.33     U           0         0  gh030
222.222.90/26    222.222.90.3       U           0         0  ga020
222.222.90.64/26 222.222.90.67      U           0         0  ga0280
222.222.91/26    222.222.91.3       U           0         0  ga021
222.222.91.64/26 222.222.91.67      U           0         0  ga022
222.222.92/26    222.222.92.3       U           0         0  ga023
222.222.92.64/26 222.222.92.67      U           0         0  ga024
224/8            link#1             UC          0         0  de0
```

In the **netstat -rn** output, you will see the routing entries for the media cards installed in the router. In the example above there are three cards, a FDDI card in slot 1, a HIPPI card in slot 3, and an ATM card in slot 2. Media cards are identified by their `Interface` names having the form `gt0y0` where `y` is the number of the chassis slot in which a specific card is installed.

For more information on the logical interface naming convention, refer to chapter 2 in this manual. Each `Interface` in the **netstat -rn** output should correspond to at least one route that specifies the reachable network in the `Destination` column.

netstat -r -s

Using both the **-r** and **-s** options, **netstat** prints routing statistics:

```
% netstat -r -s
routing:
        0 bad routing redirects
        0 dynamically created routes
        0 new gateways due to redirects
        44 destinations found unreachable
        0 uses of a wildcard route
```

netstat -i -n

Here is an example of output from **netstat -i -n** listed by media card interface name:

```
# netstat -i -n
Name    Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
de0     1500  <link1>   00:c0:80:0b:30:53 42665    0   8099    0 2584
de0     1500  198.174.11  198.174.11.249  492665   0   8099    0  2584
rmb0    596   <link2>  00:00:00:00:00:00 130022    0 129726    0    0
rmb0    596   <GRIT>    0:0x40:0        130022    0 129726    0    0
lo0     1536  <link3>                      496    0    496    0    0
lo0     1536  <GRIT>    0:0x48:0           496    0    496    0    0
gl000*  1524  <link4>                        0    0      0    0    0
gf010   4352  <link5>    00:c0:80:00:55:d1    0    0      0    0    0
gf010   4352  192.168.20  192.168.20.11      0    0      0    0    0
gf011*  4352  <link6>    00:c0:80:00:55:d2    0    0      0    0    0
gf012*  4352  <link7>    00:c0:80:00:55:d3    0    0      0    0    0
gf013*  4352  <link8>    00:c0:80:00:55:d4    0    0      0    0    0
ga020   9180  <link10>                       0    0      0    0    0
ga020   9180  222.222.90/ 222.222.90.3       0    0      0    0    0
ga021   9180  <link11>                       0    0      0    0    0
ga021   9180  222.222.91/ 222.222.91.3       0    0      0    0    0
ga022   9180  <link13>                       0    0      0    0    0
ga022   9180  222.222.91. 222.222.91.67      0    0      0    0    0
ga023   9180  <link14>                       0    0      0    0    0
ga023   9180  222.222.92/ 222.222.92.3       0    0      0    0    0
ga024   9180  <link15>                       0    0      0    0    0
ga024   9180  222.222.92. 222.222.92.67      0    0      0    0    0
ga025*  9180  <link16>                       0    0      0    0    0
ga026*  9180  <link17>                       0    0      0    0    0
ga027*  9180  <link18>                       0    0      0    0    0
```

```
ga028*  9180  <link19>                        0    0      0    0    0
ga029*  9180  <link20>                        0    0      0    0    0
ga02a*  9180  <link21>                        0    0      0    0    0
ga02b*  9180  <link22>                        0    0      0    0    0
ga02c*  9180  <link23>                        0    0      0    0    0
ga02d*  9180  <link24>                        0    0      0    0    0
ga02e*  9180  <link25>                        0    0      0    0    0
ga02f*  9180  <link26>                        0    0      0    0    0
ga0210* 9180  <link27>                        0    0      0    0    0
ga0211* 9180  <link28>                        0    0      0    0    0
ga0212* 9180  <link29>                        0    0      0    0    0
ga0280  9180  <link9>                         0    0      0    0    0
ga0280  9180  222.222.90. 222.222.90.67       0    0      0    0    0
gh030   65280 <link12>                        0    0      0    0    0
gh030   65280 204.221.156 204.221.156.33      0    0      0    0    0
#
```

netstat -s

This excerpt from **netstat -s** shows the statistics reported for the IP protocol:

```
# netstat -s
ip:
        211338 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 with bad options
        0 with incorrect version number
        29285 fragments received
        0 fragments dropped (dup or out of space)
        0 fragments dropped after timeout
        4885 packets reassembled ok
        171636 packets for this host
        2948 packets for unknown/unsupported protocol
        0 packets forwarded
        12295 packets not forwardable
        0 redirects sent
        8049 packets sent from this host
        0 packets sent with fabricated ip header
        0 output packets dropped due to no bufs, etc.
        0 output packets discarded due to no route
        0 output datagrams fragmented
        0 fragments created
        0 datagrams that can't be fragmented
```

netstat -g -n

This excerpt from **netstat -g -n** shows a multicast forwarding information base (FIB):

```
# netstat -g -n

Multicast Forwarding Cache
 Hash  Origin-Subnet     Mcastgroup   # pkts In-Vif  Out-Vifs/Forw-ttl
   4   193.167.64.154  224.42.42.2     4004250336    65535
   5   164.58.253.9    224.2.2.1         48m   65535
  16   128.223.156.117 224.2.231.173      1     0
  17   130.207.8.30    224.2.2.2       1024m   65535
  19   128.9.192.69    224.2.221.38     131k   65535
  22   128.9.192.69    224.2.134.250      0    65535
  27   204.123.13.69   224.2.144.67       0    65535
  27   132.236.77.25   224.0.14.1        24     0
  30   204.123.13.69   224.2.204.67       0    65535
  35   128.9.112.151   239.140.173.5     90m   65535
  •
  •
  •
 229   139.184.163.8   224.2.172.238      0    65535
 238   128.9.160.45    239.140.173.3   1316m   65535
 242   171.69.56.76    224.2.191.234      0    65535
 243   192.188.104.97  224.2.172.238    604m   65535
 245   139.88.39.110   224.2.167.198    140k   65535
 245   205.226.8.183   224.2.2.1          0    65535
 245   128.9.160.43    224.2.221.38       0    65535
 245   164.58.253.9    224.2.1.2          0    65535
 247   131.243.73.36   224.2.195.166     53m   65535
 247   130.240.64.47   224.2.213.97       1     0

Total no. of entries in cache: 94
```

netstat -a -n

Here is an excerpt from **netstat -a -n** showing active connections:

```
# netstat -a -n
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp       0      0  198.174.11.249.23      198.174.11.38.1073     ESTABLISHED
tcp       0      0  198.174.11.249.199     198.174.11.249.1026    ESTABLISHED
tcp       0      0  198.174.11.249.1026    198.174.11.249.199     ESTABLISHED
tcp       0      0  *.199                  *.*                    LISTEN
tcp       0      0  198.174.11.249.199     198.174.11.249.1024    ESTABLISHED
tcp       0      0  198.174.11.249.1024    198.174.11.249.199     ESTABLISHED
tcp       0      0  *.23                   *.*                    LISTEN
udp       0      0  *.*                    *.*
udp       0      0  *.161                  *.*
udp       0      0  198.174.11.249.1056    198.174.11.239.2049
udp       0      0  198.174.11.249.1054    198.174.11.239.2049
udp       0      0  198.174.11.249.1046    198.174.11.239.2049
udp       0      0  198.174.11.249.1044    198.174.11.239.2049
```

```
udp        0      0  *.*                    *.*
Active GRIT connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
grit       0      0  *:25                   *:*
grit       0      0  *:*                    *:*
grit       0      0  *:32                   *:*
grit       0      0  *:*                    *:*
grit       0      0  *:27                   *:*
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
f0d5ff80 dgram       0      0        0 f0793b94        0 f0c4d694
f0cd4500 dgram       0      0        0 f0793b94        0 f0879494
f09eb200 stream      0      0        0        0        0        0
f0c22980 stream      0      0        0        0        0        0
f0dc0f80 stream      0      0        0        0        0        0
f0cd4e00 dgram       0      0        0 f0793b94        0 f0c8a414
f0cd4b00 dgram       0      0        0 f0793b94        0        0
#
```

GRF logs

This section provides examples of logged information for the GRF and its media cards.

Space limitations require that the GRF log to a PCMCIA device or remote **syslog** server rather than to its own system memory. Procedures to configure remote logging and the PCMCIA device are in the *GRF 400/1600 Getting Started* manual, chapter 4. Logs are maintained in the directory `/var/log`.

Three logs provide specific information useful for monitoring and debugging GRF operations. If you are working with Customer Support, these are the three logs they will need to see:

– `/var/log/gr.console`

– `/var/log/messages`

– `/var/log/gr.boot`

The `/var/log` directory contains other log files that collect low-level information useful primarily to system developers.

The `gr.console` log is the most useful log. It contains status and events for the GRF system and all media cards. When a media card resets, many events of the resetting are reported, including initializing, loading run-time code, requesting and reading configuration parameters, and so on. At the end, you see a message that indicates the cause of the reset.

The `messages` log contains system-related events connected usually with the management software (also referred to as RMS, Router Management System) and the operating system kernel.

The `gr.boot` log contains events reported during system and media card boot. These can be helpful if a card has problems booting and coming up.

Accessing a log file

To display the contents of a specific log file, change directory to `/var/log` and use the **more** command to display the contents of a specific log file.

To access output of **grconsole** log, use this sequence of commands:

```
# cd /var/log
# more gr.console
```

Use grdinfo to collect logs

With a single command, **grdinfo** collects the files in local `/var/log/*` (including compressed files)and compresses them in a log file. Refer to the **grdinfo** section in this chapter for more information.

On the following pages are sample logs from the machine “`box1.ascend.com`”.

Sample gr.console log

The `gr.console` log contains messages issued by the media cards and the control board. They include run-time errors, diagnostic information, and the operational status of each media card.

```
# more gr.console
Apr 12 05:18:10 box1 gritd: from 0:0x5:0:  dst=0:0x40:16, src=0:0x5:0, type=GRID
:  hwtype=FDDI_V2 cmd=MSGP 'arp info (8:0:69:4:4a:a0) overwritten for 192.168.4.
137 by 8:0:69:4:4c:e6\r\n'
Apr 12 05:18:30 box1 gritd: from 0:0x7:0:  dst=0:0x40:16, src=0:0x7:0, type=GRID
:  hwtype=ATM_OC3_V2 cmd=MSGP '[RX] sending last GRID rsp 62270\r\n'
Apr 12 05:19:44 box1 gritd: from 0:0x2:0:  dst=0:0x40:16, src=0:0x2:0, type=GRID
:  hwtype=ATM_OC3_V2 cmd=MSGP 'sending last GRID rsp 13632\r\n'
Apr 12 05:22:11 box1 gritd: from 0:0x9:0:  dst=0:0x40:16, src=0:0x9:0, type=GRID
:  hwtype=SONET_V1 cmd=MSGP '[RX] sending last GRID rsp 43328\r\n'
Apr 12 05:28:10 box1 gritd: from 0:0x5:0:  dst=0:0x40:16, src=0:0x5:0, type=GRID
:  hwtype=FDDI_V2 cmd=MSGP 'arp info (8:0:69:4:4c:e6) overwritten for 192.168.4.
137 by 8:0:69:4:4a:a0\r\n'
Apr 12 05:28:10 box1 gritd: from 0:0x5:0:  dst=0:0x40:16, src=0:0x5:0, type=GRID
:  hwtype=FDDI_V2 cmd=MSGP 'arp info (8:0:69:4:4a:a0) overwritten for 192.168.4.
137 by 8:0:69:4:4c:e6\r\n'
Apr 12 05:28:14 box1 gritd: from 0:0x2:0:  dst=0:0x40:16, src=0:0x2:0, type=GRID
:  hwtype=ATM_OC3_V2 cmd=MSGP 'sending last GRID rsp 45379\r\n'
```

Figure 3-2. Sample entries in the gr.console log

Sample gr.boot log

When a media card boots, information about its boot status is written to `gr.boot`. Here is a sample `gr.boot` log from a GRF with host name `box1.site.com`:

```
# more gr.boot
Apr 11 16:15:02 box1 grbootd[281]: 0:0xf:0 sent BOOTME
Apr 11 16:15:02 box1 grbootd[281]: dumping 0:0xf:0
Apr 11 16:15:02 box1 grbootd[281]: 0:0xf:0 sent LOADME
Apr 11 16:15:02 box1 log2[6993]: grdump.sh exec /usr/nbin/grdump  -b -i 3 -p 0:0 xf:0
Apr 11 16:15:02 box1 grdump[6982]: Grdump of 0:0xf:0 starting up ...
Apr 11 16:15:02 box1 grdump[6982]: grinch card pre-death state (2.12.2.16.5.3=0x 5)
Apr 11 16:15:09 box1 grdump[6982]: Dump of 0:0xf:0 finished.
Apr 11 16:15:09 box1 grbootd[281]: dump of 0:0xf:0 done, booting...
Apr 11 16:15:09 box1 grbootd[281]: 2.1.4.3.12.4=/usr/libexec/portcards/atm-3.run
Apr 11 16:15:09 box1 grbootd[281]: Boot Image file is Zipped:
        /usr/libexec/portcards/atm-3.run
Apr 11 16:15:10 box1 grbootd[281]: read 881220 bytes from /usr/libexec/portcards
/atm-12.run
Apr 11 16:15:10 box1 grbootd[281]: Ready to load 0:0xf:0 with
        /usr/libexec/portcards/atm-3.run (sending ACK)
Apr 11 16:15:15 box1 grbootd[281]: 5 boot images resident (1 max)
Apr 11 16:15:15 box1 grbootd[281]: boot image ager scheduled for 60s
Apr 11 16:15:15 box1 grbootd[281]: 0:0xf:0 loaded (1541 pkts, 0 re-xmits) 881220
        data + 24656 proto bytes in 4.59s (197.57 Kb/s)
Apr 11 16:15:59 box1 grbootd[281]: 0:0x5:0 sent BOOTME
```

Figure 3-3. Sample entries in the gr.boot log

Sample messages log

This is the general operating system log. It contains boot or deadstart commentary, system-level warnings, and error messages.

This is a sample `messages` log from a GRF with host name `box1.site.com`.

```
# more messages
Mar 25 03:30:17 box1 grinchd[122]: sendto: No buffer space available
Mar 25 03:30:47 box1 last message repeated 3 times
Mar 25 03:32:47 box1 last message repeated 8 times
Mar 25 03:42:47 box1 last message repeated 40 times
Mar 26 10:49:57 box1 kernel: de0: framing error
Mar 27 10:45:39 box1 su: scottsw to root on /dev/ttyp1
Mar 27 10:45:58 box1 kernel: gh070:  GigaRouter HIPPI, GRIT address 0:7:0
Mar 27 10:46:00 box1 kernel: gh030:  GigaRouter HIPPI, GRIT address 0:3:0
Mar 27 11:23:26 box1 kernel: gh070:  GigaRouter HIPPI, GRIT address 0:7:0
Mar 27 11:23:27 box1 kernel: gh030:  GigaRouter HIPPI, GRIT address 0:3:0
Mar 27 15:57:39 box1 kernel: uid 26 on /usr: file system full
Mar 27 15:57:54 box1 last message repeated 2 times
Mar 27 12:57:13 box1 login: ROOT LOGIN (root) ON ttyp1 FROM summa
Mar 27 13:00:43 box1 login: ROOT LOGIN (root) ON ttyp1 FROM othermac
```

Figure 3-4. Sample entries in the messages log

grclean utility

The **grclean** utility is an internal program that compresses, archives, and manages dump files, and saves them to a specified file name ending with `.gz`.

You can set size limits for various system logs in the `/etc/grclean.logs.conf` file. Here are several such entries:

```
size=150000
logfile=/var/log/gr.console

size=10000
logfile=/var/log/fred.log

size=10000
logfile=/var/log/aitmd.log

size=10000
logfile=/var/log/grinchd.log
```

Managing media card dumps

The GRF control board memory provides limited file system space for dumps. This section tells you how the system manages dumps and describes site management options. The "Management Tasks" section describes how to collect, configure, and ftp dumps.

grdump

grdump is the background program that captures memory dump images. This program acts according to variables set in the DUMP profile. Each dump image is stored in a file named with the convention: `grdump.n.x.gz`
where `n` is the card slot number and `x` is the number of the saved dump. The first dump of the day is labeled `grdump.n.old.gz` to distinguish the first dump from any other dumps that might occur during reboot or other event.

grdump runs on the fly, it uses the **gzip** utility to compress dumps and save space. Compressed files are appended with `.gz`.

Reset and dump card

grreset -D *slot* is a user command that causes a media card to be reset and **grdump** to dump its memory. Refer to the *GRF Reference Guide* for more information about **grreset**.

Panic dumps sent to external flash device

When a media card panics and there is a formatted external flash device plugged into either PCMCIA slot, a copy of the dump is automatically saved under the `portcards` directory of the external flash.

DUMP profile

System-level settings in the Dump profile set how many are saved and which events will cause a dump. Each Card profile has a dump section in which you can customize dumps for an individual card. Refer to chapter 1 for a description of dump options in profiles.

Default settings enable two dumps saved daily per media card in addition to the first and last dumps of the day. The default settings will automatically manage the available space so that the file system does not fill and cause a crash. It is recommended that you send dumps to external storage.

Use grdinfo to collect dumps

With a single command, **grdinfo** collects media card dumps, utility dumps, core and other dumps, mini dumps, and a kernel dump, if available., and compresses them in a log file. Refer to the **grdiag** section in this chapter for more information.

RMS monitoring functions

Router management software performs a variety of monitoring and reporting functions, including the following:

- The kernel tracks communications bus packets sent by the media cards. If no packet is seen from a media card for a specified period (according to a timer), the kernel forwards an echo request packet once per second until a response is received. The timer period can be specified, its default is five seconds.
- If the media card does not respond to the echo request after the specified time, the kernel determines the card is hung and begins an automatic card reset.
- By default at start-up, the system initializes every media card with a snapshot of the current routes and configuration of GRF interfaces.
- During normal operations, any change to the state of a GRF interface or to the system route table is sent to each media card. This also enables the system to synchronize the system configuration.
- When a media card panics, the system resets the card. After a media card has reported configuration errors, the card may need to be held in reset rather than be rebooted.
- When a media card reports configuration errors as it is being configured, the system resets the card.

grdebug options

The **grdebug** command enables/disables the set of monitoring functions listed in Table 3-1. The command `grdebug -p 3 off` disables all monitoring of the media card in slot 3.

Table 3-1. Enable/disable options for grdebug

Option:	**Function enabled / disabled:**
grdebug -C	the configuration reset for a specific media card
grdebug -E	the timed echo request for a specific media card
grdebug -H	hold in reset for a specific media card
grdebug -I	automatic initiation for a specific media card
grdebug -P	the panic reset for a specific media card
grdebug -p	with “on” or “off”, turns off all monitoring of a specific media card
grdebug -R	automatic reset for a specific media card that the system assumes is hung
grdebug -T	the kernel “watch” timer for a specific media card
grdebug -U	automatic updating of a specific media card’s route table

Refer to the *GRF Reference Guide* for more information.

A note about the combus

The media cards and control board communicate across the communications bus, or combus.

Combus traffic includes error messages, status requests, route updates to media cards, route updates from media cards, configuration changes, log messages, and keepalives. Heavy traffic on the combus affect other parts of the system. For example, if heavy error message traffic blocks media card responses to RMS keepalive messages, the system understands that the card is not responding and in certain circumstances may reset it.

Tools described in the next sections may cause heavy traffic across the combus. Be aware that some system communications may be disrupted while **grdinfo**, **threshpoll**, or CFMS operate.

Field diagnostic tool – grdiag

This section describes the diagnostic capability provided by the **grdiag** command. Users can run a set of internal BIST-level diagnostics to verify media card hardware. A media card that fails this set of diagnostics must be replaced. **grdiag** operates on GRF 400 and GRF 1600 routers as well as on the GR-II. HIPPI media cards do not support the **grdiag** command.

The **grdiag** script puts the selected media card(s) into diagnostic mode and runs the diagnostics. After the diagnostics complete, **grdiag** reloads the media card's software and configuration currently saved in flash memory, then reboots the card. For this reason, it is very important that you save any configuration changes before you run **grdiag**. Unsaved media card changes will be lost. These diagnostics affect the operation of only the target card or cards. You can run diagnostics on all the chassis cards at the same time. The length of time needed for the diagnostic to run depends on the type of media card and how many cards are being tested at one time.

What is tested

grdiag is intended to help users determine whether hardware is causing a problem that is being seen. These diagnostics do not determine which type of hardware failure occurred. The diagnostics report no error information, only pass-fail results.

The diagnostics verify the following media card and slot functions:

- all memory
- all media hardware logic (media card)
- all serial hardware logic (serial daughter card)
- the connection between the slot and the switch
- the connection between the slot and the communications bus
- the connection between the slot and power delivery

The diagnostics do not exercise the physical interfaces or transceivers. Generally, you can expect to test 95% of the media card.

grdiag log files

grdiag reports to `/var/log/grdiag.log` and to the `/var/log/gr.console` log. If you are logging remotely, check that location for **grdiag** reports.

Pass-fail status reports from the diagnostic tests are sent to `/var/log/grdiag.log`. This is the same information that is displayed to you after **grdiag** completes:

```
# vi /var/log/grdiag.log

Start date: Mon Apr 20 19:40:12 CDT 1998;  Tested by: netstar
Test time: 0 hrs; 10 min.  End date: Mon Apr 20 19:51:18 CDT 1998
************************************************************************
*           Field Diagnostic Test Ended. 3 Passed 1 Failed.            *
************************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     ethernet-v1       BIST monitor   Failed
1     hssi              Idle           Passed
2     fddi-v2           Idle           Passed
3     fddi-v2           Idle           Passed
```

Event and error code reporting is done in the `gr.console` log. The diagnostic start and stop events are reported:

```
# vi gr.console

!! Start of Diagnostic Test !!
    .
    .
    .
!! End of Diagnostic Test !!
```

If a media card fails, an error code is reported to `gr.console` log. The first two digits are the slot number of the failed media card. The next number is the major error descriptor, the last number is the minor descriptor. Record the error code and send it to your support staff.

In this example, the card in slot 1 has failed:

```
!! End of Diagnostic Test !!

Built-In Self-Test Error Code = 01-3333-44
                                 |    |   |
                               Slot   |   |
                                    Major |
                                        Minor
```

Stopping or halting grdiag

You can use Control-C to stop the diagnostic sequence at any time. After you enter Control-C, **grdiag** reloads the card’s run-time binary and last-saved configuration, and then reboots the card.

When a media card does not boot

For **grdiag** to run, a card must be able to boot. If the **grcard** display does not include the slot in which the problem card resides, **grdiag** cannot operate on that card.

For example, **grdiag** cannot run diagnostics on the card in slot 1 of this GRF 400:

```
# grcard
 0      ATM_OC3_V2     running
 2      ATM_OC3_V2     running
 3      HSSI_V1        running
```

Special login

Do not log in directly as `root` to use the **grdiag** command. To use **grdiag**, you must log in as a user and then **su** to `root`.

This example uses the `netstar` login (password = Ascend) that the GRFs are shipped with:

```
User: netstar
Password: ......
erase ^H, kill ^U, intr ^C status ^T
$
```

If you changed the default password `Ascend` as recommended, use the new password.

At the next prompt, enter **su** and use the `root` password at the prompt. You will see the UNIX prompt appear:

```
$ su
Password:
#
```

Now you can run **grdiag**:

```
# grdiag
```

Running the grdiag startup script

The script is simple to run. These are the choices you will make:

– choose to save unsaved changes – y / n ?

– enter slot number(s) of media card(s) to test

Enter the **grdiag** command:

```
# grdiag
                       Portcard Field Diagnostic
#####################################################################
# WARNING: Make sure your current Configuration is Saved to FLASH!! #
#####################################################################
If Not Would you Like to Save it Now  y/n? [y]:
```

You see this warning whether or not there are unsaved configuration changes. If you enter yes, save changes, the activity on internal flash is reported back:

```
If Not Would you Like to Save it Now  y/n? [y]: y
Device /dev/wd0a mounted on /flash.
Device /dev/wd0a unmounted.
```

If you enter No and you do have unsaved changes, the last-saved configuration will be reloaded after the diagnostic sequence runs. A reply is not made to a No entry.

The first **grdiag** display is an inventory of the current media card status
 "N/A" indicates that these diagnostics do not run on the HIPPI card):

```
------------------------------------------------------------------------
-                         Media Card Inventory                         -
------------------------------------------------------------------------
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        running
2     hssi              running
3     hippi-v1          running        N/A
```

After the inventory display, you are asked to enter the slot numbers of the card(s) to test:

```
Enter the media card slot numbers to test
   Use "all" or a space separated list (0 1 2 etc.): 1 2
```

The list of cards queued to be tested is displayed:

```
------------------------------------------------------------------------
-                        Cards Queued for Test                         -
------------------------------------------------------------------------
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        running        Queued
2     hssi              running        Queued
3     hippi-v1          running        N/A

Are you absolutely sure you want to proceed?  y/n? [No]:
```

After the queue list is displayed, you are asked to verify that you want to start the diagnostic, the default is No. If you answer No, you are given a chance to change the parameters you have already specified. If you answer No again to changing parameters, the **grdiag** script ends and you are back at the shell prompt:

```
Are you absolutely sure you want to proceed?  y/n? [No]: n
  Re-enter test parameters?  y/n? [No]: n
#
```

If you enter Yes, continue with the diagnostic, **grdiag** automatically accesses the CLI and reads the target card(s) Card profile(s).

Activity during the testing

grdiag saves the card's last-saved configuration to a file, and then changes Card profile load parameters so that the diagnostic code is loaded and run as you specified. The new settings are saved just as they are when you change parameter settings. You may see some of this activity on the screen, most of it speeds by too quickly to read:

```
ncli: waiting for mibmgrd to initialize, hit ^c to abort.
ncli: mibmgrd initialized.
Attempting to connect to mibmgrd... timeout in 15 secs

super> read card 2
CARD/2 read
super> cd load
super> new boot-seq-table 1
boot-seq-table/1 created
super> cd boot-seq-table 1
index = 1
hw-type = no-media
rx-path = ""
tx-path = ""
super> set hw-type = hssi
super> set rx-path = /usr/libexec/portcards/hssi_rx_diag.run
super> set tx-path = /usr/libexec/portcards/hssi_tx_diag.run
super> write
CARD/2 written
```

Now you see **grdiag** reports that show loading and testing events:

```
************************************************************************
*             Waiting 114 sec. for Queued Cards to Load                *
************************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        loading        Queued
2     hssi              loading        Queued
3     hippi-v1          running        N/A


Test started: Mon Apr 20 13:24:44 CDT 1998; Tested by: netstar
Test time: 0 hrs; 0 min; 1 sec.
************************************************************************
*                 Field Diagnostic Test in Progress                    *
************************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        diagnostic     Testing
2     hssi              diagnostic     Testing
3     hippi-v1          running        N/A
```

The testing report is updated five or six times a minute:

```
Test started: Mon Apr 20 13:24:44 CDT 1998; Tested by: netstar
Test time: 0 hrs; 1 min; 18 sec.
```

```
***********************************************************************
*                  Field Diagnostic Test in Progress                  *
***********************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        diagnostic     Testing
2     hssi              diagnostic     Testing
3     hippi-v1          running        N/A


Test started: Mon Apr 20 13:24:44 CDT 1998; Tested by: netstar
Test time: 0 hrs; 1 min; 39 sec.
***********************************************************************
*                  Field Diagnostic Test in Progress                  *
***********************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        diagnostic     Testing
2     hssi              diagnostic     Testing
3     hippi-v1          running        N/A
```

When testing completes

As the diagnostics complete, **grdiag** again accesses the Card profiles and writes the parameters back to the original settings. The tested cards are rebooted even if they failed the test. Again, the display speeds by too quickly to read:

```
ncli: waiting for mibmgrd to initialize, hit ^c to abort.
ncli: mibmgrd initialized.
Attempting to connect to mibmgrd... timeout in 15 secs

super> read card 2
CARD/2 read
    .
    .
    .
CARD/2 written

Ports reset: 2
```

After cards reboot, you see the final report. The report is also sent to `/var/log/grdiag.log`:

```
Test time: 0 hrs; 2 min.  End date: Mon Apr 20 13:27:49 CDT 1998
***********************************************************************
*           Field Diagnostic Test Ended. 2 Passed 0 Failed.           *
***********************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     atm-oc3-v2        running
1     atm-oc3-v2        Idle           Passed
2     hssi              Idle           Passed
3     hippi-v1          running        N/A
```

Though Card Status is reported as idle, the cards are actually up. Use **grcard** to verify status:

```
# grcard
 0      ATM_OC3_V2     running
 1      ATM_OC3_V2     running
 2      HSSI_V1        running
 3      HIPPI_V1       running
#
```

When a card fails...

This is the report you see when a media card fails the diagnostic. It is the same information sent to `/var/log/grdiag.log`:

```
Start date: Mon Apr 20 19:40:12 CDT 1998;  Tested by: netstar
Test time: 0 hrs; 10 min.  End date: Mon Apr 20 19:51:18 CDT 1998
************************************************************************
*           Field Diagnostic Test Ended. 3 Passed 1 Failed.            *
************************************************************************
Slot     Card Type       Card Status   Test Status
----  ----------------  -------------  ------------
0     ethernet-v1       BIST monitor   Failed
1     hssi              Idle           Passed
2     fddi-v2           Idle           Passed
3     fddi-v2           Idle           Passed

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                Possible Additional Error Information                 !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Filtered contents of /var/log/gr.console:

Apr 20 19:51:05> [RMS] rmb0: Resetting Media Card 0
Apr 20 19:51:18> [1] UNEXPECTED Router Manager Interrupt
Apr 20 19:51:18> [2] UNEXPECTED Router Manager Interrupt
Apr 20 19:51:18> [3] UNEXPECTED Router Manager Interrupt
```

Remember that the error code is sent to the `/var/log/gr.console` log.

Data collection utility - grdinfo

The **grdinfo** utility enables the site to use a single command to collect a comprehensive set of debug and configuration information for the GRF. **grdinfo** options specify the type of information collected, including logs, dumps, media card statistics, protocol statistics, and control board data. Target data can be obtained at the system level or at the card level.

You can execute **grdinfo** while the GRF is running although there will be an impact on performance while the information is collected. This is a diagnostic tool. If the media cards are busy forwarding data and are unable to respond to statistics requests, or if not enough disk space is available, you will get an error message reporting the condition. In most cases, **grdinfo** stops and ends.

The **grdinfo** utility is to be used in conjunction with Ascend customer support staff. Some files **grdinfo** creates are very large. As a result, the collection process can interfere with system operations. Data is saved in compressed TAR files for ease in transfers to Technical Support.

Information specified by the specified option or options is collected and compressed into a `grdinfo.tar.gz` file in the `/var/tmp/grdinfo` directory. One **grdinfo** file is collected at a time. The size of a particular `grdinfo.tar.gz` file will vary widely and can tax system file system resources. The **grdinfo** utility is intended to collect information, not to store it. After you use **grdinfo** to collect the needed information, copy the data to external storage such as a file server, and then clean up the`/var/tmp/grdinfo` directory.

It is suggested that you move the output file off the GRF for analysis. Do not extract the data out of the `grdinfo.tar.gz` output file while it is on the GRF.

Options

Options are:

– **grdinfo -card=*slot* | all**
This command returns configuration and state information for a specific media card or for all installed cards. This includes ATM OC-3c, ATM OC-12c,HSSI, SONET, Ethernet, FDDI, and HIPPI cards.

– **grdinfo -config**
the collection of system configuration data, including Card, System, Dump, and Load profiles, and all `/etc/*.conf` configuration files, including `/etc/gated.conf`.

– **grdinfo -log**
the collection of all log files from the local `/var/log` directory

– **grdinfo -dump**
the collection of all dump files from the local `/var` directory, including media card, utility, and kernel dumps

– **grdinfo -system**
an extensive collection of control board (RMS) data

– **grdinfo -frame**
the collection of system-wide Frame Relay status, configurations, and statistics

– **grdinfo -bridge**
the collection of system-wide bridging status, configurations, and statistics

– **grdinfo -dr**
  The dynamic routing option is not available in this release.

– **grdinfo -all**
  collects and combines all the data the other options collect (not recommended)

Caution: The **grdinfo all** command can fill up the file system. It will hang, and you must use Control-C to end the process. Clean up any files that were saved before the abort.

Generated files

grdinfo collects and compresses the requested information into a `tar.gz` file in the `/var/tmp/grdinfo` directory:

```
grdinfo.tar.gz
```

When you unzip and tar `grdinfo.tar.gz`, at least two files are extracted (if specified, dump files will be added):

```
# gunzip grdinfo.tar.gz
# tar xvf grdinfo.tar
# ls
  grdinfo.203150.errors
  grdinfo.203150.info
  grdinfo.tar.gz
```

Note that the `grdinfo.tar.gz` file remains in `/var/tmp/grdinfo`. It will be overwritten by the next **grdinfo** command.

The `.errors` file contains any error messages produced by **grdinfo** while it was running. An `.errors` file is always generated even though it is usually empty. The `.info` file contains the collected debug and status information.

Each time the **grdinfo** command is run, the `grdinfo.tar.gz` file is generated and automatically overwrites the previous `grdinfo.tar.gz` file. This means you will lose the information from the first **grdinfo** command unless you unzip and tar the `grdinfo.tar.gz` file it produces.

After you tar the `grdinfo.tar` file, the **grdinfo** utility assigns a unique number to the resulting `.errors` and `.info` files. In the example above, that number is 203150.

File system usage

It is difficult to predict the size of **grdinfo**-generated files. Here are two examples:

The files from a **grdinfo -system** command (in bytes):

```
-rw-r--r--  1 root  wheel     242 May  4 20:31 grdinfo.203150.errors
-rw-r--r--  1 root  wheel  171456 May  4 20:31 grdinfo.203150.info
```

The files from a **grdinfo -card** command on an Ethernet card (in bytes):

```
-rw-r--r--  1 root  wheel    242 Apr 28 14:27 grdinfo.142749.errors
-rw-r--r--  1 root  wheel  19322 Apr 28 14:27 grdinfo.142749.info
```

File system full messages

If there is not enough disk space to hold the data collected by a particular **grdinfo** command, **grdinfo** will begin the operations anyway. When the file system is nearly full, **grdinfo** issues a series of “Write failed - File system full” messages..

At this point, you must abort **grdinfo** and remove all data collected by this particular command.

Use a Control-C command to do the cancellation. Files created by this invocation of **grdinfo** may not be cleaned up when the user aborts **grdinfo**. The user should check the `/var/tmp/grdinfo` directory for leftover files in case the cleanup is not complete. Any files generated by previous **grdinfo** commands are not removed.

Alternate output file

By default, **grdinfo** writes to the same output file, `/var/tmp/grdinfo/grdinfo.tar`, unless you specify a different destination using the **grdinfo -ofile=*file_name*** command. The output file can be specified to external flash (PCMCIA) or to an NFS file system.

Remote logging

If the GRF is configured to log remotely, **grdinfo** does not collect files from the remote site. Also, the output file must be local, **grdinfo** does not send data to a remote site.

Data collections

Each **grdinfo** option collects a different set of data. Table 3-2 shows a representative data set for each option. Additional data may be added/deleted over time. The **grdinfo -all** command attempts to collect every set of system and media card data included in the table.

*Table 3-2. System and media card data collected by **grdinfo** command options*

Data request	Data source
ATM OC-3c information: **grdinfo -card=**	maint 2, maint 3 1 0/maint 3 1 1, maint 3 2 0/maint 3 2 1, maint 3 3 0/maint 3 3 1, maint 4 0/maint 4 1, maint 5, maint 6, maint 8, maint 10, maint 13 0/maint 13 1, maint 113 0/maint 113 1, maint 14 0 /maint 14 1, maint 15 0/maint 15 1, maint 110, maint 118, maint 20 0/maint 20 1, maint 56/maint 156, maint 58/maint 158, maint 62 300/maint 162 300
ATM OC-12c information: **grdinfo -card=**	maint 2, maint 3, maint 4, maint 5, maint 6, maint 8, maint 10, maint 11, maint 12, maint 15, maint 46, maint 56, maint 156, maint 105, maint 109, maint 110, maint 111, maint 112, maint 116, maint 117, maint 120
HSSI maint information: **grdinfo -card=**	maint 2, maint 3, maint 4, maint 5, maint 6, maint 106, maint 8, maint 108, maint 12, maint 112, maint 56, maint 156, maint 58, maint 158
SONET maint information: **grdinfo -card=**	maint 2, maint 3, maint 4, maint 5, maint 6, maint 8, maint 12, maint 56, maint 156, maint 108

Table 3-2. System and media card data collected by ***grdinfo*** *command options (continued)*

Data request	Data source
Ethernet maint information: **grdinfo -card=**	maint 2, maint 3, maint 4 0/maint 4 1/maint 4 2/maint 4 3/maint 4 4/ maint 4 5/maint 4 6/maint 4 7, maint 5, maint 6, maint 8 0/maint 8 1/maint 8 2/maint 8 3/maint 8 4/maint 8 5/maint 8 6/maint 8 7, maint 12, maint 112, maint 56, maint 156
FDDI maint information: **grdinfo -card=**	maint 2, maint 3, maint 4, maint 5, maint 6, maint 56, maint 58, maint 60 0/maint 60 1/maint 60 2/maint 60 3, maint 61 0/maint 61 1/maint 61 2/maint 61 3, maint 62 0/maint 62 1/maint 62 2/maint 62 3, maint 63 0/maint 63 1/maint 63 2/maint 63 3, maint 70 56, maint 70 58, maint 70 8
HIPPI maint information: **grdinfo -card=**	maint 132
Frame Relay information: **grdinfo -frame**	- system configuration and status - link configuration and status - PVC configuration and status - board status - PVC statistics - interface configuration and status
Bridging information: **grdinfo -bridge**	- output from **brinfo -all** - statistics from **brstat** output
Control board (RMS) information **grdinfo -system**	- software version, **getver** output (GRF), **sysctl** contents, kernel messages - output from **ifconfig -a**, **netstat -in**, **netstat -rn**, **netstat -a** - cardq information from media cards via **cardq -v** - ARP information from media cards via **grarp -a** - number of routes in each media card - card counter output via **grstat** - **mount** command output, external device data via **csconfig -a** - output from **vmstat -sm**, process information via **ps -lam** - **fstat** output - mounted file system data via **df**
System data: **grdinfo -conf**	- all `/etc/*.conf` files - complete GateD configuration file (using **gdexpand**) - all Card, System, Dump, and Load profiles in `/etc/prof`
System logs: **grdinfo -log**	- all.files in local `/var/log/*` (including compressed files)
System dump data: **grdinfo -dump**	- media card dumps `/var/portcards/grdump.*` - utility dumps `/var/run/*.core/` - other dumps `/var/tmp/*.core/` - mini dumps `/var/tmp/*.mcore/` - kernel dump, if available, `/var/crash/grsavecore.out`

Using grdinfo

This section describes how to work with **grdinfo** and provides several examples of file output.

A brief overview of **grdinfo** usage:

- execute a **grdinfo** command, output goes to `/var/tmp/grdinfo/grdinfo.tar.gz`
- ftp the tar.gz file to technical support
  or
  move it off the GRF
- unzip (**gunzip**) and tar the `grdinfo.tar.gz` file
- examine the information, save it to external storage if needed
- delete files from `/var/tmp/grdinfo` to maintain system file space

You can apply more than one option to the **grdinfo** command. This example collects the system configuration files and the **maint** command information from the media card in slot 9:

grdinfo -conf -card=9

Starting up

The **grdinfo** command executes from the UNIX shell. Entering just the command causes a brief description of command options to be displayed. The **help** display uses abbreviated spellings for several options:

```
super> sh
# grdinfo

Usage: grdinfo [options]
 Options: [defaults are in brackets after descriptions]
   -help       prints this usage message
   -sys        collect GRF system info [ off ]
   -conf       collect configuration files [ off ]
   -log        collect log files [ off ]
   -dr         collect dynamic routing information [off]
   -br         collect bridging information [ off ]
   -fr         collect frame relay information [ off ]
   -dump       collect system dumps[ off ]
   -card=<# | all>    collect maint command information
                      from cards [ off ]
   -quiet        do not print any progress messages
   -version      print the version of this file
   -ofile=info_file     have the information collected in this file
                        default path: var/tmp/grdinfo/grdinfo.tar
   -all          collect all debug information
```

The **grdinfo** man page is available and contains more details about each option.

Example: grdinfo -config

To obtain the system configuration information, use the **-config** option:

```
# grdinfo -config
```

You immediately see this message as **grdinfo** begins to collect the files. The response may take a minute or two, depending upon system activity level and the number of files to be collected. When **grdinfo** is finished, the prompt returns.

```
Output from /usr/nbin/grdinfo is going to file
/var/tmp/grdinfo/grdinfo.tar.gz
#
```

Change directory to the default output path and list the contents:

```
# cd /var/tmp/grdinfo
# ls -l
total 26
-rw-r--r--  1 root  wheel  26611 Apr  8 14:09 grdinfo.tar.gz
```

Technical support staff may ask that you place the `grdinfo.tar.gz` file on your ftp site for retrieval or that you ftp it to an Ascend site.

If it will not be transferred, move it off the GRF and then unzip the file:

```
# gunzip *.gz
# ls -l
total 124
-rw-r--r--  1 root  wheel  122880 Apr  8 14:09 grdinfo.tar
```

When you tar the file, you see the files that were collected:

```
# tar xvf grdinfo.tar
aitmd.conf
bridged.conf
dm.conf
filterd.conf
gated.conf
grarp.conf
grass.conf
gratm.conf
grclean.conf
grclean.logs.conf
grfr.conf
grifconfig.conf
gritd.conf
grlamap.conf
grppp.conf
grroute.conf
inetd.conf
login.conf
man.conf
pccard.conf
snmpd.conf
syslog.conf
/var/tmp/grdinfo/grdinfo.140957.info
/var/tmp/grdinfo/grdinfo.140957.errors
tar: tar vol 1, 25 files, 122880 bytes read.
```

Use a UNIX editor to access the configuration files.

Clean up /var/tmp/grdinfo

The `grdinfo.tar.gz` file is always overwritten by the output of subsequent **grdinfo** commands, multiple iterations do not accumulate.

The other files obtained from a **grdinfo** command remain in the `/var/tmp/grdinfo` directory. It is good practice to clean up `/var/tmp` when you are finished with the information, especially if the data files are large.

When you have finished the debug session, or `/var/tmp` has grown to 3–4MB, remove the current `grdinfo` directory:

```
# cd  /var/tmp
# rm -r grdinfo
# ls
bridged.trace   gated_bgp       gated_parse     vi.recover
#
```

Example: grdinfo -card

Collect card statistics for the Ethernet card in slot 9:

```
# grdinfo -card=9
Output from /usr/nbin/grdinfo is going to file
/var/tmp/grdinfo/grdinfo.tar.gz
# cd /var/tmp/grdinfo
# ls -l
total 3
-rw-r--r--  1 root  wheel  2738 Apr  8 14:27 grdinfo.tar.gz
```

Move the `grdinfo.tar.gz` file off the GRF:

```
# mv grdinfo.tar.gz _____________
# gunzip *.gz
# tar xvf grdinfo.tar
/var/tmp/grdinfo/grdinfo.142749.info
/var/tmp/grdinfo/grdinfo.142749.errors
tar: tar vol 1, 2 files, 30720 bytes read.
# ls -l
total 50
-rw-r--r--  1 root  wheel    242 Apr  8 14:27 grdinfo.142749.errors
-rw-r--r--  1 root  wheel  19322 Apr  8 14:27 grdinfo.142749.info
-rw-r--r--  1 root  wheel  30720 Apr  8 14:27 grdinfo.tar
```

Use a UNIX editor to read the `.info` file, only a portion is shown here:

```
# vi grdinfo.142749.info

============================================================
This file contains data collected by grdinfo while running.
============================================================

grdinfo is being run on a GRF.
Not collecting configuration files
Not collecting log files
Not collecting system information
Not collecting frame relay information
```

```
Not collecting transparent bridging information
Getting maint information from cards : args -> 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Card 09 / ETHER.
============================================================
Maint 2
[RX] Ethernet Port Card Hardware and Software Revisions:
[RX] ===============================================
[RX] HW:
[RX]    Power-On Self-Test (POST) result code: 0x0.
[RX]    Ethernet Media Board HW Rev: 0x4, with 4M Sram.
[RX]    Ethernet Xilinx Version: 0x0.
[RX]    SDC Board HW Rev: 0xe  (SDC2).
[RX]            SDC2 Combus Xilinx version: 0x6.
[RX]            SDC2 Switch Transmit Xilinx version: 0x5.
[RX]            SDC2 Switch Receive Xilinx version: 0x0.
[RX] SW:
[RX]Ethernet Code Version: 1_4_7,Compiled Sat Apr 4 13:00:33 CST1998
[RX]            in directory: /test/A1_4_7_9/ether/rx.
[RX] Library Version: 1.1.0.0,Compiled on Sat Apr 4 2:55:18 CST 1998
------------------------------------------------------------
Maint 3
[RX]
[RX] Ethernet Configuration and Status.
[RX] Up Time: 0 days, 0:21:53
[RX] Free Memory: 3120508
[RX] Port: [MAC.Address...] Link Method...Configuration....->Partner
[RX] ------------------------------------------------------------------
[RX] 0 : [00:c0:80:89:08:65] Down Negotiate
[RX] 1 : [00:c0:80:89:08:66] Down Negotiate
[RX] 2 : [00:c0:80:89:08:67] Down Negotiate
[RX] 3 : [00:c0:80:89:08:68] Down Negotiate
[RX] 4 : [00:c0:80:89:08:69] Up   Negotiate 100/FDX/Multicast-> 100/FDX
[RX] 5 : [00:c0:80:89:08:6a] Down Negotiate
[RX] 6 : [00:c0:80:89:08:6b] Down Negotiate
[RX] 7 : [00:c0:80:89:08:6c] Down Fixed-Neg  10/HDX/Disabled -> ?/?
-------------------------------------------------------------------
Maint 4 0
[RX]                    Media Statistics
[RX] input:
[RX] Port       Bytes              Packets         Errors    Discards
[RX] -------------------------------------------------------------
[RX]0 0000000000000000000 00000000000000000000 0000000000 0000000000
[RX]
[RX] Port 0:
[RX]    Unsupported type: 0
[RX]    CRC errors:       32896
[RX]    Runt errors:      33152
[RX]    Oversize Frames:  128
[RX]    Alignment errors: 33152
[RX]    Out of buffers:   0
[RX]
[RX] output:
[RX] Port       Bytes              Packets        Discards
grdinfo.142749.info: unmodified: line 559.
```

(This is a partial listing of the collected information.)

Threshpoll tracking utility

threshpoll enables the network administrator to track specific traffic items for a logical interface. The user can specify a threshold for each of the traffic items counted. A trap can be sent to an SNMP gateway according to whether the threshold is reached, exceeded, or not reached. Counts are also logged, but not to an NFS system, only to an external PCMCIA device.

Traffic items that can be counted include octets received and transmitted, and dropped packets and errors on the receive and transmit sides.

threshpoll checks the specified statistics for each item at five-minute or other user-specified intervals and then compares the latest value against a specified threshold. The results are sent to **threshpoll** log files in `/var/log`.

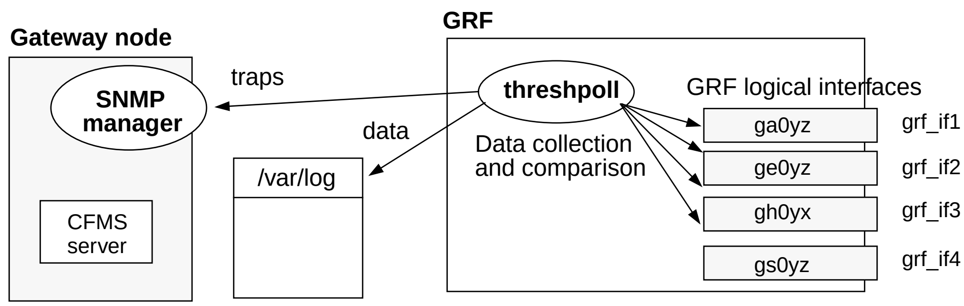


Figure 3-5. Diagram of threshpoll data and trap functions

Figure 3-5 illustrates a basic overview of **threshpoll** data collection and SNMP trap activity.

The purpose of the **threshpoll** utility is to gather information that will help the user see what is happening at the level of the logical interface. This type of information can help determine where to redirect traffic or fine-tune a congested area. For example, you can determine if a specific GRF interface is receiving more than or less than a certain number of octets in a typical 5-minute period, and redirect as needed.

What can be monitored

These are the some of the items monitored, along with their equivalent SNMP names:

- length of time system has been up	`sysUpTime`
- octets received and transmitted	`ifInOctets` and `ifOutOctets`
- unicast packets received and transmitted	`ifInUcastPkts` and `ifOutUcastPkts`
- packets dropped on receive side	`ifInDiscards`
- packets dropped on transmit side	`ifOutDiscards`
- errors occurring on receive side	`ifInErrors`
- errors occurring on transmit side	`ifOutErrors`

Poll group

A poll group is a list of items you want to monitor for a particular logical interface.

A poll group is defined in an `/etc/threshpollPoll.`*`number`* configuration file where *number* indicates the instance of a **threshpoll** collection task.

For example, you might want to create one instance of **threshpoll** to collect an interfaces's incoming (receive) statistics, and a second instance to collect the interface's transmit side statistics. Instances three and four could collect those statistics for a different logical interface. An instance contains a unique set of poll groupss. **threshpoll** will perform a separate data collection for each defined instance.

An item entry consists of the item's unique OID, and the identity of the interface this item will be collected for. The interface is identified by its `if index` as assigned in the **netstat -in** output. where each interface is identified as *`linkx`*.

Here is the full list of items **threshpoll** can monitor:

SNMP name	OID	.if Index
`sysUpTime`	`1.3.6.1.2.1.1.3.0` *(required entry)*	
`ifInOctets`	`1.3.6.1.2.1.2.2.1.10.`*`linkx`*	
`ifInUcastPkts`	`1.3.6.1.2.1.2.2.1.11.`*`linkx`*	
`ifInDiscards`	`1.3.6.1.2.1.2.2.1.13.`*`linkx`*	
`ifInErrors`	`1.3.6.1.2.1.2.2.1.14.`*`linkx`*	
`ifOutOctets`	`1.3.6.1.2.1.2.2.1.16.`*`linkx`*	
`ifOutUcastPkts`	`1.3.6.1.2.1.2.2.1.17.`*`linkx`*	
`ifOutDiscards`	`1.3.6.1.2.1.2.2.1.19.`*`linkx`*	
`ifOutErrors`	`1.3.6.1.2.1.2.2.1.20.`*`linkx`*	
`grGatedStatus`	`1.3.6.1.4.1.1080.1.1.1.6`	
`ifHCInOctets`	`1.3.6.1.2.1.31.1.1.1.6.`*`linkx`*	
`ifHCInUcastPkts`	`1.3.6.1.2.1.31.1.1.1.7.`*`linkx`*	
`ifHCInMulticastPkts`	`1.3.6.1.2.1.31.1.1.1.8.`*`linkx`*	
`ifHCInBroadcastPkts`	`1.3.6.1.2.1.31.1.1.1.9.`*`linkx`*	
`ifHCOutOctets`	`1.3.6.1.2.1.31.1.1.1.10.`*`linkx`*	
`ifHCOutUcastPkts`	`1.3.6.1.2.1.31.1.1.1.11.`*`linkx`*	
`ifHCOutMulticastPkts`	`1.3.6.1.2.1.31.1.1.1.12.`*`linkx`*	
`ifHCOutBroadcastPkts`	`1.3.6.1.2.1.31.1.1.1.13.`*`linkx`*	

In any poll group, you must always specify `sysUpTime` as the first item.

Trap types and trap variables

For each item in a poll group, you must include the specific trap type you want sent when the specified threshold is broken.

These traps send no variables:

`grGatedDown`	35
`grSnmpReset`	36

The following traps send two variables:

`grIfInOctetsHigh`	19
`grIfInOctetsLow`	20

`grIfOutOctetsHigh`	21
`grIfOutOctetsLow`	22
`grIfInUcastPktsHigh`	23
`grIfInUcastPktsLow`	24
`grIfOutUcastPktsHigh`	25
`grIfOutUcastPktsLow`	26
`grIfInErrorsHigh`	27
`grIfInErrorsLow`	28
`grIfOutErrorsHigh`	29
`grIfOutErrorsLow`	30
`grIfInDiscardsHigh`	31
`grIfInDiscardsLow`	32
`grIfOutDiscardsHigh`	33
`grIfOutDiscardsLow`	34

The variables are:

`grTPPreviousCount`

- this is the value of the item at the previous time it was polled.

`grTPCurrentCount`

- this is the value of the item at the time it was most recently polled.

The following traps send four variables:

`grIfHCInOctetsHig`	37
`grIfHCInOctetsLo`	38
`grIfHCInUcastPktsHigh`	39
`grIfHCInUcastPktsLow`	40
`grIfHCInMulticastPktsHigh`	41
`grIfHCInMulticastPktsLow`	42
`grIfHCInBroadcastPktsHigh`	43
`grIfHCInBroadcastPktsLow`	44
`grIfHCOutOctetsHigh`	45
`grIfHCOutOctetsLow`	46
`grIfHCOutUcastPktsHigh`	47
`grIfHCOutUcastPktsLow`	48
`grIfHCOutMulticastPktsHigh`	49
`grIfHCOutMulticastPktsLow`	50
`grIfHCOutBroadcastPktsHigh`	51
`grIfHCOutBroadcastPktsLow`	52

The variables are:

`grTPPreviousCount`
- this is the lower 32 bits of the item being polled the last time it was polled.

`grTPCurrentCount`
- this is the lower 32 bits of the item being polled at the time it was most recently polled.

`grTPPreviousCount`:
- the upper 32 bits of the item being polled the last time it was polled.

`grTPCurrentCount`
- the upper 32 bits of the item being polled at the time it was most recently polled.

Threshpoll logging

All **threshpoll** logging is done in `/var/log`.

Caution: Do not run **threshpoll** if the GRF is configured for NFS logging. Only run **threshpoll** if the GRF is configured to log to a PCMCIA device.

After you start **threshpoll**, you can look at the `/var/log` directory and see the error log file and the message log directory:

```
# cd /var/log
# ls
NSM.threshpoll.errlog.06.05.98.15:47:24.1        (a file)
NSM.threshpoll.msglog.06.05.98.15:47:24.1        (a directory)
    .
    .
    .
```

Error logs report errors that occur while **threshpoll** is running. The message directory contains a report of item counts at each poll interval. A new log file is created at each specified interval, but since the previous file is not overwritten, log files can accumulate rapidly.

Caution: The smallest recommended interval is 60 seconds.

Because **grclean** does not archive and remove **threshpoll** log files, another mechanism is provided. This mechanism ages and removes a **threshpoll** log file after it is 15 minutes old.

The CFMS configuration management utility can retrieve copies of these log files via the **cfms bulkstat** command. (Note that this command is not available in the 1.4.8 release.)

Error log

Errors are logged to:

```
/var/log/NSM.threshpoll.errlog.M.D.Y.h:m:s.i
```

where `M.D.Y.h:m:s.i` equals:

Month.Day.Year.hour:minute:second:instance-running)

This example reports error 32:

```
# more NSM.threshpoll.errlog.12.15.97.12:35:13.100

Starting Application threshpoll @ 12.15.97.12:35:13
log directory = /var/log/NSM.threshpoll.msglog.12.15.97.12:35:13.100
device configuration file = /etc/threshpollDevice.100
poll configuration file = /etc/threshpollPoll.100
threshold configuration file = /etc/threshpollThreshold.100
polling interval = 5
/usr/nbin/threshpoll -x 1
Mon Dec 15 12:36:38 1997, error = 32, Proc: Sigterm received.
```

The error log files should be monitored by **grclean** to make sure they do not get too large. Add the following lines to `/etc/grclean.logs.conf` in the “`of some interest`” section:

```
size = 25000
logfile=/var/log/NSM.threshpoll.errlog.*
```

Message log

The message files contain the counts:

```
# cd NSM.threshpoll.msglog.06.05.98.15:47:24.1
# ls
entry.Jun.05.1998.20:47:24      entry.Jun.05.1998.20:48:24

cat entry.Jun.05.1998.20:47:24

Jun 05 1998 20:47:24|box1|box1_ga000|24|3|282796|0|0|0|END
Jun 05 1998 20:47:24|box1|box1_ga0180|27|3|282796|0|0|0|END
Jun 05 1998 20:47:25|box1|box1_ga020|26|3|282796|0|0|0|END
Jun 05 1998 20:47:25|box1|box1_go060|46|3|282796|0|0|0|END
Jun 05 1998 20:47:25|box1|box1_ge070|34|3|282797|0|0|0|END
```

Note that **grclean** cannot monitor the size of the files in the message directory.

Reading the message logs

Figure 3-6 defines the fields in a message log file with three items specified in the `/etc/threshpollPoll.`*x* file:

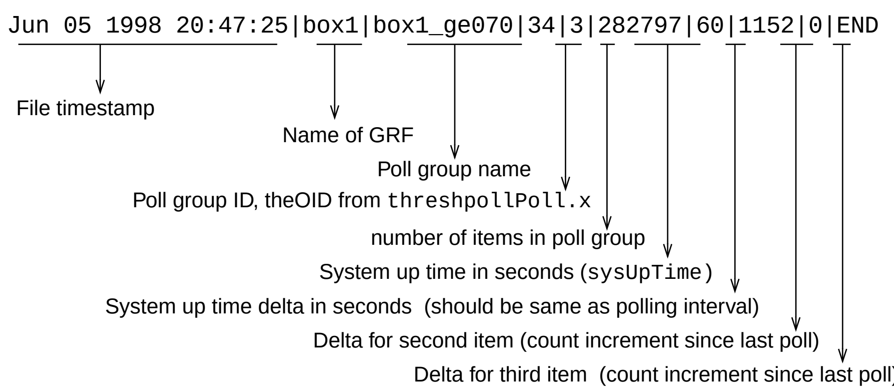


Figure 3-6. Definitions of fields in threshpoll messages file

System up time is required as the first item counted and the values are displayed in seconds. Values for the remaining items vary with the type of item specified. END defines the close of each log entry.

Configuration steps

There are four steps to configure **threshpoll**. Once established, it runs automatically and can be set to run in the background. An instance of threshpoll can be thought of as a data "run." It can collect data based on media type such as all HSSI interfaces, traffic type such as incoming or dropped packets, or a combination of those factors. For each instance of **threshpoll** you want to run, you must create a set of three files.

– `/etc/threshpollPoll.`*`instance`*

– `/etc/threshpollThreshold.`*`instance`*

– `/etc/threshpollDevice.`*`instance`*

These files each have a related purpose:

– define all data to collect on the "run", and the related traps to send

– define the thresholds to measure the data against

– define list of data to collect currently

1 Create a poll file for each instance of **threshpoll** you want to run.

You may want to have one instance to cover all the ATM OC-3c cards, another to cover all HSSI cards, or one to cover all the error counts and packets dropped for all cards.

Inside the poll file are poll groups. A poll group usually contains the list of items you want to monitor for a particular interface, and specifies which trap type or types are to be sent for each item.

A poll file name is in the form:
`/etc/threshpollPoll.`*`instance`* file

where *`instance`* is the number you assign an instance of **threshpoll** that will run to collect a specific list of items. *`instance`* is always specified as a number, never as a text string.

The items you specify in this file are logged into the message directory in
`/var/log/NSM.threshpoll.msglog.M.D.Y.h:m:s.i`

For example: `NSM.threshpoll.msglog.06.05.98.15:47:24.1`

2 Define the threshold for each item listed in each poll group.

Thresholds are defined in the `/etc/threshpollThreshold.`*`instance`* file

where *`instance`* is the number you assigned the instance of **threshpoll** associated with the poll file and poll groups defined in step 2.

3 Specify the poll groups currently enabled.

The groups are listed in the `/etc/threshpollDevice.`*`instance`* file

where *`instance`* is the number you assigned the instance of **threshpoll** associated with the poll file and poll groups defined in step 2.

Using this file, you can keep many poll groups defined but have only those of current interest actually running.

4 Last, specify the gateway SNMP manager to which you want the traps to be sent.

Edit the `/etc/snmpd.conf` file.

Threshpoll configuration example

The example takes you through the configuration tasks needed to monitor and track one ATM interface, `ga038`, and one HSSI interface, `gs0ab`.

Two poll groups will be used to track incoming and outgoing traffic on the ATM interface. A third poll group will be used to track errors on the HSSI interface.

Before you start - getting if index numbers

You need to know the `if index` number for each logical interface that will be tracked. The `if index` numbers are obtained from the **netstat -in** output. This excerpt from **netstat -in** shows the `if index` for several logical interfaces. Their `if index` numbers are 27, 26, 23, and 51, respectively:

```
# netstat -in
Name    Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
                                     .
                                     .
                                     .
ga0180  9180  <link27>    00:c0:80:f8:34:80    0    0        0     0     0
ga0180  9180  205.1.11    205.1.11.156         0    0        0     0     0
ga038   9180  <link26>    00:c0:80:f7:b2:00    0    0        0     0     0
ga038   9180  205.1.12    205.1.12.156         0    0        0     0     0
ga0380  9180  <link23>    00:c0:80:fb:10:80    0    0        0     0     0
ga0380  9180  205.1.13    205.1.13.156         0    0        0     0     0
gs0ab   9180  <link51>    00:00:00:00:00:00    0    0        0     0     0
gs0ab   9180  208.1.10    208.1.10.156         0    0        0     0     0
```

ATM interface `ga038` has an if index of 26, the HSSI `gs0ab` interface has an index of 51.

Choose items for each poll group

These are the items in the ATM receive side poll group, its transmission rates requires the HC (high count, 64-bit) items and traps:

```
sysUpTime                      1.3.6.1.2.1.1.3.0
ifHCInOctets                   1.3.6.1.2.1.31.1.1.1.6.link26
ifHCInUcastPkts                1.3.6.1.2.1.31.1.1.1.7.link26
ifHCInMulticastPkts            1.3.6.1.2.1.31.1.1.1.8.link26
ifHCInBroadcastPkts            1.3.6.1.2.1.31.1.1.1.9.link26
```

These are the items in the ATM transmit side poll group, its transmission rates requires the HC (high count, 64-bit) items and traps:

```
sysUpTime                      1.3.6.1.2.1.1.3.0
ifHCOutOctets                  1.3.6.1.2.1.31.1.1.1.10.link26
ifHCOutUcastPkts               1.3.6.1.2.1.31.1.1.1.11.link26
ifHCOutMulticastPkts           1.3.6.1.2.1.31.1.1.1.12.link26
ifHCOutBroadcastPkts           1.3.6.1.2.1.31.1.1.1.13.link26
```

These are the items in the HSSI interface poll group.

```
sysUpTime                      1.3.6.1.2.1.1.3.0
ifInErrors                     1.3.6.1.2.1.2.2.1.14.link51
ifOutErrors                    1.3.6.1.2.1.2.2.1.20.link51
```

1. Create a poll file - threshpollPoll.*instance* file

Poll files list the items (in poll groups) you want monitored and the trap type you want sent when the threshold is broken.

A poll group is defined in an `/etc/threshpollPoll.instance` configuration file where *instance* indicates the instance of a **threshpoll** collection run.

Task: Create one poll file containing three poll groups: `/etc/threshpollPoll.33`

ATM `ga038` interface needs two:
- collect incoming traffic statistics
- collect outgoing traffic statistics

HSSI `gs0ab` interface needs one:
- collect all error statistics

Here is how the information in a poll group entry is structured:

```
OID index:poll group name:oid|general trap-type|specific trap-type:
                          oid|general trap-type|specific trap-type:
                          oid|general trap-type|specific trap-type:
                          END
```

where:

`OID index`	= same as *instance* number
`poll group name`	= text string identifying this instance
`oid`	= SNMP object ID of the item you wish to poll, includes `if index` from **netstat** (*linkx*)
`general trap-type`	= should always be 6
`specific trap-type`	= the number given a certain type of trap

Poll group information

Here is the poll group information to collect the ATM receive side statistics:

```
33:atm_38rx:.1.3.6.1.2.1.1.3.0|6|0:  # system uptime requirement, no if index needed
               .1.3.6.1.2.1.31.1.1.1.6.26|6|37:
               .1.3.6.1.2.1.31.1.1.1.7.26|6|39:
               .1.3.6.1.2.1.31.1.1.1.8.26|6|41:
               .1.3.6.1.2.1.31.1.1.1.9.26|6|43:
                END
```

Here is the poll group information to collect the ATM transmit side statistics:

```
33:atm_38tx:.1.3.6.1.2.1.1.3.0|6|0:   # system uptime requirement, no if index
               .1.3.6.1.2.1.31.1.1.1.10.26|6|45:
               .1.3.6.1.2.1.31.1.1.1.11.26|6|47:
               .1.3.6.1.2.1.31.1.1.1.12.26|6|49:
               .1.3.6.1.2.1.31.1.1.1.13.26|6|51:
                END
```

Here is the poll group information to collect the HSSI error statistics. The file name is `/etc/threshpollPoll.300`:

```
33:hssi_ab:.1.3.6.1.2.1.1.3.0|6|0:   # system uptime requirement, no if index
               .1.3.6.1.2.1.2.2.1.14.51|6|27:
               .1.3.6.1.2.1.2.2.1.20.51|6|29:
                END
```

However, in the `/etc/threshpollPoll.33` file, you remove the line breaks and the group entries actually look like these:

```
33:atm_38rx:..1.3.6.1.2.1.1.3.0|6|0:.1.3.6.1.2.1.31.1.1.1.6.link26|6|37
:.1.3.6.1.2.1.31.1.1.1.7.link26|6|39:.1.3.6.1.2.1.31.1.1.1.8.link26|6|4
1:.1.3.6.1.2.1.31.1.1.1.9.link26|6|43:END
33:atm_38tx:.1.3.6.1.2.1.1.3.0|6|0:.1.3.6.1.2.1.31.1.1.1.10.link26|6|45
:.1.3.6.1.2.1.31.1.1.1.11.link26|6|47:.1.3.6.1.2.1.31.1.1.1.12.link26|6
|49:.1.3.6.1.2.1.31.1.1.1.13.link26|6|51:END
33:hssi_ab:.1.3.6.1.2.1.1.3.0|6|0:.1.3.6.1.2.1.2.2.1.14.link51|6|27:.1.
3.6.1.2.1.2.2.1.20.link51|6|29:END
```

Colons (:) separate the individual items. Bars (|) separate item and trap parts. Do not use spaces to separate items, let the line wrap.

Here is a copy of the `/etc/threshpollPoll.1.template` file with a sample entry:

```
# vi /etc/threshpollPoll.1.template

###################################################################
## Poll group list
##
## Format
## OID index:poll group name:object id to poll|general trap-type|
## specific trap-type:
## object id to poll|general trap-type|specific trap-type:etc:END
##
## Maximum number of oids per poll group is 23.
##
##   Pollable items:
##  .1.3.6.1.2.1.1.3.0          =   sysUpTime
##  .1.3.6.1.2.1.2.2.1.10.x     =   ifInOctets
##  .1.3.6.1.2.1.2.2.1.11.x     =   ifInUcastPkts
##  .1.3.6.1.2.1.2.2.1.13.x     =   ifInDiscards
##  .1.3.6.1.2.1.2.2.1.14.x     =   ifInErrors
##  .1.3.6.1.2.1.2.2.1.16.x     =   ifOutOctets
##  .1.3.6.1.2.1.2.2.1.17.x     =   ifOutUcastPkts
##  .1.3.6.1.2.1.2.2.1.19.x     =   ifOutDiscards
##  .1.3.6.1.2.1.2.2.1.20.x     =   ifOutErrors
##  .1.3.6.1.4.1.1080.1.1.1.6.0 =   grGatedStatus
##  .1.3.6.1.2.1.31.1.1.1.6.x   =   ifHCInOctets
##  .1.3.6.1.2.1.31.1.1.1.7.x   =   ifHCInUcastPkts
##  .1.3.6.1.2.1.31.1.1.1.8.x   =   ifHCInMulticastPkts
##  .1.3.6.1.2.1.31.1.1.1.9.x   =   ifHCInBroadcastPkts
##  .1.3.6.1.2.1.31.1.1.1.10.x  =   ifHCOutOctets
##  .1.3.6.1.2.1.31.1.1.1.11.x  =   ifHCOutUcastPkts
##  .1.3.6.1.2.1.31.1.1.1.12.x  =   ifHCOutMulticastPkts
##  .1.3.6.1.2.1.31.1.1.1.13.x  =   ifHCOutBroadcastPkts
##
```

```
##  Available specific trap types:
##
##   grIfInOctetsHigh          19
##   grIfInOctetsLow           20
##   grIfOutOctetsHigh         21
##   grIfOutOctetsLow          22
##   grIfInUcastPktsHigh       23
##   grIfInUcastPktsLow        24
##   grIfOutUcastPktsHigh      25
##   grIfOutUcastPktsLow       26
##   grIfInErrorsHigh          27
##   grIfInErrorsLow           28
##   grIfOutErrorsHigh         29
##   grIfOutErrorsLow          30
##   grIfInDiscardsHigh        31
##   grIfInDiscardsLow         32
##   grIfOutDiscardsHigh       33
##   grIfOutDiscardsLow        34
##   grGatedDown               35
##   grSnmpReset               36
##   grIfHCInOctetsHigh        37
##   grIfHCInOctetsLow         38
##   grIfHCInUcastPktsHigh     39
##   grIfHCInUcastPktsLow      40
##   grIfHCInMulticastPktsHigh  41
##   grIfHCInMulticastPktsLow   42
##   grIfHCInBroadcastPktsHigh  43
##   grIfHCInBroadcastPktsLow   44
##   grIfHCOutOctetsHigh       45
##   grIfHCOutOctetsLow        46
##   grIfHCOutUcastPktsHigh    47
##   grIfHCOutUcastPktsLow     48
##   grIfHCOutMulticastPktsHigh 49
##   grIfHCOutMulticastPktsLow  50
##   grIfHCOutBroadcastPktsHigh 51
##   grIfHCOutBroadcastPktsLow  52
##
## This device list is to be polled once every 5 minutes
##
####################################################################
1:grf_if1:.1.3.6.1.2.1.1.3.0|6|0:.1.3.6.1.2.1.2.2.1.11.1|6|23:.1.3.6.1.
2.1.2.2.1.17.1|6|0:.1.3.6.1.2.1.2.2.1.14.1|6|27:.1.3.6.1.2.1.2.2.1.20.1
|6|29:.1.3.6.1.2.1.2.2.1.10.1|6|19:.1.3.6.1.2.1.2.2.1.16.1|6|21:END
```

2. Specify thresholds - threshpollThreshold.*instance* file

Specify a threshold for each item listed in a poll group. You must have a threshold entry for each of the poll groups you have configured. A group's threshold entry must include a threshold value for each item in the group.

Thresholds are specified in the `/etc/threshpollThreshold.`*`instance`* file where *`instance`* is a number that identifies a particular instance of a **threshpoll** collection run.

Here is how a `threshpollThreshold.`*`instance`* entry is structured:

```
poll group name:threshold operation and value:
                threshold operation and value:
                threshold operation and value:
                threshold operation and value:
                END
```

where:

`poll group name`	=	the name of the poll group to which the thresholds apply
`threshold operation`	=	specifies the action taken against the threshold level, expressed as less than, greater than, equal to, and so on
`value`	=	the actual size of the threshold expressed in an integer

ATM receive side poll group

Assign each item in the group (except system up time) an operation and a threshold value, enter I for system up time:

```
sysUpTime               - I
ifHCInOctets            - > 4000
ifHCInUcastPkts         - > 2500
ifHCInMulticastPkts     - < 1000
ifHCInBroadcastPkts     - > 1000
```

Use the file structure to assemble an entry that assigns thresholds to group 100:

```
atm_38rx:I:
              >4000:
              >2500:
              <1000:
              >1000:
              END
```

However, in the file you remove the spaces between the entries and it actually looks like this:

```
atm_38tx:I:>4000:>2500:<1000:>1000:END
```

Colons (:) separate the individual items. Do not use spaces to separate items, let the line wrap.

Here is the threshold file covering all three poll groups for our sample configuration:

```
# /etc/threshpollThreshold.33
atm_38tx:I:>4000:>2500:<1000:>1000:END
atm_38rx:I:>4000:>2500:<1000:>1000:END
hssi_ab:I:>500:>500:END
```

Here is a copy of an `/etc/threshpollThreshold.1.template` file with several entries:

```
# vi /etc/threshpollThreshold.1.template

###################################################################
##
## Thresholding List
##
## Format:
##   poll group name:threshold operation and value:
##   threshold operation and value:etc:END
##
##   Valid threshold operations are:
##   NA - Not Applicable, snmp traps will not be sent
##        but data will be logged.
##   I  - Informational, snmp traps will be sent and
##        data will also be logged.
##   <  - less than, snmp traps will be sent if the
##        condition is met and data will be logged.
##   >  - greater than, snmp traps will be sent if the
##        condition is met and data will be logged.
##   =  - equal to, snmp traps will be sent if the
##        condition is met and data will be logged.
##   != - not equal to, snmp traps will be sent if the
##        condition is met and data will be logged.
##
## This device list is to be polled once every 5 minutes
##
#####################################################################
gated:I:>0:END
grf_if1:I:>600:I:>0:>0:>2000:>1500:END
grf_if2:I:>20:I:>0:>0:>500:>500:END
grf_if3:I:>20:I:>0:>0:>3000:>3000:END
grf_if5:I:>5:I:>0:>0:>0:>0:END
```

3. Enable monitoring of poll groups - threshpollDevice.*instance* file

Turn **threshpoll** on or off for each individual poll group in the `threshpollDevice.`*`instance`* file where *`instance`* is a number that identifies a particular **threshpoll** instance.

Here is how an entry is structured:

```
gateway:enterprise oid:device:community:poll group|threshold flag:END
```

where:

`gateway**`	= where the trap is to be sent (ignored, defined in `snmpd.conf`)
`enterprise oid`	= vendor object identifier (always .1.3.6.1.3.1.1080 for a GRF)
`device` **	= name of the device to be polled (entry is ignored)
`community`	= community to be set in the trap, typically "`public`"
`poll group`	= defined poll group name
`threshold flag`	= turns monitoring on (0) or off (1) for the named group

Use the structure to enable or disable **threshpoll** for each group defined in this instance of threshpoll. In this example, only the HSSI poll group is disabled:

```
## /etc/threshpollDevice.33
gateway:.1.3.6.1.3.1.1080:site_admin:public:atm_38rx|0:atm_38tx
|0:hssi_ab|1:END
```

Colons (:) separate the individual items. A bar (|) separates the group name and threshold on/off setting. Do not use spaces to separate items, let the line wrap.

Here is a copy of the `/etc/threshpollDevice.1.template` file with one sample entry:

```
# vi threshpollDevice.1.template

######################################################################
## Device list
## Lists all the devices and the polling groups to be polled.
##
## Format:
## gateway:enterprise oid:device:community:poll group|threshold flag:END
##
## gateway      - where the trap is to be sent
## enterprise   - vendor object identifier
## device       - device to be polled.
## community    - community
## poll group   - referencing the oids to be polled in the
##                poll group file.
## thresholding - whether to perform thresholding on the poll
##                group or not. 0 - OFF, 1 - ON.
##
## In addition, the threshold value of each device's poll group
## object identifiers is initialized to the default value found
## in the poll group file.
##
## This device list is to be polled once every 5 minutes
#######################################################################
testbox:.1.3.6.1.4.1.1080:device-name:public:gated|1:grf_if1|1:grf_if2
|1:grf_if3|1:grf_if5|1:END
```

4. Specify SNMP trap destination

Specify the SNMP manager gateway or gateways where you would like the traps sent.

Either of these entries forward all **threshpoll** traps as specified.

To send traps to a Gateway named '`batman`', add the following entry to `/etc/snmpd.conf`:

```
MANAGER   batman
          SEND ALL TRAPS
```

To send traps to a Gateway at 192.168.0.22, add the following entry to `/etc/snmpd.conf`:

```
MANAGER   192.168.0.22
          SEND ALL TRAPS
```

Now send **snmpd** a HUP signal to re-read `/etc/snmpd.conf` and restart with the changes.

Starting/stopping threshpoll

Start and stop scripts are used to initiate and terminate configured **threshpoll** instances.

Starting threshpoll

Start **threshpoll** using the **start_threshpoll** program**.**

The script sets up the appropriate environment variables and begins the **threshpoll** program. The command here uses default values to start instance 1 at polling intervals of five minutes:

```
# start_threshpoll
```

When threshpoll starts, an identification number is displayed for the instance. This example shows the identification for instance 33:

```
# start_threshpoll 33 60 &
[1] 21998
```

Note: To have threshpoll run in the background, add an ampersand (&) to the end of the **start_threshpoll** command. This is shown in the example above.

The **start_threshpoll** command has two optional arguments, **instance** and **interval**.

The ***instance*** option is always first, and specifies the instance number to start. If no instance number is given, instance 1 is always started. If **interval** is specified, **instance** must also be specified.

```
# start_threshpoll 33
```

Start instance 33 to run at 2-minute intervals with this command:

```
# start_threshpoll 33 120
```

The ***interval*** option specifies the period of time (in seconds) that **threshpoll** waits before polling again. The default interval is five minutes.

Stopping threshpoll

Once started, threshold polling continues until a **stop_threshpoll** command is executed and terminates all necessary processes in the appropriate order.

When you stop **threshpoll**, you stop *all* instances that are running. You cannot stop only one or two specific instances. Here is an example of the **stop_threshpoll** execution:

```
# stop_threshpoll
kill 21998
kill -15 22016
[1] + Terminated                start_threshpoll 33
#
```

Pinglog monitoring utility

The **pinglog** program is used to ping GRF interfaces and attached devices to see if the interfaces are alive and responding. If an interface does not respond, a trap is sent to the configured gateway.

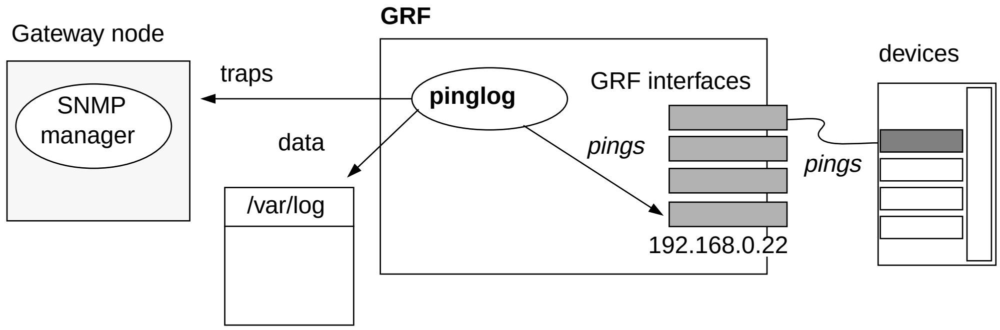


Figure 3-7. Basic components in pinglog operation

Configuring pinglog

There are two steps to configure **pinglog**. Once established, it runs automatically.

1 Specify the SNMP manager gateway or gateways where you would like the traps sent.

To send traps to a gateway named '`batman`', add the following entry to `/etc/snmpd.conf`:

```
MANAGER   batman
          SEND ALL TRAPS
```

To send traps to a gateway at 192.168.0.22, enter:

```
MANAGER   192.168.0.22
          SEND ALL TRAPS
```

Now send **snmpd** a HUP signal to re-read `snmpd.conf` and restart with the changes.

2 Set up the addresses of the interfaces and devices to be pinged regularly in the `/etc/mpingBackbone.1` file.

Here is how an entry is structured:

```
IP address:interface name:description of interface
```

– IP address of device/interface to be pinged.

– GRF interface name in the format `gx0yz`.

– Description is a string that will appear in the trap messages.

Use that structure to assemble an entry for ATM logical interface `gx0yz`:

```
/etc/mpingBackbone.1
192.168.0.22:gx0yz:ATM card to NAP
```

Use the `/etc/mpingBackbone.1.template` file as a model to create your copy of `/etc/mpingBackbone.1`.

Here is an unedited `/etc/mpingBackbone.1.template` file:

```
# /etc/mpingBackbone.1.template
############################################################
#
# Interfaces that will be monitored by pinglog
#
#  format:
#  IP address:interface name:description
#
# Example:
# 222.222.1.1:ga000:Some description of this interface
```

Logs

GRF logging is done in `/var/log`. Ping results are logged in `mpingLog.msglog.M.D.Y.h:m:s.i`

pinglog monitors this log file and sends a trap when it detects that one of the devices has gone down. Errors are sent to `NSM.pinglog.errlog.M.D.Y.h:m:s.i`.

(Month Day Year Hour Minute Second Instance-running)

```
$ more mpingLog.msglog.12.15.97.12:59:41.1
Dec 15 12:59:41 Reading 1 hosts...
Dec 15 12:59:41 ga010 went down (192.168.0.22) ATM card to NAP
Dec 15 12:59:41 1 of 1 hosts down
```

The error log file and message log file should be monitored by **grclean** to make sure it does not grow too big. Add the following lines to `/etc/grclean.logs.conf` in the "`of some interest`" section:

```
size = 25000
logfile=/var/log/NSM.pinglog.errlog.*
size=25000
logfile=/var/log/mpingLog.msglog.*
```

Starting and stopping pinglog

Start **pinglog** using the **start_pinglog** program. The script sets up the appropriate environment variables and begins the **pinglog** program. The **start_pinglog** command has two optional arguments, ***instance*** and ***interval***.

The ***instance*** option always comes first and specifies the instance number to start. If no instance number is given, instance 1 is always started.

```
# start_pinglog 120
```

The ***interval*** option specifies the period of time (in seconds) that **pinglog** waits before polling again. The default interval is 5 minutes. If ***interval*** is specified, ***number*** must also be specified.

Once started, pinging continues until a **stop_pinglog** command is executed and terminates all necessary processes in the appropriate order.

Configuration File Management System (CFMS)

The Configuration File Management System (CFMS) enables network administrators to monitor and manage the configuration files for multiple remote GRF routers. CFMS builds a CVS tree structure that contains local copies of the files and change histories when files are changed.

Using CFMS commands, administrators can review, edit, and track changes to those configuration files from their management workstation. CFMS runs on a UNIX server attached to client routers via a secure administrative LAN. CFMS is not loaded on a GRF.

Refer to Appendix C, *Configuration Tracking via CFMS*, for installation and usage information.

Here is a diagram of a typical CFMS installation:

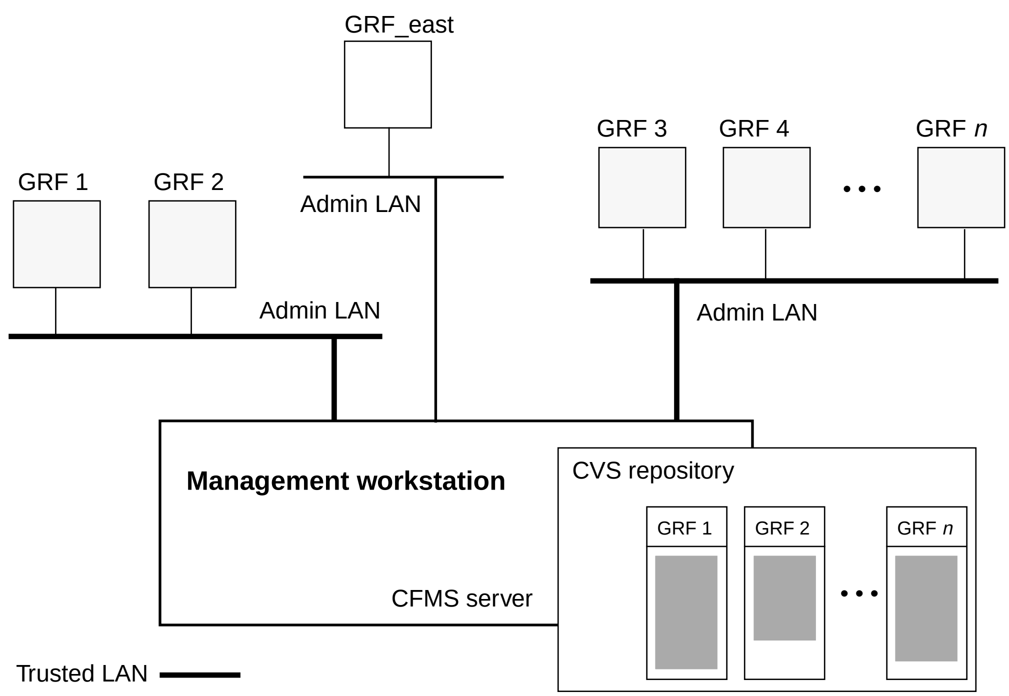


Figure 3-8. Diagram of CFMS server and client GRF routers

Figure 3-8 shows a CFMS server supporting GRFs on multiple administrative LANs. CFMS is intended to be used from a server on the administrative LAN connected to one or more routers. To protect against unauthorized access to routers on the network, CFMS must be run on a trusted LAN. In the example above, the connecting LAN to GRF_east is not secure and thus GRF_east cannot come under CFMS management.

Space availability on the server hosting the CFMS application determines how many routers can be managed. CFMS does not otherwise limit the number of client routers.

CFMS uses **expect** and **telnet** to move configuration files between the server and managed routers, and maintains a CVS repository on the CFMS server of configuration files for all routers under CFMS management.

Capabilities

CFMS performs the following functions:

- places routers under management
- removes routers from managed category
- collects the default set of configuration (`.conf`) files from the `/etc` directory of each managed router or a user-specified set of files (ascii) located anywhere in the router
- maintains the collected files in a CVS tree structure that enables revision and other file attributes to be tracked
- maintains a log of configuration file changes for archival and audit purposes
- enables the administrator to edit a copy of a specific configuration file, then places the new file version to the target router' own `/etc` directory

Note: CFMS cannot remotely activate the new configuration file on the target GRF. The administrator must log in to that router to activate the configuration change.

Security

Router passwords are stored on the server on which CFMS is run. CFMS does not encrypt the passwords it uses, and transmits passwords in clear text when it accesses managed routers. To protect against unauthorized access to routers on the network, CFMS must only be run on a trusted LAN.

Server disk space requirements

The base CFMS application requires 50Kb. Each router requires approximately 200Kb for its CVS repository space. The CVS repository is typically located in the same directory where CFMS is installed, but you can specify that it be located elsewhere.

Server software requirements

The CFMS server must be running or have resident the following software:

- **CVS** version 1.8.1 or later
- **perl** version 5.003 or later
- **ksh**
- **make**
- **expect** version 5.22.0 or later
- Solaris version 5.4 or greater, BSDI/OS 2.1 or greater
  This feature will work on these OS levels and may also work on BSDI/OS 3.0.

GRF requirements

The GRF router must have the following software installed:

- GRF software release 1.4.8 or later
- `/usr/bin/uudecode`
- `/usr/bin/uuencode`
- `/bin/tar`
- `/usr/contrib/bin/gzip`

The `user` and `root` accounts should operate properly with the default shell. These accounts should have **tar** and **uuencode**/**uudecode** in the command search path.

CFMS has no storage requirements on the GRF.

Logging

Logging is done to a file named `cfms-trace.log`. This file is created when CFMS starts and is located in the directory where CFMS is installed. It collects error and other messages that are generated from the operations of CFMS commands.

The log file can grow over time as CFMS is used. The system administrator of the CFMS server needs to see that it is periodically cleaned up.

Command set

CFMS has the following command set:

Command	Function
cfms add_host	Adds a GRF to CFMS management
cfms remove_host	Removes a GRF from CFMS management
cfms edit	Edits a configuration file that is under CFMS management.
cfms list	Lists the GRFs currently under CFMS management
cfms status	Lists the status of configuration files for GRFs under CFMS.

Sections that appear later in this chapter describe options for each command and provide examples of usage.

CFMS installation and usage information is in Appendix C, *Configuration Tracking via CFMS*

Management Tasks

4

Chapter 4 provides a procedure or outline for a variety of administrative and maintenance tasks.

Chapter 4 includes these tasks:

Note: A reference to GR-II or RMS node systems also includes GigaRouter systems.

Preparing to update software

This section includes a few things you should do before installing a new release.

1 Know which files are updated, which are replaced during the install:

- `xxx.conf.template` files in the `/etc` directory are updated if needed
- `xxx.conf` files in the `/etc` directory are not touched
- all other files are replaced

2 Check the current software version you are running:

```
# getver
Current Revision: 1.4.10   Version: default
```

Or, you can use the **cat** command:

```
# cat /etc/motd
```

3 Check internal flash device capacity:

```
# flashcmd df
Filesystem  K-blocks   Used    Avail  Capacity  Mounted on
/dev/wd0a    78751     11543   63270   15%       /flash
```

4 Back up your current configuration files:

```
# grwrite -v
```

Changes to /etc/services are overwritten

GRF systems
When you use the **grass** command to modify the `/etc/services` file on a GRF, those changes only get saved to flash memory when you do a **grwrite**.

GR-II systems
Your changes to `/etc/services` are saved and archived by the **grc** command. Subsequent software upgrades on GRF and GR-II systems will install the new release version of `/etc/services`, overwriting any changes you may have made. Be sure to record your changes as you will need to manually reinstate them when you upgrade.

GRF installation command

Software updates are available via ftp and are in the form of gzip (`.gz`) files. After the new software is downloaded, the **grfins** command looks for the specified release directory, extracts the files, and installs them.

The syntax of the **grfins** command to execute the installation script is:

```
# grfins --source=<source_location> --release=<release_name> --activate
```

where

<release_name> is the actual release name as in `1.4.12`

<source_location> is usually `/flash/tmp`

Doing an ftp and a software upgrade

To upgrade, download a new release to your site and then install the new software by running the **grfins** script.

Note: You must reboot after installing software. If you do not reboot after running **grfins**, the GRF will be in an unpredictable state. Remember that rebooting interrupts *all* GRF operations.

1 Log on and start the UNIX shell:

```
super> sh
#
```

2 Mount the internal flash device and create the directory to install from (`/flash/tmp`):

```
# mountf -w
# cd /flash
# mkdir tmp
# cd tmp
```

3 Do the `ftp`.

In the following display example, 1.4.12 software is downloaded:

```
$ ftp ftp.ascend.com
Connected to ftp.ascend.com.
220 ftp FTP server (Version wu-2.4(2) Sat May 2 14:34:24 PDT 1998) ready.
Name (ftp.ascend.com:holly): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-This archive contains tools and software upgrades for users of Ascend
230-equipment.  For more information email info@ascend.com.
230-
230-Ascend.COM logs all connections. If you don't like this, log off now.
230-
230-If you do have problems, please try using a dash (-) as the
230-first character of your password -- this will turn off the
230-continuation messages that may be confusing your ftp client.
230-
230 Guest login ok, access restrictions apply.
```

At the first ftp prompt, set the mode to binary:

```
ftp> bin
ftp> cd pub/Software-Releases/GRF/1.4/1.4.12
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.

A_1_4_12.TAR.gz
A_1_4_12.root.gz
RN1_4_12.pdf
addendum.pdf
gated.pdf
cfms.tar.gz
1.4.12.TAR.gz
```

```
1.4.12.root.gz

226 Transfer complete.
118 bytes received in 0.0018 seconds (63 Kbytes/s)
```

These are the files you need to get:

```
ftp> get 1.4.12.TAR.gz
ftp> get 1.4.12.root.gz
```

Change directory to install the remaining files in `/flash` so they are not removed when the **flashcmd** executes:

```
ftp> lcd ..
ftp> get gated.pdf
ftp> get cfms.tar.gz
ftp> get RN1_4_12.pdf
ftp> get addendum.pdf
ftp> quit
221 Goodbye.
```

This procedure has downloaded the 1.4.12 release files onto the internal flash device.

The GRF 400 and GRF 1600 require two installation binaries: `1.4.12.TAR.gz` and `1.4.12.root.gz` (RMS node systems require only the `TAR.gz` file).

4 Unmount the flash device

```
# cd /
# umountf
```

5 **grfins** decompresses the files, installs the new files, and replaces the configuration file templates as needed. The `--activate` option ensures the installation will occur during the next boot:

```
# grfins --source=/flash/tmp --release=1.4.12 --activate
```

6 Remove the release files from the internal flash device and reboot to load RAM:

```
# flashcmd -w rm -rf /flash/tmp
# reboot
```

When the system comes up, it should indicate it is running 1.4.12.

Note: Before upgrading, please contact Ascend support for information on appropriate releases for your hardware. In the US call: 800-272-3634, from elsewhere call: 510-814-2333, or send email to `support@ascend.com`.

Power off or reboot the system

If you need to power off or reboot the GRF, first save the system configuration files and then reboot the system.

Save configuration files

GRF 400 and GRF 1600

Use the **grwrite -v** command to save the `/etc` configuration directory from RAM to a flash device. This preserves the configuration files over a reboot. The -v verbose option shows which files have changed and are being saved.

```
# grwrite -v
```

To save an alternate configuration on the internal flash (destination, Primary) based upon the currently-running configuration on the internal flash device (source, Primary):

```
# grsnapshot -sP=revision,version -dP=revision,version
```

RMS node systems

Use the **grc** command to archive the default set of GRF configuration files to a specified directory on a diskette, enter:

`# grc save -F -d` *`directory_name`*

Use the **-a** option instead of **-F** to archive to a specified file.

Reboot using shutdown (root login)

To cleanly stop and reboot the system from `root` login, use the UNIX **shutdown** command. The **shutdown** command performs an orderly shutdown, saving memory and allowing any transfers to complete. When the reboot option is specified, the system is rebooted and all media cards are reset.

```
# shutdown -r now
```

Reboot using grms (non-privileged login)

When working at the VT-100 terminal (or RMS node), use the **grms** command to halt, reboot, or shut down the GRF from the UNIX prompt. **grms** has a man page and is also described in the *GRF Reference Guide*. **grms** performs the same function as **shutdown**, but does not require the user to be logged in as `root`. However, it can only be used from the VT-100 terminal.   To perform an orderly reboot of the system, enter:

`# grms -r   ##` performs an orderly reboot

`# grms -h   ## halts the system, like` **shutdown -h**

`# grms -s   ## performs an orderly shutdown`

Upgrading from 1.3 to 1.4

Upgrades from 1.3 software to a 1.4 release are supported by an internal transition script and are not complicated.

Save your configuration files.

Save your configuration files.

Save your configuration files.

Record changes you make to certain files

Record the changes you make to the following files as they are always overwritten by a software installation using **grfins** (or **grinstall** on an RMS node system):

– `/etc/services`

– `/etc/*.conf.template` files

– `/etc/grclean.logs.conf`

– `/grass` directory

– `/etc/rc.local` (is kept but is modified, record any changes)

– `/etc/rc`

Changes made with grinch commands are temporary

This is because the CLI program, **ncli**, writes changes made via the CLI to **grinch** variables, these settings are permanent. Changes made with **grinch** commands do not get written to **ncli**, therefore, these changes are temporary and do not survive system reboot.

Save /etc configuration directory

GRF systems

Use **grwrite** to save the /etc directory:

```
# grwrite -v
```

RMS node systems

Use the **grc** command to save a copy of the `/etc` configuration directory

To use **grc** to archive the default set of GR-II configuration files to a specified directory on a diskette, enter:

```
# grc save -F -d directory_name
```

Use the **-a** option instead of **-F** to archive to a specified file. Here is the default list of files **grc** archives:

```
etc/aitmd.conf            etc/gritd.conf          etc/namedb
etc/bridged.conf          etc/grlamap.conf        etc/netstart
etc/crontab               etc/group               etc/networks
etc/fstab                 etc/grppp.conf          etc/passwd
etc/filterd.conf          etc/grpvc.conf          etc/printcap
etc/gated.conf            etc/grroute.conf        etc/pwd.db
etc/gettytab              etc/grstart             etc/rc
etc/grarp.conf            etc/hostname.txt        etc/rc.local
etc/gratm.conf            etc/hosts               etc/resolv.conf
etc/grclean.conf          etc/hosts.equiv         etc/services
etc/grclean.logs.conf     etc/localtime           etc/snmpd.conf
etc/grfr.conf             etc/master.passwd       etc/spwd.db
etc/grifconfig.conf       etc/motd                etc/syslog.conf
etc/grinchd.conf          etc/named.boot
```

Any changes you make to the `/etc/services` file are overwritten when you install a new software release. Record these changes and add them back after the upgrade.

Update changes to grclean.logs.conf *(if needed)*

grclean is an internal program that compresses, archives, and manages dumps and log files, and saves them to a specified file name.

Software release 1.3.11 changed the contents of the `/etc/grclean.logs.conf` file.

If you are upgrading to 1.4 from a 1.3.9 or earlier software release, the previous `/etc/grclean.logs.conf` file is renamed to `/etc/grclean.logs.conf.old,` and a new copy of `/etc/grclean.logs.conf` is installed. You may see the message describing this transfer if you are watching the console as **grfins** operates

If you have never made any changes to `/etc/grclean.logs.conf`, the upgrade has no effect. However, if you did change `/etc/grclean.logs.conf` in the past, then you must now make those changes again as soon as possible after the 1.4 update procedure is finished.

Cut and paste just your changes into the 1.4 version of the file, take care not to overwrite the new sections in the file. **grclean** manages many more logs than in prior releases.

Testing a new binary

Before you upgrade to a new binary file such as a media card binary, first run it in a test situation. After you test the binary and determine that it fixes a problem or adds a desired feature, then use the **grsite** command to save that file as part of the currently-running release.

1 Copy the new binary file to the proper place in system RAM.

For example, to install a new ATM OC-12c binary, `atm-12.testrun`, in place of the current binary, download it via ftp and put in the administrative home directory:

```
# cd /usr/libexec/portcards
# cp /home/admin/atm-12.testrun /usr/libexec/portcards
```

The original `atm-12.run` binary remains in the directory and can be re-installed if necessary.

2 Test the binary to make sure it is running properly in your configuration.

3 When ready, **cd** to the directory where the file was put:

```
# cd /usr/libexec/portcards
```

4 Issue the **grsite** command to save the file.

You can specify **grsite** to save the change to the next version, it will run the next time you reboot:

```
# grsite filename --next
```

You can specify **grsite** to permanently save the change to the currently-running version, it will continue to run:

```
# grsite filename --perm
```

Files saved using **grsite** without the **--perm** option are not carried forward during an upgrade when the **grfins** command is used.

Testing a new configuration

Using the steps in this section, you can create and test a new or experimental configuration and return to the current configuration at a later time.

First, save the current configuration with **grwrite**.

Then do a **grsnapshot** to save the current configuration on internal flash as `source 1.4.12,default` (in this example) and save a copy of the current as `1.4.12,experimental` (in this example).

Use **setver** to have the system load and run from the experimental configuration at the next reboot, then reboot.

Here are the commands for this example:

```
# grwrite
# grsnapshot -sP=1.4.12,default -dP=1.4.12,newstuff_config
# setver 1.4.12,newstuff_config
# reboot
```

At the reboot you are running on the copy of your current configuration. Make your test changes to this copy as you need. When all the changes are in, save them with a **grwrite**:

```
# grwrite
```

You are now running under the new changes. When you want to go back to the standby configuration, `1.4.12,default`, use **setver** again and reboot:

```
# setver 1.4.12,default
# reboot
```

At this point, you are back to exactly where you were at the beginning of this process.

If you are confused about which system you are actually running, use **getver** to check:

```
# getver
```

If you are not sure which system will be loaded at the next boot, use:

```
# getver -n
```

Note:
If the `1.4.12,newstuff_config` software performs as you intend, there is no reason you cannot continue to run this software.

Backing up the system

This section provides a brief backup process. Refer to the *GRF Reference Guide* for more information about the commands used here.

GRF systems

First, save the current `/etc` configuration directory with **grwrite**. This will capture any unsaved configuration changes you have made:

```
super> grwrite -v
```

Then use **grsnapshot** to save all files in the currently-running software on the internal flash to a backup file on the external flash in slot B. You must know the slot number of the external device.

grsnapshot does save all files in a single release, it is not possible to do incremental backups at this time. The **grsnapshot** process takes 4–5 minutes.

A December 7th backup could be saved to `120797_backup`:

```
super> grsnapshot -sP=1.4.12,default -dB=1.4.12,120797_backup
```

Later, you can use **grsnapshot** to restore the operating system from the external storage:

```
super> grsnapshot -sB=1.4.12,120797_backup -dP=1.4.12,default
```

You can also archive files to an NFS mounted file system using this command and the compress the directory afterward:

```
# grsnapshot -sP=1.4.12,default -dd=/grf/backups/120797
```

RMS node systems

First, save the current `/etc` configuration directory with **grc**:

```
# grc save -a archive_file
```

Later, use **grc** to restore the archived files:

```
# grc restore -a archive_file -d directory
```

Duplicating configs among GRF systems

As shown in Figure 4-1, the GRF control board supports the use of external 85MB and 175MB flash devices in PCMCIA slot A. Slot B supports a modem disk or flash device. A 520MB spinning disk is supported for Slot A. A vendor list is on the next page.

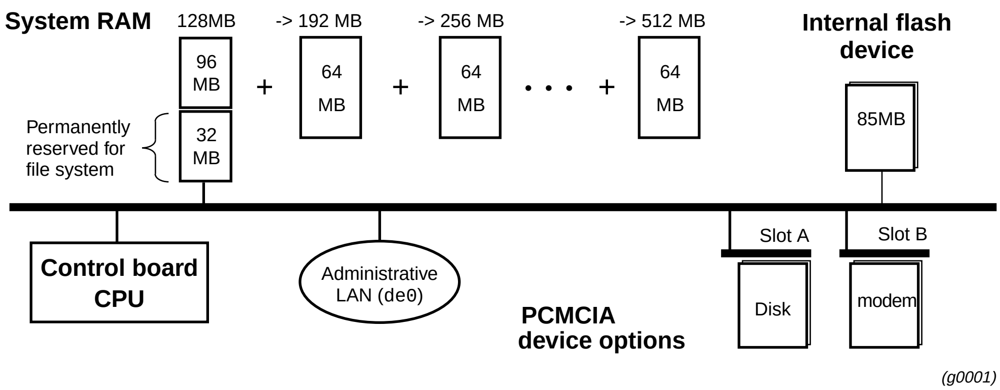


Figure 4-1. Support for external flash devices in PCMCIA slot A

This procedure demonstrates the process and commands you use to copy GRF A's release 1.4 configuration onto a flash device, and then load and install it on GRF B.

Replace italicized values with appropriate text. For example, `1_4,default` means a release 1.4 default system, `1_4,test` would mean a release 1.4 test system.

Note: The GRF cannot boot from an external device.

GRF A steps

1 Put the external flash device into GRF A, PCMCIA slot A is recommended.

2 Log onto the GRF as `root`.

3 Execute this command:

```
super> grsnapshot -sP=1_4,default -dA=1_4,default
```

4 Remove the flash device from GRF A.

GRF B steps

1 Now put the external flash device into GRF B, PCMCIA slot A is recommended.

2 Log onto the GRF as `root`.

3 Execute this series of commands:

```
super> grsnapshot -sA=1_4,default -dP=1_4,default
super> setver 1_4,default
super> mountf -w
super> sh
# cd /flash/etc.1_4,default
```

4 Edit `/etc/netstart` to change the host name and IP address to reflect the correct name and address

5 Edit `/etc/hosts` to change the hostname and IP address to reflect the correct name and address

6 Edit and correct any other files in which you may have references to the wrong `hostname`, these may include `/etc/grclean.logs.conf` and `/etc/syslog.conf`.

7 Execute this series of commands:

```
super> cd /
super> umountf
```

8 Reboot GRF B.

9 Check the new configuration.

Note: Do not try to access a PCMCIA slot when it is empty.

On remote machines, use the **csconfig** command to determine whether a PCMCIA slot is empty or full. Log in to the GRF as super user, and from the UNIX shell enter:

```
# csconfig -a
```

The status of each slot interface connection (up or down) and resident device is returned:

```
Slot 0: flags=0x5<UP,EMPTY>
Slot 1: flags=0x5<UP,FULL>
```

Slot A is reported as 0, slot B as 1.

The **csconfig *interface*** returns more information about a full slot:

```
# csconfig 1
Slot 1: flags=0x3<UP,RUNNING>
        Attached device: wdc1
        Manufacturer Name: "SunDisk"
        Product Name: "SDP"
        Additional Info1: "5/3 0.6"
        Function ID: 4 (PC card ATA)
        Assigned IRQ: 11
        Assigned I/O port1: 0x3d0-0x3df
```

Available PCMCIA devices

Ascend certifies the following ATA-compliant devices for GRF operation:

- Kingston Datapak 520MB, P/N CT520RM
- Sandisk 175MB Flash, P/N SDP3B
- Sandisk 85MB Flash, P/N SDP3B-85-101
- Aved 85MB Flash, P/N AVEF385MB25ATA501

Specifying an alternate Load configuration

This procedure describes how to switch the run time binaries for all media cards of a specific type by creating and running with a new Load profile. This is a system level change that affects all media cards of a single type. The last step describes how to restore the original binaries.

The example changes run time code for the Ethernet cards, here is a summary of the process:

- first, save your current Load profile configuration
- edit the current Load profile to specify new pathnames for Ethernet media and save the changed profile to one with a new name, `new_ether`.
- install the `new_ether` Load profile
- later, if necessary, restore the original Load profile

1 Begin by copying the current Load profile into memory (as `original.load`) so it can be reloaded if necessary:

```
super> save original.load load
```

2 Read and list the current Load profile into memory so it can be edited:

```
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A}{2 /us+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = { /usr/libexec/portcards/hssi_rx.run
                      /usr/libexec/portcards/hssi_tx.run +
dev1 = { /usr/libexec/portcards/dev1_rx.run
                      /usr/libexec/portcards/dev1_tx.run +
atm-oc3-v2 = { /usr/libexec/portcards/atmq_rx.run
                      /usr/libexec/portcards/atmq_t+
fddi-v2 = { /usr/libexec/portcards/fddiq-0.run
                      /usr/libexec/portcards/fddiq-1.r+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = { /usr/libexec/portcards/ether_rx.run
                      /usr/libexec/portcards/ethe+
sonet-v1 = { /usr/libexec/portcards/sonet_rx.run
                      /usr/libexec/portcards/sonet_t+
```

List the Ethernet load fields:

```
super> cd ethernet
type = ethernet-v1
rx-config = 0
rx-path = /usr/libexec/portcards/ether_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/ether_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

3 Specify the receive and transmit run-time code path names as `test1` and `test2`, respectively:

```
super> set rx-path=/usr/libexec/portcards/test1.run
```

```
super> set tx-path=/usr/libexec/portcards/test2.run
super> wr           (using an abbreviation for write)
LOAD written
```

4 Save the test pathnames in the new Load profile named `new_ether`:

```
super> save new_ether.load load
save to new_ether.load successful
```

5 Verify the Ethernet test pathnames in the new new_ether Load profile:

```
super> read new_ether.load
LOAD read
super> list ethernet
type = ethernet-v1
rx-config = 0
rx-path = /usr/libexec/portcards/test1.run
tx-config = 0
tx-path = /usr/libexec/portcards/test2.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

6 Now load the new `new_ether` Load profile:

```
super> load new_ether.load
 WARNING:  You are about to overwrite all or part of your current
 configuration.  If you wish to preserve your current configuration,
 please use the save command.
 Do you wish to continue without saving the current configuration?
[y/n] y
LOAD read
LOAD written
```

7 Verify the contents of the `new_ether` Load profile:

```
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A}{2 /us+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = { /usr/libexec/portcards/hssi_rx.run
                      /usr/libexec/portcards/hssi_tx.run +
dev1 = { /usr/libexec/portcards/dev1_rx.run
                      /usr/libexec/portcards/dev1_tx.run +
atm-oc3-v2 = { /usr/libexec/portcards/atmq_rx.run
                      /usr/libexec/portcards/atmq_t+
fddi-v2 = { /usr/libexec/portcards/fddiq-0.run
                      /usr/libexec/portcards/fddiq-1.r+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = { /usr/libexec/portcards/test1.run
                      /usr/libexec/portcards/test2+
sonet-v1 = { /usr/libexec/portcards/sonet_rx.run
                      /usr/libexec/portcards/sonet_t+
```

8 If necessary, restore your original Load profile:

```
super> load original.load
```

Swap in a media card load path

This procedure quickly replaces the binary load path on a single media card.

The change is made in the target media card's Card profile, in the `load` section. The steps given here are only the entries you make at the CLI `super>` prompt. Card profile display lines are omitted.

Verify the slot number of the target Ethernet media card:

```
super> grcard
0      HSSI_V1        running
1      ATM_OC12_V1    running
2      ATM_OC12_V1    running
3      ETHER_V1       running

super> read card 3
super> cd load
super> new hw-table ethernet-v1
super> cd hw-table
super> cd ethernet-v1
super> set rx-path=/usr/libexec/portcards/ether_rx.run
super> set tx-path=/usr/libexec/portcards/ether_tx.run
super> write
```

This is the procedure for removing the new media card binary load path:

```
super> read card 3
super> cd load
super> delete hw-table ethernet-v1
Delete hw-table/ethernet-v1? [y/n] y
hw-table/ethernet-v1 deleted
super> write
```

Re-running the config_netstart script

When you need to change the IP address or other configuration parameter, re-run the **config_netstart** script. The script runs automatically the first time a router is booted on site. The administrator first configures the IP address, host name, and administrative Ethernet LAN connection to get the router ready for the network.

If later you want to change any of these parameters, you must re-run the script. Running the script the first time is described in the *GRF 400/1600 Getting Started* manual. There is very little different when the script is re-run. Re-running the script eliminates individually editing several files, including `/etc/hosts`, `/etc/hostname.txt`, and `/etc/resolv.conf`

Reboot the router to make the configuration changes take effect.

config_netstart operates on all GRF and GR-II routers. You can also specify 10BaseT (TP), AUI, or BNC cable types for the GR-II. The GRF 400 and GRF 1600 control board connector autosenses the connection rate.

Script prompts

The script prompts for the following information:

- host name for GRF
- if you wish to configure the maintenance Ethernet interface
- IP address on GRF's maintenance Ethernet adapter
- netmask for GRF IP address
- if you want a default (static) route to another router on the maintenance Ethernet
- that router's IP address

You can see the current values displayed at each script prompt. At the end, you have the option to re-run the script to make more changes, to simply drop your responses without writing to any files, or to exit and have the answers written out as required:

```
You have now answered all the questions necessary for basic network
configuration.  If you didn't make any mistakes while entering your
answers, simply continue and the appropriate configuration files
will be created.

If you wish to exit this program without writing out the
configuration files, type <Control>-C.

Your current answers are:

Host Name: grf.tester.com
IP Address: 198.168.160.133
Ethernet interface: de0 (Digital DC21040/21140/21041)
Special Netmask:
Default Route: (none)

Do you wish to go through the questions again? [yes]
```

Enter `no` if you have no changes.

The script automatically saves your entries to the appropriate configuration file.

```
/etc/netstart configuration completed successfully.
These changes will take effect at the next reboot.
```

The `de0` reference is to the internal name for the control board's Ethernet interface through which the GRF connects to your site's maintenance/administrative LAN. You will see an entry for `de0` in the `/etc/grifconfig.conf` configuration file when you add media card interface and IP address information to that file.

The GR-II configuration script and file refer to the Ethernet interface as ef0.

Testing via ping

The basic tests for connectivity from a remote workstation or host include:

- Ping each media card interface from RMS.
- Ping another device attached to the router on a different subnet.
- Ping an interface external to the GRF (on some other router or station) via each interface. This can be to a workstation, an attached host, or another router.
- Test `telnet`, `ftp`, etc., through the router.

The UNIX **ping** command is modified to support GRF control board components. A ping does not usually disturb normal router operations. If a ping is properly returned, the media card interface is up, configured, and has an active media connection.

Caution: Do not perform a flood ping from the RMS to one of the GRF interfaces. This causes excess traffic on the combus that can degrade the reliability of the box.

ping media cards from the RMS

The RMS can locally ping all defined interfaces with nominal size ICMP packet (64 bytes). This use of ping only tests internal communication between the control board and a specified media card. It does not test message routing between media cards or communication between media cards and external devices.

The **ping -P grid <*slot number*>** command sends a message to a specified media card asking the card to respond back with another message. Log in as root. Specify the appropriate media card by its chassis slot number. For example, to act on the media card in slot 3, enter:

```
# ping -P grid 3
```

This is what you see when the media card responds:

```
68 bytes from 0:0x3:0:  time=0.293 ms
68 bytes from 0:0x3:0:  time=0.251 ms
68 bytes from 0:0x3:0:  time=0.288 ms
```

Use Control-C to stop the ping and view ping statistics:

```
-- 2 GRID ECHO Statistics --
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.969/1.060/1.172 ms
```

ping the control board

A router can send a ping of size 1000 bytes to the RMS on a directly-attached interface. To send a ping to the control board, enter:

```
# ping -P grid 66
```

Running a ping pattern

This command pings an external interface and generate some basic patterns, up to 16 bytes, repeated (count 20, pattern 7e, size 100 bytes, destination):

```
# ping -c 20 -p 7e -s 100 10.200.88.33
```

Adding a UNIX user (adduser)

This procedure describes how to add a UNIX user on the administrative Ethernet. (This is not the same as adding a user to the GRF CLI, Command Line Interface.) When you begin an **adduser** session, you are prompted for several pieces of information:

```
# adduser

Press to continue:
Login name: jon
Password: adminxx1
Retype new password: adminxx1
Primary group (? for list of choices)[admin_group]: admin_group
Full name: Jon Doe
Office: site_somewhere
Office Phone: 555-555-5555
Home Phone: 555-555-5555
Home directory [/usr/home/jon]:
Login shell (? for list of choices)[/bin/csh]: ksh
```

At the end of the questions, the user information is displayed.

```
Login name:     jon
Primary group:  admin_group
Full name:      Jon Doe
Office:         site_somewhere
Office phone:   555-555-5555
Home phone:     555-555-5555
Home directory: /usr/home/jon
Shell:          /bin/ksh
```

You are given a chance to change information. If there are no changes, press return and the prompts begin again:

```
Add this user to the password file?[yes]: yes
Cached passwd entry for new user: jon (uid: 102) the entry
      will be written when you are done adding users.
Add `jon’ to other secondary groups?[no]: no
adduser: can’t chdir to /usr/share/skel
#
#login (doesn’t accept new user at this point)
```

If you see that message, you will need to repeat the **adduser** process to add the new user.

Removing a UNIX user

Use the **rmuser** command to delete a UNIX user. In the shell

```
# rmuser
    Login name: cliff
    Delete user `cliff'? [yes]:  yes
    Deleting user: cliff
# cd /usr/home
# ls
  andy  brad  cliff   david  edward   frank
# rm -r /usr/home/cliff
# ls
    andy  brad   david  edward   frank
```

Adding a CLI user

To add a new user on the CLI interface, you create a new user profile.

CLI user profiles are not UNIX accounts. They only allow local router configuration access to users with a profile. A profile cannot be copied and then renamed because two profiles of a given type cannot have the same index (name). A new user profile is created in either of two ways.

- use the **read**, **set**, and **write** commands together
- use the **new** command

In this example, a new user profile for George is created using.

```
super> dir user
92  01/31/98 10:16:08  default
103  01/31/98 10:16:09  admin
106  01/31/98 10:16:10  super
99  02/03/98 15:27:00  frank

super> read user frank
USER/frank read

super> set name = george
super> write
USER/george written
super> dir user
 92  01/31/98 10:16:08  default
103  01/31/98 10:16:09  admin
106  01/31/98 10:16:10  super
 99  02/03/98 15:27:00  frank
100  09/03/98 16:00:00  george
```

Editing the new profile

Look at the new profile:

```
super> read user george
USER/george read
super> list
name* = george          * means a read-only field
password = ""
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace }}
active-enabled = yes
allow-system = no
allow-update = no
allow-password = no
allow-debug = no
prompt = ""
log-display-level = none
```

Set access user parameters, permissions, prompt text, password and so on.

```
super> set password = jo3jak
super> set prompt = geo
super> set allow-system = yes
super> set allow-update = yes
super> set allow-debug = yes
super> write
USER/george written

super> read user george
USER/george read
super> list
name* = george
password = jo3jak
auth-method = { PASSWD { "" 1645 udp "" } { 5500 udp 5510 tcp /var/ace }}
active-enabled = yes
allow-system = yes
allow-update = yes
allow-password = no
allow-debug = yes
prompt = geo>
log-display-level = none
```

Deleting a user profile

After one month at a prestigious ISP, George has gone back to the University to study the meaning of life. His CLI profile must be removed. Use the CLI **delete** command:

```
super> delete user george
Delete profile USER/george? [y/n] y
USER/george deleted

super> dir user
 92  01/31/98 10:16:08  default
103  01/31/98 10:16:09  admin
106  01/31/98 10:16:10  super
 99  02/03/98 15:27:00  frank
```

Obtaining system dumps

You can cause a dump to be saved by resetting the system with the **grreset -D** command. A nearly-full file system may prevent a dump from being taken. This situation causes a message to be sent to the `gr.console` log.

Execute the grreset -D command:

```
# grreset -D 3; grconslog -pvf
        Ports reset: 3
```

Here is the kind of information you see displayed as the system reboots:

```
Mar 12 16:52:27> [3] [RX] PORT 1:
Mar 12 16:52:27> [3] [RX] NEW0/50
Mar 12 16:52:27> [3] [TX] Static ARP entry mapped 222.222.222.1 to
    Port 1 IF 0x80 VPI/VCI 0/50
Mar 12 16:52:27> [3] [TX] Port 0 VC 0/250 does not exist for IF 0
    static ARP entry 222.4.4.101
Mar 12 16:52:30> [3] [TX] Inverse ARP mapped 222.4.4.12 to interface
    0x00 VPI/VCI 0/502
Mar 12 16:52:30> [3] [TX] Inverse ARP mapped 222.4.4.18 to interface
    0x00 VPI/VCI 0/501
Mar 12 16:52:30> [3] [TX] Inverse ARP mapped 222.4.4.14 to interface
    0x00 VPI/VCI 0/500
Mar 12 20:22:59> [3] Combus Xilinx Rev 4
Mar 12 20:22:59> [3] ATMQ Loader Rev 1.2.1
Mar 12 20:22:59> [3] Built on: Tue Feb 11 11:30:59 CST 1997
Mar 12 20:22:59> [3] BOOT ME.
Mar 12 20:22:59> [3] System dump in progress...
Mar 12 20:23:10> [3] ...dump complete, 8479 dump records sent.
Mar 12 20:23:10> [3] Loaded section 0 at 0x1000000 length 0x140
Mar 12 20:23:10> [3] Loaded section 1 at 0x1002000 length 0x3e6a0
Mar 12 20:23:10> [3] Loaded section 2 at 0x10406a0 length 0x2ba8
Mar 12 20:23:10> [3] Loaded section 3 at 0x1067000 length 0x468
Mar 12 20:23:10> [3] Loaded section 4 at 0x1067468 length 0x10
Mar 12 20:23:10> [3] Loaded section 5 at 0x1067478 length 0x24130
Mar 12 20:23:10> [3] Program load complete
Mar 12 20:23:13> [3] [RX] LOAD ME
Mar 12 20:23:13> [3] [RX] Loaded section 0 at 0x2000010 length 0xa8
```

Force a process to dump core

To force a process to dump core in the current directory, use the **kill -6 *pid*** command. For example, to force **snmpd** to dump core:

```
# cd /etc
# ps -aux | grep snmp
 root 10900 0.0 0.2 512 388 ?? S 6:00PM 0:00.07 snmpd /etc/snmp
# kill -6 10900
```

Use **ls** to look for the dump file:

```
# ls
```

Changing dump defaults

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes.

You can change dump settings for an individual media card in the Card profile. This procedure is included in each media configuration chapter in this manual.

Number of dumps saved

You can change the number of dumps saved per day in the `keep-count` field.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system has room for the default setting of `keep-count = 2` which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default. Default settings are shown in this example.

Here is the procedure to change the count to four:

```
super> read dump
DUMP read

super> list
hw-table = < { hippi 20 var 0 } { rmb 20 var 3} { hssi 20 var 7 }+
dump-vector-table = <{{ 3 rmb "RMB default dump vectors" < { 1 SRAM 2621+
config-spontaneous = off
keep-count = 2
super> set keep-count = 4
super> write
DUMP/ written
```

Dump events

The `hw-table` section has settings to specify when dumps are captured and where dumps are stored for each type of media card. Here is the path to examine the ATM OC-3c settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list atm-oc3-v2
media = atm-oc3-v2
config = 20
path = /var/portcards/grdump
vector-index = 5
```

The `config` field value controls when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The default setting is `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

To restrict dumps to during panics and hangs, change the field to 14:

```
super> set config = 14
super> write
DUMP/ written
```

Memory sectors dumped

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for each type of media card. These settings are read-only, and so not configurable by the user, but you can look at them.

Here is the path. While you are at the config level, use **cd** .. to go up to the main level:

```
super> cd ..
super> list dump-vector-table
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" < {1 "atm inst memory" 16+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" < { 1 "fddi/Q CPU0 core mem+
7 = {7 hssi "HSSI default dump vectors" < { 1 "hssi rx SRAM memory" 2097+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" < {1 "Ethernet rx SRA+
9 = {9 dev1 "DEV1 default dump vectors" < { 1 "dev1 rx SRAM memory" 2097+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" < {1 "ATM-12 SDRAM+
11 = {11 sonet-v1 "SONET default dump vectors" < {1 "SONET rx SRAM memor+
```

This sequence shows a portion of the areas in the ATM OC-3c card that are dumped. The fields are read-only and cannot be changed:

```
super> list 5
index = 5
hw-type = atm-oc3-v2
description = "ATM/Q default dump vectors"
segment-table = <{ 1 "atm inst memory" 16777216 4194304}{2 "SAR0-TX cont+

super> list seg
1 = { 1 "atm inst memory" 16777216 4194304 }
2 = { 2 "SAR0-TX control memory" 50462720 131072 }
3 = { 3 "SAR0-RX control memory" 50593792 131072 }
4 = { 4 "SAR1-TX control memory" 50724864 131072 }
5 = { 5 "SAR1-RX control memory" 50855936 131072 }
```

```
6 = { 6 "dual port memory" 33554432 32768 }
7 = { 7 "shared memory" 50331648 131072 }

super> list 1
index = 1
description = "atm inst memory"
start = 16777216
length = 4194304

super> list s 7
index = 7
description = "shared memory"
start = 50331648
length = 131072
```

Temporarily enable LMI debugging

To check whether LMI debugging is enabled on a card in slot 3 use the **grinch -p <*slot* #> 2.12.2.<*slot* # +1>.4.1.2.2.5** command:

```
# grinch -p 3 2.12.2.4.4.1.2.2.5
```

0 = off

1 = on

To temporarily enable LMI debugging, use the following **grinch** command:

```
# grinch -p <slot #> 2.12.2.<slot # +1>.4.1.2.2.5 = 1
```

The LMI output will be displayed in file `/var/log/gr.console`.

Remember to disable LMI debugging:

```
# grinch -p <slot #> 2.12.2.<slot # +1>.4.1.2.2.5 = 0

# grinch -p 3 2.12.2.4.4.1.2.2.5 = 0
```

Sending dumps – put command

You can ftp memory and log dumps to customer support.

Process overview

- Go to the location of the files you want to send, for example, Monday's dump in `/var/portcards`.
- Then ftp to the ftp server.
- Once on the server, you need to change to the `/incoming` directory and create a new directory with the name of your site.
- After a site directory is created, use the **put** command to send your collected files there.

ftp to the server

1 **cd** to the file directory and execute the ftp command:

```
# cd /var/portcards/monday_dump
# ftp ftp.ascend.com
```

The screen displays messages and a confirmation of connection similar to the following:

2 Do the `ftp`.

In the following display example, 1.4.12 software is downloaded:

```
$ ftp ftp.ascend.com
Connected to ftp.ascend.com.
220 ftp FTP server (Version wu-2.4(2) Sat May 2 14:34:24 PDT 1998) ready.
Name:
```

3 Type "anonymous" at the `Name:` prompt:

```
Name (ftp.ascend.com:holly): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
```

4 Enter your email address at the `Password:` prompt:

```
Password: someone@someplace.xxx
230-This archive contains tools and software upgrades for users of Ascend
230-equipment.  For more information email info@ascend.com.
230-
230-Ascend.COM logs all connections. If you don't like this, log off now.
230-
230-If you do have problems, please try using a dash (-) as the
230-first character of your password -- this will turn off the
230-continuation messages that may be confusing your ftp client.
230-
230 Guest login ok, access restrictions apply.
```

5 At the first ftp prompt, set the mode to binary:

```
ftp> bin
```

6 Go to the `incoming` directory and create a new directory to contain your site's files:

```
ftp> cd incoming
ftp> mkdir <site_name>
```

Make sure the name given the `<site_name>` directory clearly identifies your site or particular system:

```
ftp> cd <site_name>
```

7 Execute the **put** command for each file:

```
ftp> put <file1>
ftp> put <file..n>
```

8 Check that the files have transferred:

```
ftp> cd ..
ftp> ls incoming/<site_name>
```

9 Type "quit" to end the ftp session:

```
ftp> quit
#
```

Collecting system debug information

You can use the **grdinfo** tool to conveniently collect a large amount of system and RMS/control board data and statistics.

For the system, a single **grdinfo** command collects all `/etc/*.conf` files, the complete GateD configuration file (using **gdexpand**), and all Card, System, Dump, and Load profiles in `/etc/prof`, and compresses the data into one .gz file.

For the RMS and control board, a single **grdinfo** command collects the following and compresses it into one .gz file.

- software version, **getver** output (GRF), **sysctl** contents, kernel messages
- output from **ifconfig -a**, **netstat -in**, **netstat -rn**, **netstat -a**
- cardq information from media cards via **cardq -v**
- ARP information from media cards via **grarp -a**
- number of routes in each media card
- card counter output via **grstat**
- **mount** command output, external device data via **csconfig -a**
- output from **vmstat -sm**, process information via **ps -lam**
- **fstat** output
- mounted file system data via **df**

Other options collect and compress all system logs or dumps into a single file.

Refer to the "Management Commands and Tools" chapter in this manual for usage information.

Hot swapping media cards

GRF media cards are hot-swappable per media type. That is, you can swap out a HSSI card and replace it with another HSSI card. When the new HSSI card starts up and boots, it is identified to the system and is ready to be configured. Any IP addresses assigned to the HSSI card removed from slot 5, for example, are automatically assigned to the new HSSI card inserted into slot 5.

If you plan to change the type of media card that will replace the HSSI card, then you must reset the system to re-identify the new card.

After you insert the new type of media card but before you reset the GRF, output from the **grcard** command displays the actual media type but also indicates the previous media. This is the **grcard** output after a FDDI card has been inserted into the newly-vacated slot but before the GRF is reset:

```
# grcard
0   HSSI   running
1   HSSI   running
2   FDDI   held-reset (ERROR: must be HSSI)
3   HSSI   running
```

Resetting cards during traffic

When a significant amount of traffic is flowing from card A to card B and you reset card B, this does not cause a problem for card A. However, if you remove card B from the chassis, this can cause card A to hang or reboot.

Monitoring temperature and power

This section describes how system temperature and power supply output levels are monitored. Temperature is monitored directly by control board sensors and by the power supply module(s). Fan speed is also monitored.

Warning: The temperature extremes and failures described here are considered serious. The GRF can recover only with human intervention.

Warnung: Die oben beschriebenen Versagensfälle und Abschaltvorgänge stellen ernsthafte Situationen dar. Der GRF kann nur durch persönliches Eingreifen wieder betriebsbereit gemacht werden.

Temperature monitoring

When the sensor on the control board detects that a temperature level has reached the warning level, a signal is sent to the control component. On the GRF 400 the control board's amber FAULT LED comes on and flashes. On the GRF 1600, the TEMP LED comes on. The control component triggers an audible alarm and a warning message is sent, logged into `/var/log/messages,` and displayed on the screen. The alarm will continue to sound until the temperature drops below a problematic level.

If the temperature stays above the set level for longer than five minutes, the control board shuts down power to media cards. The control board continues to periodically print a message indicating the media cards are shut down and requesting a system reboot.

Keep the intake and outlet vents free of obstructions.

The network administrator can use the **temp** command to access temperature sensor data and determine the current board level temperature. **temp** is in the **grrmb** command set.

Power supply failure and shutdown

On the GRF 400 and GRF 1600, if a power supply fails, the PS1 or PS2 LED goes on, and a message is sent indicating a power supply failure.

One of the following messages are displayed:

```
“Power Supply PS1 Failure”
“Power Supply PS2 Failure”
```

The power supply module monitors its internal temperature, current, and voltage. Excessive heat and incorrect current and voltage readings can cause the power supply to shut down.

Fan monitoring – GRF 400

The GRF 400 is cooled by three fans that pull ambient air through one side of the chassis and push heated air out through the other side.

Each fan has a tachometer that measures fan speed in number of revolutions. The tachometer detects slowdowns in fan speed as well as failures. If a fan has failed and remains failed for 10 seconds, a message goes to the console and the fault LED lights. The fault LED remains lit unless the fan recovers and maintains speed for 10 seconds, the fans do not recover unless the GRF system is power cycled off/on.

Fan monitoring – GRF 1600

Two rotary fans cool the chassis, excluding the power supply compartment. The fans operate in tandem. At start-up, both fans operate at 100% of RPM capability. Gradually each fan slows down so that, in normal conditions, each fan operates at 50% speed. When the GRF is plugged in, you can hear the changes in fan speeds.

Tachometers on each fan unit ensure steady, sufficient airflow. When a tachometer detects that a fan is dropping below the 50% rate, it causes a signal to the other fan to speed up. When a problem occurs with either fan, the control board “FAN” LED lights.

The GRF 1600 fan tray is site replaceable, but must be returned to Ascend for service. The fan tray sits on a pair of guides and is held in place by two captured screws. While the fan tray is hot swappable, the amount of time the GRF can operate without a fan tray in place depends upon the number of installed media cards and the ambient air temperature. The temperature sensor on the control board shuts the GRF down if the operating temperature is exceeded.

Intervention

Intervention may require pushing the reset button on the control board or power cycling the chassis. If a temperature failure has been sensed, a problem exists in the GRF and you need to contact Customer Support.

ATM OC-3c Configuration Guide

5

Chapter 5 describes the implementation of ATM OC-3c on the GRF and other features supported on the ATM OC-3c media card. It includes the information needed to configure ATM OC-3c interfaces and parameters in CLI profiles as well as in the `/etc/grifconfig.conf` and `/etc/gratm.conf` files.

The ATM OC-3c media card provides two independent physical ATM interfaces, each of which supports 70 logical interfaces.

Release 1.4 supports only version 2 of the ATM OC-3c media card, ATM/Q. The first version of ATM OC-3c, sometimes referred to as "ATM classic," is supported only in 1.3 and earlier releases. In 1.4 manuals, ATM/Q and ATM OC-3c both refer to version 2.

Chapter 5 covers these topics:

ATM components

This section briefly describes components used in ATM configuration.

Virtual circuits and VCIs

Each virtual circuit is identified by a pair of numbers, representing a virtual path identifier (VPI) and a virtual circuit identifier (VCI). This pair is represented using a slash (/) to separate them (for example, 0/2645). The VPI/VCI must be unique on a link. Because it is acceptable to use the same VPI/VCI on different links, a GRF can have the same VPI/VCI active on each physical interface.

The ATM OC-3c media card supports up to 1024 virtual circuits (VCs).

Virtual circuit identifiers (VCIs) name virtual circuits. Virtual circuits 0-31 on each VPI are reserved for use by the ATM Forum for specific functions:

- VPI/VCI 0/5 is used for UNI signalling.
- VPI/VCI 0/16 is used for ILMI SNMP registration of NSAP addresses.

Virtual paths and VPIs

A virtual path consists of one or more virtual circuits. Virtual path identifiers (VPIs) 0 through 15 are available.

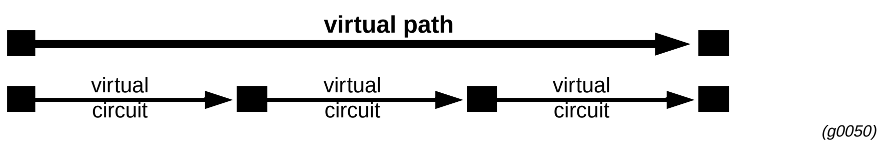


Figure 5-1. Components that form a virtual path

VPI/VCI

VPI/VCI specifies the Virtual Path Identifier and Virtual Circuit Identifier of the virtual circuit, separated by a slash (/), for example, 0/126. VPI/VCI is assigned in the `/etc/gratm.conf` file.

VPI 0	VCI 0 through VCI 511 can be used.
VPI 1. .15	VCI 0 through VCI 32767 can be used.

You cannot assign the same VPI/VCI to more than one circuit on the same physical interface.

Permanent virtual circuits

A permanent virtual circuit (PVC) is a logical connection across a physical path. Multiple PVCs share the same physical path. PVCs are configured statically and can be assigned a quality of service in terms of an amount of bandwidth. PVCs are configured in `/etc/gratm.conf`.

Switched virtual circuits

Switched virtual circuits (SVCs) are created/destroyed dynamically using standard signaling protocols. These protocols allow ATM devices to create/destroy connections in response to traffic demands. The VPI/VCI for a given SVC is determined at the time of connection setup, and thus requires no manual configuration in `/etc/gratm.conf`.

The host switch and the remote host of the ATM link must use the same version of the signaling protocol.

UNI signaling uses ATM addresses, not IP addresses. The ARP server maps an IP address to an NSAP address so that ATM signaling can create/use the appropriate SVCs for traffic destined for the given IP address. The ARP server's NSAP address must be configured in the Service section of `/etc/gratm.conf`. It is necessary to specify which of the UNI signaling standards (UNI3.0 or UNI3.1) you wish to use, and to assign an ARP server to each logical interface on which SVC operation is desired.

PVCs and SVCs can be used simultaneously on the same physical interface (port). PVCs and SVCs can also coexist on the same logical interface.

ILMI

Interim Local Management Interface (ILMI) is the ATM Forum specification for incorporating network-management capabilities into the ATM UNI. ILMI is a management information base (MIB) that provides status and communication information to ATM UNI devices. ILMI provides status information about virtual paths, connections and address registration. ILMI also determines the operational status of a logical port.

Traffic shaping

Traffic shaping is a specification of transmission parameters designed to ensure a specific quality of service (QoS) between endpoints in ATM virtual circuits.

Traffic shaping parameters can be specified for PVCs and SVCs (via logical interface settings), but only for output; the GRF does not control (police) incoming cell packets. Outbound traffic flow is determined by the rates set on the transmitting interface. Traffic shaping only affects cells *leaving* the ATM card.

The GRF receives and sends IP packets. When a received packet has an ATM destination, the packet is sent over the switch to the forwarding ATM media card. The ATM card segments the packet into cells and sends them out over the appropriate virtual circuit.

Parameters

Traffic shaping on the ATM card uses three parameters that effectively manage the timing of the transmission of ATM cells over SONET OC-3c media.

The parameters are set in `/etc/gratm.conf` and include:

– peak cell rate, in kilobits/second (PCR)

– sustained cell rate, in kilobits/second (SCR)

– maximum burst size, in cells (MBS)

Quality of Service (qos) is also set in `/etc/gratm.conf`:

– qos priority is either high or low

Note: Remember to specify PCR and SCR in kilobits/second, not in bits/second. For example, use 155520, not 155. If you specify 155, the ATM card moves data at 19 bytes per second, and appears to be non-functional.

Peak cell rate

Peak cell rate is the maximum rate at which cells will be output. Cells can be sent at rates lower than the specified peak, but never faster.

Peak rate is the most basic level of traffic shaping. The peak is set to match the highest rate at which the receiving endpoint is able to accept incoming cells. The maximum peak rate for ATM OC-3c is 155520 kilobits/second.

The GRF has a large buffer memory in which to buffer cells when they are arriving faster than the selected peak rate allows. If the mismatch in speeds is large, packets on the faster incoming network eventually will be lost, and retransmission will be required.

Sustained cell rate

The sustained cell rate (SCR) is generally the effective transfer rate. The sustained rate is the upper bound on the average cell rate (number of cells transmitted/duration of connection). If not specified, it defaults to the specified peak rate (PCR).

Software adjusts each specified sustained cell rate so that it is a simple fraction (1/2, 1/3, 1/4, ... , 1/63) of the associated peak cell rate, rounding up.

Maximum burst size

Maximum burst size is the specified number of cells allowed to burst at the peak rate for some short period of time. If not specified, it defaults to the peak rate.

Maximum burst size has no meaning unless the specified sustained rate (SCR) is less than the peak rate (PCR). As long as the VC has data to send, it sends its cells at the sustained rate. If the VC runs out of data, it can accumulate a certain number of “credits” for cells not sent. Then when a packet is queued for output, cells can be sent at the peak rate until the credits (one per cell) are used up. After that, transmission goes back to the sustained rate. Within a certain latitude determined by the MBS, this allows a VC to transmit at the sustained rate on the average, even though it cannot supply data at that steady rate. The MBS value is the maximum credit in cells that a VC can accrue.

When setting the MBS, consider the ability of the connecting ATM switch to buffer cells. The larger the buffer, the bigger you can set MBS. A 1500-byte IP datagram takes 32 cells. A 9180-byte datagram uses 192. If the switch can handle it, it is likely that setting MBS to at least one of these values means that an entire packet can be sent at the peak rate even while the VC maintains a lower average rate.

Burst rate credits

Burst rate credits come from unused sustained rate transmit credits. This means that the virtual circuit (VC) has to have been transmitting below the sustained rate in order for any burst rate credits to accumulate. For bursting to occur, the VC must average less than the sustained rate. Unused sustained rate transmit credits can accumulate due to recent idle and under-subscribed periods.

– In a recent idle period, the circuit usually transmits at the sustained rate but has been idle for the last N cell times.

– In an under-subscribed period, the circuit usually transmits below the sustained rate.

Burst credits are accumulated at the sustained rate but are transmitted at peak rate.

Here is an example in which:

– PCR = 155000000

– SCR = 77500000

– MBS = 2048

```
Time per cell  = ((53-byte cell x 8bits/byte)  / PCR)
               = (53x8) / 155000000
               = 2.7us per cell
Burst time     = MBS x time per cell
               = 2048 x 2.7us
               = ~5 ms
```

With these credits available, the VC could transmit at the peak rate for up to 5 milliseconds at the largest burst size. In this example, burst credits are 0.5% of total transmission time.

In summary, if a circuit is not able to send a cell when it is its “turn”, the circuit accumulates a credit. When there is an accumulation of such credit, the circuit can issue cells at the peak rate until the credit is used.

Rate queues and QoS

You define a rate queue in `/etc/gratm.conf` with a Traffic_Shape entry. In the entry, you must define maximum, sustained, and burst rates, and a QoS value. If no QoS is defined, the default is high

You can assign up to eight rate queues to interfaces on one logical interface, no more than four can be low, no more than four can be high. High priority queues have Quality of Service qos=high. Low priority queues have Quality of Service qos=low. High QoS queues get serviced before low.

You can create as many traffic shapes as you like, but there can be no more than eight rate queues, four with high QoS and four with low QoS. All traffic shapes with the same QoS and peak cell rate values refer to a single rate queue.

Multiple virtual circuits (VCs) and logical interfaces can be assigned to the same rate queue. If the available bandwidth is oversubscribed during high traffic times or because of multiple assignments, the available bandwidth is stochastically shared.

If the high-priority rate queues are over-subscribed and all the assigned virtual circuits are active, those assigned to low-priority queues may not get serviced.

For a given rate queue, all VCs assigned to that rate queue are serviced at the assigned rates. In turn, the rate queue is serviced at its assigned traffic shaping parameters (priority).

Each VC that has a packet ready to go transmits a packet. As an example, if five VCs all share a 10 Mbit rate queue, each VC is allocated 10 Mbits of bandwidth, the VCs do not share the 10 Mbit bandwidth.

If you specify a ninth peak rate, an error message reminds you of the limit. The error is generated when you try to set the fifth maximum bit rate in either the high or low QoS. If you have four high and zero low QoS, and try to implement a fifth high QoS, you will get the error. The limit is based on four high and four low QoS.

Priority

Priority is a characteristic of a rate queue.

The rate queues are divided into two groups. Four are high-priority, and four are low-priority. PVCs and logical interfaces assigned to rate high rate queues have absolute priority for transmission over those assigned to low-priority queues. In practice, all high-priority queues have the same high level of access, all low-priority queues have the same low level of access.

If high-priority and low-priority messages are both queued for output and are equally eligible to be sent as determined by traffic shaping, all high-priority queues are serviced before any low-priority queues.

Priority becomes an attribute of the logical interface. SVCs have the priority level of their assigned logical interface. A high-priority (for access) queue means a QoS = high. A low-priority (for access) queue equates to a QoS = low. QoS is specified in `/etc/gratm.conf` as part of the traffic shaping name.

Rate queue example

This example discusses what can happen when large numbers of VCs are assigned to low priority rate queues in order to reserve resources for a smaller number of VCs assigned to high priority queues.

Here is the starting assignment of rate queues as shown in the **maint 125** command:

```
GR 1> maint 125 0
[TX] RQ State   Rate(Kbs)  VPCIs
---------------------------------------------------------------------
[TX] 00 ENABLE   10000  0/44 0/45
[TX] 01 ENABLE   48000  1/44 1/45 1/46 1/47 1/48 1/49
[TX] 02 DISABLE
[TX] 03 DISABLE
[TX] 04 ENABLE    6000
[TX] 05 ENABLE    4800
[TX] 06 ENABLE   30000
[TX] 07 ENABLE   36000  7/40 7/41 7/42 7/43 7/44 7/45 7/46
[TX]                    7/47 7/48 7/49 7/50 7/51 7/52 7/53
[TX]                    7/54 7/55 7/56 7/57 7/58 7/59 7/60
```

The site reports experiencing slow response time on this ATM OC-3c card and on remote interfaces. The low QoS rate queues show high packet loss, the high QoS rate queues show minimal packet loss.

Here is an analysis of the problem. Based on the number of switch receive overflows, either there are multiple cards sending lots of traffic to the ATM card, or the input to rate queue 07 is greater than the output of that rate queue. The sum of the incoming packets destined for rate queue 07 is greater than the output bandwidth of the rate queue.

As packets process, the transmit buffers will queue upstream of the SAR chip. Nothing prevents a majority of the transmit buffers from being consumed by packets destined for VCs on a low priority, low bandwidth rate queue. Having a majority of the buffers tied up on low priority, low bandwidth rate queues robs high priority, high bandwidth traffic of buffers.

As a solution, the rate queue allocations are changed:

```
GR 1> maint 125 0
[TX] RQ State   Rate(Kbs)  VPCIs
---------------------------------------------------------------------
[TX] 00 ENABLE   10000  0/44 0/45
[TX] 01 ENABLE   48000  1/44 1/45 1/46 1/47 1/48 1/49
[TX] 02 ENABLE   36000  7/40 7/41 7/42 7/43 7/44 7/45 7/46
[TX]                    7/47 7/48 7/49 7/50 7/51 7/52 7/53
[TX]                    7/54 7/55 7/56 7/57 7/58 7/59 7/60
[TX] 03 DISABLE
```

```
[TX] 04 DISABLE
[TX] 05 DISABLE
[TX] 06 DISABLE
[TX] 07 DISABLE
```

This configuration avoids oversubscribing rate queues having low bandwidth, low priority traffic. The 21 VCs do not jam the transmit buffers behind a low priority rate queue. Although oversubscription also applies to high priority traffic, it is worse with low priority traffic because those packets must wait for all high priority packets to leave the buffers before being served.

Another technique is to use the sustained rate per VC to meter the output.

In `/etc/gratm.conf`, let every VC have a peak=155000:

```
Traffic_Shape name=Bulldozer peak=155000 sustain=155000 burst=2048 qos=high
Traffic_Shape name=T1 peak=155000 sustain=1544 burst=2048 qos=high
Traffic_Shape name=T3 peak=155000 sustain=45000 burst=2048 qos=high
Traffic_Shape name=10baseT peak=155000 sustain=10000 burst=2048 qos=high
```

Or, since high priority rate queues are handled in order (00, then 01, then 02, then 03), you could do something like this:

```
Traffic_Shape name=Bulldozer peak=155000 sustain=155000 burst=2048 qos=high
Traffic_Shape name=T1  peak=100000 sustain=1544 burst=1024 qos=high
Traffic_Shape name=T3  peak=100000 sustain=45000 burst=1024 qos=high
Traffic_Shape name=10baseT peak=100000 sustain=10000 burst=64 qos=high
```

In this way you assure the “`Bulldozer`” traffic always gets serviced before the T1, T3, or 10baseT traffic does. The SAR round robins among the high priority queues, giving you a priority scheme within the high priority queue class.

This configuration meters traffic based on sustained rate, not peak rate, and creates priority based on the servicing order of the rate queues. With all the peak rates set high, you minimize delay experienced by bursty, mostly idle circuits, and put all of them on the high priority queue to prevent the transmit buffers from filling with low priority packets.

Using this approach means you will only run into the "all buffers consumed by low priority, low bandwidth packets" condition when the input to the ATM card is greater than 155Mbps, and input is greater than output.

Queues are metered by the SAR based on the sustained rate. Having the peak = 155000 means that bursty, mostly idle sources will get served because they will transmit at the peak rate for burst size cell times. Average usage on continuously busy VCs will still average sustained rate because the SAR meters that on a per VC basis.

Setting output rates

Sending at a controlled rate

To ensure that the transmission of cells does not exceed a specific rate, you can create a traffic shape specifying that peak rate.

When the optional sustained rate and maximum burst size are not specified, the ATM card automatically sets sustained rate to equal the specified peak rate. The GRF card attempts to steadily issue cells at the peak rate, but no faster.

Should cells come in faster than the specified peak rate allows them to go out, the GRF's memory will buffer them as necessary. Buffering serves to smooth the speed mismatch that can occur if, for example, data from a HIPPI source is being sent to an ATM end point.

However, if the speed mismatch is large enough, packets on the faster network will eventually be lost and retransmission will be required.

Allowing an average or fluctuating rate

To ensure that a defined average rate of cell transmission is maintained over the duration of a connection, specify a sustained cell rate (SCR), a maximum burst size (MBS), and a peak cell rate (PCR) for the VC.

A *sustained* rate is the upper bound of an average or *sustainable* rate.

If SCR and MBS are specified, cells issue at the sustained rate. The sustained rate can be thought of as equivalent to assigning cell "slots" to the VC at a certain time interval. If the VC is not able to use its slot because no cell is ready to send, it accumulates a "credit". Whenever there is accumulated credit, cells can issue at the peak rate until the credit is exhausted, and then cells will again issue at the sustained rate.

Due to the time-slotted nature of ATM, the sustained cell rate must be no more than one-half of the peak rate to be effective. It is not possible, for instance, to operate with PCR = 130000 and SCR = 100000. Software will set SCR = PCR in this case.

Make SCR a simple fraction of PCR: 1/2, 1/3, 1/4, ... , 1/63. Software adjusts each SCR to make a simple fraction, rounding up as needed.

Specify maximum burst in units of 32 cells, in other words, in an amount evenly divided by 32. Software adjusts other amounts, rounding down as needed. The largest maximum burst size is 2048 cells.

ATM OC-3c on the GRF

This section describes the implementation of ATM OC-3c card features on the GRF router.

The GRF ATM OC-3c card supports two types of traffic, VC multiplexing and classical IP over ATM. The IP packet is carried directly over ATM.

In VC multiplexing, the Protocol Data Unit (PDU) inside the ATM cell is preceded by a LLC header. LLC is needed when several possible protocols are carried over the same VC. There is a SubNetwork Attachment Point (SNAP) header which follows the LLC header that specifies distinct routed or bridged PDUs.

Physical and logical interfaces

Figure 5-2 shows the organization of physical and logical ATM interfaces on the ATM OC-3c media card:

ATM OC-3c media card:

	Logical interfaces	VPI	VCI	Total # of active VCs
Physical interface 0 (top)	**0 – 7f (range)**	**0**	**0 – 32767**	**512**
		1 – 15	**0 – 511**	
Physical interface 1 (bottom)	**80 – ff (range)**	**0**	**0 – 32767**	**512**
		1 – 15	**0 – 511**	
	(70 logical interfaces per physical interface)			**(1024 per card)**

(G0048)

Figure 5-2. ATM physical and logical interfaces

The ATM OC-3c media card supports two physical interfaces, 0 (top) and 1 (bottom).

Logical interfaces provide a simple way of mapping many IP addresses onto a single ATM port. The logical interface serves as the connection between ATM and IP, and is assigned a unique IP address in the `/etc/grifconfig.conf` file. Logical interfaces are numbered between 0 and 127 (0-7f) on the top interface, interface 0. Logical interfaces are numbered between 128 and 255 (80-ff) on the bottom interface, interface 1.

Modes of operation

SDH and SONET

The ATM physical layer is set to either SONET or SDH mode, SONET is the default. Mode is configured per top or bottom physical interface (`connector=`), and not on a logical interface.

You specify mode in the Signalling section of the `/etc/gratm.conf` file. The example shows how mode is specified for the top interface as SDH and the bottom interface as SONET:

```
# Signaling parameters
Signaling card=5 connector=top protocol=UNI3.1  mode=SDH
Signaling card=5 connector=bottom protocol=UNI3.1  mode=SONET
```

Clock source

The ATM OC-3c SUNI component has a receive and a transmit clock. The receive clock is always at the SUNI's internal setting.

The transmit side clock setting can be toggled between the recovered receive clock (default) the SUNI's own internal clock, and the external oscillator. The clock setting is specified in `/etc/gratm.conf` in a *Signalling* line entry.

Transmit clock can be toggled temporarily using the ATM card's **maint 22** command. The setting reverts back to the recovered receive clock (internal) at ATM card reboot and system reset.

Loop timing configures the transmit port to the recovered receive clock, receive and transmit are synchronized.

AAL 5

The ATM OC-3c media card supports only AAL-5. The system ignores any other AAL settings.

The ATM Adaptation Layer (AAL) supports the different types of traffic that can cross over ATM. The AAL consists of the Convergence Sublayer and a Segmentation and Reassembly (SAR) layer. The Convergence Sublayer consists of two smaller parts, the Common Part CS (CPCS) and the Service Specific CS (SSCS). The SSCS is used to specify which type of encapsulation is inside an ATM cell.

Protocols supported

The ATM OC-3c media card supports the protocols listed here. Each protocol has an associated `proto=` field in the `/etc/gratm.conf` file. This field is used to assign a protocol to a PVC.

This protocol is switched:

- raw adaptation layer (AAL-5) packets,
  (`proto=raw`)

These protocols are routed:

- IP with LLC and SNAP encapsulation (ARP and IP),
  `(proto=ip)`
- null encapsulated IP, no ARP, also called VC multiplexed,
  needs ARP entry in `/etc/grarp.conf` (`proto=ipnllc or proto=vc`)
- LLC encapsulated ISIS, IS-IS only
  (`proto=isis`)
- LLC/SNAP encapsulated IP (IP, ARP, ISIS) and LLC encapsulated ISIS EXCEPT for RFC 1483 bridging
  (`proto=llc`)
- LLC encapsulated protocol including RFC 1483 bridging, needed for an interface using `bridge_method=llc_encapsulated`.
  (`proto=llc,bridging`)

- ATMP protocol to support home network connections using LLC encapsulation (`proto=llc_atmp`)
- VC multiplexed IP, an interface using `bridge_method=vc_multiplexed`. (`proto=vc`)
  An additional parameter (described in next section) specifies the protocol carried on the VC, for example, `proto=vc,bpdu`.
- VC multiplexed bridging, an interface using `bridge_method=vc_multiplexed`. (`proto=vc_mux,bridge`)
  An additional parameter (described in next section) specifies the protocol carried on the VC, for example, `proto=vc_mux,bridge,fddi_nofcs`.

For a bridging PVC with `bridge_method=vc_multiplexed`, one of these additional parameters specifies one of the following protocols to run on the P VC:

- Ethernet frames with Frame Check Sequence (`ether_fcs`)
- Ethernet frames without Frame Check Sequence (`ether_nofcs`)
- FDDI frames with Frame Check Sequence (`fddi_fcs`)
- FDDI frames without Frame Check Sequence (`fddi_nofcs`)
- 802.1D Bridging Protocol Data Units (`bpdu`)
- IP datagrams

Using the protocols

There are a number of ways to encapsulate a packet, specifically LLC and LLC/SNAP. Another way is to have no encapsulation, specifically NULL.

The LLC and LLC/SNAP methods can encapsulate many datagram types, not just IP. The GRF can determine if an encapsulated packet is a type it can process based on fields that indicate payload type in the LLC or LLC/SNAP header.

A NULL encapsulated circuit is assumed to carry only one kind of traffic (based on its configuration) because there is no encapsulation header to tell a router the packet type on a per packet basis. This type of traffic is also referred to as ATM VC Multiplexing, a single protocol per VC. Circuits that you wish to reserve for only NULL encapsulated IP are assigned `proto=vc`. Circuits that you wish to reserve for only LLC/SNAP encapsulated IP and ARP are assigned `proto=ip`.

The `proto=llc` type supports routed PDUs. When you specify `proto=llc`, the ATM card handles all the LLC or LLC/SNAP types it can —ISIS, IP, and ARP). Hence, you can specify IS-IS and IS-IS with IP as `proto=llc`. (This is referred to a wide-open LLC, anything which can be routed, is routed.)

On an LLC/SNAP encapsulated circuit, the GRF can determine payload type on a per packet basis from the encapsulation header. It can be useful to restrict which encapsulated protocols the GRF actually processes. In the `/etc/gratm.conf` PVC statement, `proto=ip` refers to LLC/SNAP encapsulated IP and ARP. In this case, all non-IP and non-ARP packets are discarded. These packets are reflected in the IP discard column of **maint 14**, the count of unknown LLC.

Using `proto=raw`, you can switch two ATM PVCs from one interface to another. The ATM circuit acts as an ATM AAL 5 switch, not an ATM cell switch, since it reassembles everything before "switching" the packets. The mapping is port-VPI-VCI -> port-VPI-VCI, operating like a switch to extract the port from the destination interface field. This is non-routed, transparent transport of successfully reassembled AAL 5 PDUs from input to output, not a switch of ATM cells.

UNI signaling and SVCs

Each physical interface (port) supports UNI 3.0 and UNI 3.1. UNI signaling, and an option to set signaling off.

UNI signaling enables an ATM device to dynamically establish a connection to another ATM device without human intervention. This connection is called a switched virtual circuit (SVC), and is created entirely in software – no manual configuration is performed. The signaling protocol provides a mechanism through which switches, routers, and end stations obtain information needed to establish a connection. Signaling requires connection to an ATM switch.

MTU

The maximum transmit unit for an IP ATM OC-3c packet is 9180 bytes, it cannot be set to a higher value, but it can be set to a lower value. MTU settings are done per interface in the `/etc/grifconfig.conf` file.

Large route table support

ATM OC-3c card software maintains route tables containing up to 150K entries, and hardware support for full table lookups. Use the first command to find out the number of table entries, use the second to display the system route table:

```
# netstat -rn | wc -l
# netstat -rn
```

On-the-fly configuration of PVCs

On ATM OC-3c cards you can configure PVCs in the `/etc/gratm.conf` file without rebooting the card. This does not apply to reconfiguring SVCs, UNI signaling, or ARP servers. The process uses the **gratm** command and is described in the “ATM on-the-fly PVCs” section of this chapter.

Packet buffering

Buffering on the ATM media cards is done in terms of packets, one packet per buffer. Buffering is provided for 256 packets on the receive side and 256 packets on the transmit side. Each packet can contain up to 9180 bytes, the default IP MTU for ATM.

A full packet contains 192 cells (192 is obtained by dividing 9180 by 48 bytes, the length of a cell’s data payload). The transmit and receive sides can each output 49152 cells (256 buffers x 192 cells).

The **maint 10** (receive side) and **maint 110** (transmit side) commands display usage on receive and transmit side memories and buffers. They report free, fragmented, and available units. Refer to the “Getting ATM information” section at the end of this chapter.

ICMP throttling

The Internet Control Message Protocol (ICMP) is a message control and error-reporting protocol between a host and a gateway to the Internet. ICMP uses IP datagrams, and the messages are processed by the TCP/IP software. ICMP throttling is a way of limiting the number of messages generated per GRF card.

You can specify how many of several types of ICMP messages can be generated by the ATM OC-3c media card per one-tenth second. These are the message types:

– number of replies to echo requests
– number of “cannot deliver packet” replies (unreachable)
– redirect messages, number is not limited
– number of time-to-live replies
– number of parameter problem (packet discard) messages
– number of time of day time stamp replies to send

Specify ICMP throttling parameters in the Card profile.

Encapsulated bridging

The GRF implements RFC-1483 encapsulated bridging over PVCs on GRF ATM OC-3c interfaces using either VC-based multiplexing or LLC encapsulation.

A GRF functioning as a bridge is able to interoperate with other bridges to forward frames from one bridge to the other over ATM. This allows two independent bridged LANs at remote locations to function as one logical network transparently connected by ATM.

A PVC must be configured on the ATM OC-3c logical interface to support this function. Refer to the *Transparent Bridging* chapter in this manual for more information.

ATMP

The ATM OC-3c supports the Ascend Tunnel Management Protocol (ATMP). ATMP is a layer 3 UDP/IP-based protocol that provides a cross-WAN (Internet or other) tunnel mechanism using standard Generic Routing Encapsulation between two Ascend units. ATMP is described in RFC 2107.

The ATMP tunnel protocol creates and tears down the tunnel between a foreign agent and a GRF home agent. The GRF connection to a home network is made across a PVC from an ATM OC-3c card. The home network router connects to the GRF ATM PVC through an ATM VC. The ATM circuits are created and assigned ATMP parameters in `/etc/gratm.conf`.

Please refer to the “ATMP Configuration Guide” chapter for information about ATMP functions and configuration.

Laser shut off option

The laser component on an ATM OC-3c single mode media card can be controlled by the **maint 80** ***interface*** **0|1** (off | on) command. This example sets the laser off (0) in the bottom interface (1) of the ATM OC-3c card in slot 3:

```
# grrmb
GR 66> port 3
Current port card is 03.
GR 03> maint 80 1 0
```

ARP service for SVCs

ARP maps an IP address to an ATM VC. If no destination address is available, a request goes to the ARP server for IP to NSAP address mapping. For an SVC, the request is going to the ARP server through the time the circuit is being established.

Packets may be queued while awaiting ARP resolution. PVC packets only await resolution, then are dequeued and transmitted. On SVCs the GRF initiates, packets wait for resolution and for the circuit to be set up. Packets are dequeued once the SVC setup is complete.

Inverse ARP

The ATM OC-3c media card supports ATM inverse ARP.

The GRF supports Inverse ATM ARP for determining the IP address of the other end of the VPI/VCI. If the connecting device does not support Inverse ATM ARP, an ARP entry for the IP and VPI/VCI of the other device must be made in `/etc/grarp.conf`.

The GRF takes the ARP entry learned via ATM inverse ARP as opposed to the one in the `/etc/grarp.conf` file. If no ARP entry exists for a given PVC when **grarp** is run, the ARP entry given in the `/etc/grarp.conf` file is accepted.

When a GRF ATM interface receives an ARP entry via ATM inverse ARP for a PVC and the **gratm** process also tries to add an ARP entry for the same PVC, then **gratm** may exit with a message similar to this:

```
Jun 17 15:32:49 GigaRouter grinchd[120]:
/usr/sbin/grarp -i ga0yz -f /etc/grarp.conf exited status 1
```

ATM statistics and configuration data

The ATM OC-3c card has transmit and receive side processors, CPU0 and CPU1. **maint** commands are provided for each CPU, these commands are described at the end of this chapter. Other tools useful for looking at the ATM OC-3c media card include:

– **netstat -in**

– **ifconfig -a**

– **grstat**, displays layer 2 and 3 statistics

Examples of these tools are in the “Management Commands and Tools” chapter in this manual.

Looking at the ATM card

The ATM OC-3c media card provides two full-duplex interfaces. ATM OC-3c cards are available in single and multimode versions. Figure 5-3 shows a single mode ATM OC-3c faceplate. Single and multimode faceplates are the same except that each single mode interface has a “LASER ON” LED.

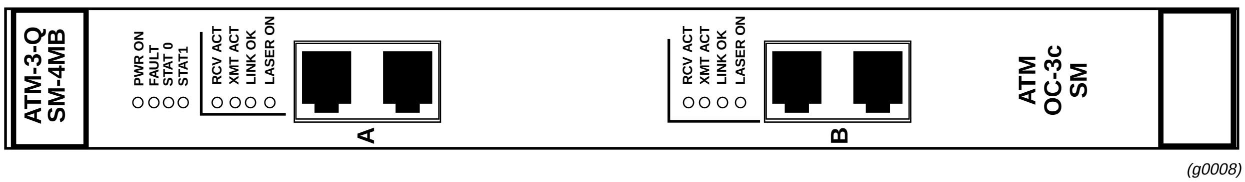


Figure 5-3. Faceplate of the ATM OC-3c single mode media card

LEDs on the faceplate

The top four LEDs on the faceplate indicate card status. The duplex interfaces A and B each have a set of LEDs. Refer to Table 5-1 for a description of each LED.

Table 5-1. ATM OC-3c LEDs

LED	Description
Power	This green LED is on when GRF power is on.
Fault	This amber LED turns on and remains on if an error condition is detected.
STAT 0 STAT 1	These green LEDs blink during self-test. When self-test completes, STAT 0 blinks ten times a second and STAT 1 blinks once a second. STAT 0 and STAT 1 indicate the activity of normal system interrupts. If the media card hangs, they either turn off and remain off, or they turn on and remain on.
RCV ACT	This amber LED blinks as ATM cells are received at the interface.
XMIT ACT	This amber LED blinks as ATM cells are transmitted out of the interface.
LINK OK	This green LED goes on when an optic cable is plugged into an interface and remains on while connection is good at both cable ends.
LASER ON	This green LED provides a safety warning on single mode ATM cards. One should not look into a laser-active interface component if a cable is not plugged in.

Ping times

You may notice some local pings to an ATM card can take a long time while other pings to that card are much faster. The following short discussion attempts to explain the differences in ping times.

Ping times are affected by:

- amount of traffic going through the router generally
- low or high priority of the assigned rate queue
- traffic on VCs assigned to low priority rate queues in relation to the traffic on VCs assigned to high rate queues

Answering local pings from the RMS is a low priority task for any media card. The more packets there are passing through the router, the longer a local ping may take since packet processing has priority over local ping processing.

Another factor is the priority of the assigned rate queue. Any packet on a high priority rate queue superceedes ALL traffic on low priority rate queues. All QoS = high packets are transmitted before any QoS = low packets are transmitted. Therefore, pinging a low priority rate queue in the presence of high priority traffic should have high delay. The ping packets are the least likely to be processed.

Also, if many more VCs are assigned to the low priority queues than are assigned to the high priority queues, and you ping a VC on rate queue 07, that one low priority packet has to wait for all high priority traffic to be processed.

List of ATM configuration steps

These are the steps to configure ATM cards:

1 Assign IP address to each logical interface

Edit `/etc/grifconfig.conf` to assign an IP address to each logical ATM interface.

2 Configure PVCs and SVCs in /etc/gratm.conf

Configure PVCs and SVCs in the `/etc/gratm.conf` file.

Step 3 includes options a site may wish to configure, none of them are required:

3 Specify ATM card parameters in the Card profile

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

These next steps are optional, they describe tasks that are performed infrequently:

4 Change Load profile (optional)

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every ATM card.

If you want to change the run-time code in one ATM card , make the change in the Card profile, in the `load` field.

5 Change Dump profile (optional)

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.

If you want to change dump settings for one ATM card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1. In the command-line interface, use **set** and **write** commands to save a profile. The profiles are stored in the `/etc` directory.

2. To save files in the `/etc` configuration directory, use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configuring an ATM interface

This section describes how to configure an ATM interface in the `/etc/grifconfig.conf` file. Defining the logical interface is the first step to configure an ATM virtual circuit. Use a UNIX editor to make entries in `/etc/grifconfig.conf`.

Each logical ATM interface is identified in `/etc/grifconfig.conf` as to its:

– interface name, `ga0yz` (always lower case)
– interface address
– netmask (optional)
– broadcast/destination address (optional)
– arguments field (optional)

The format for an entry in the `grifconfig.conf` file is:

```
name   address    netmask       broad_dest    arguments
```

Interface name *ga0yz*
Each logical GRF interface is given an interface name `ga0yz` where:

– the “ga” prefix indicates an ATM interface
– the chassis number is always “0”
– “y” is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
– “z” is the logical interface number in hex

Logical interfaces on connector 0 (top) range from `0` to `7f`. On connector 1 (bottom), they range from `80` to `ff`.

Address
Enter the IP or ISO address to be assigned to this interface.

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured.

Broadcast address
Use the broadcast address when you wish to specify other than all 1s as the broadcast address.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO addess is being added to an interface. Specify the MTU value as `mtu` *xyz*. Leave the `arguments` field blank if you are not using it.

Examples

The first entry assigns an IP address for logical interface 0 (upper physical interface) on the ATM card in slot 2, and specifies an MTU value lower than default. A dash is used as a placeholder for the broadcast address:

```
#/etc/grifconfig.conf
#name   address    netmask       broad_dest    arguments
#
ga030 10.20.2.234 255.255.255.0 10.20.2.235 mtu 9100
ga027f 10.20.2.238 255.255.255.0 10.20.2.239
```

The second entry sets an IP address for logical interface 130 (lower physical interface) on the ATM card in slot 13, and specifies a `zz.zz.zz.zz` destination address.

Save the /etc file

After you use the editor to save and close an `/etc` configuration file, write the file to the `/etc` configuration directory. Use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

Check port-level IP configuration

The set of **maint 3** commands display IP, VC, broadcast group, NSAP, and ARP server information for each port on the ATM card. The display example here shows the IP addresses configured on port 0:

```
maint 3 1 0
GR 2>
[RX] Port 0: LINK UP
[RX] Port 1: LINK DOWN
[RX]
[RX] IF       IP         STATE  |  IF        IP        STATE
------------------------------------------------------------------------
[RX] 00  10.20.2.234       UP      7f  10.20.2.238       UP
```

Enter **maint 3** to see the list of command options:

```
GR 2> maint 3
GR 2> [RX]
[RX] maint 3 1 0 -- IP config per IF port 0
[RX] maint 3 1 1 -- IP config per IF port 1
[RX] maint 3 2 0 -- VC config per IF port 0
[RX] maint 3 2 1 -- VC config per IF port 1
[RX] maint 3 3 0 -- BROADCAST GROUP config per IF port 0
[RX] maint 3 3 1 -- BROADCAST GROUP config per IF port 1
[RX] maint 3 4 0 -- NSAP config per IF port 0
[RX] maint 3 4 1 -- NSAP config per IF port 1
[RX] maint 3 5 0 -- ARP SERVER config per IF port 0
[RX] maint 3 5 1 -- ARP SERVER config per IF port 1
```

Check system-level IP configuration

The UNIX **ifconfig** ***interface*** command returns system level information for the specified interface name, here is the interface for logical interface 0 (`ga020`):

```
# ifconfig ga020
ga020: gritatm flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 9180
        inet 10.20.2.234 netmask 0xfffffffc broadcast 10.20.2.235

# ifconfig ga027f
ga027f: gritatm flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 9180
        inet 10.20.2.238 netmask 0xfffffffc broadcast 10.20.2.239
```

Check contents of grifconfig.conf file

The **netstat -in** command returns the contents of the `/etc/grifconfig.conf` file. Please refer to the **netstat** man page for information about other **netstat** options and explanantions of the type of information presented.

Here is the output from a **netstat** command looking at the ATM interfaces:

```
# netstat -in | grep ga
Name    Mtu   Network     Address          Ipkts Ierrs  Opkts Oerrs Coll
ga000   9180  <link14>    00:c0:80:fb:0f:00    437     0     14     0    0
ga000   9180  205.1.10    205.1.10.156         437     0     14     0    0
ga010   9180  <link15>    00:c0:80:f8:33:00      0     0      0     0    0
ga010   9180  208.1.11    208.1.11.156           0     0      0     0    0
ga0180  9180  <link16>    00:c0:80:f8:34:80     13     0     13     0    0
ga0180  9180  205.1.11    205.1.11.156          13     0     13     0    0
ga020   9180  <link37>    00:c0:80:f7:b2:00     12     0     12     0    0
ga020   9180  205.1.12    205.1.12.156          12     0     12     0    0
ga0380  9180  <link38>    00:c0:80:f7:72:8      14     0     14     0    0
ga0380  9180  205.1.13    205.1.13.156          14     0     14     0    0
ga040   9180  <link13>    00:00:00:00:00:00     16     0    189     0    0
ga040   9180  208.1.10    208.1.10.156          16     0    189     0    0
ga090   9180  <link17>    00:c0:80:fa:54:00      4     0      4     0    0
ga090   9180  204.101.11  204.101.11.156         4     0      4     0    0
```

Using the gratm.conf file

This section describes the `/etc/gratm.conf` configuration file. All ATM circuits and circuit parameters are configured in `gratm.conf`.

The file has five sections: Service, Traffic Shaping, Signalling, Interfaces, and PVC.

Service section

In the Service section you define the available ARP services. Give a different name to each type of ARP service you define. These names are assigned to the interfaces defined in the Interface section and specify the ARP service a logical interface will use.

Traffic shaping section

In the Traffic Shaping section you define the available traffic shapes. Give a different name to each type of traffic shape you define. These names are assigned to the interfaces defined in the Interface and PVC sections, and specify the traffic resources allotted to a logical interface.

Signalling section

In the Signalling section you assign a signalling protocol to each physical interface, top and bottom. When Switched Virtual Circuits are created on an interface, they will automatically use the protocol you have assigned. Optionally, you can also change the default mode and transmit clock settings in a Signalling statement.

Interfaces section

In the Interfaces section you identify the logical interfaces configured on this ATM card. Optionally, you can assign an ARP service name, a Traffic shaping name, and a bridging method to each logical interface.

PVC section

In the PVC section you assign three required parameters to each Permanent Virtual Circuit: the interface name (`ga0yz`), a VPI/VCI, and a protocol. Optional parameters include: an input AAL, a traffic shape name, or a destination interface

When editing `/etc/gratm.conf`, remember:

- Statements in `gratm.conf` may be longer than a single line. A statement can span multiple lines by ending each incomplete line of the statement with a back slash (\ ) character.
- Comments follow the Bourne Shell style. All characters following a # on a line are ignored.
- Names for ARP services and traffic shapes must be defined before they are assigned in the Interface and PVC sections.

A copy of the template for `/etc/gratm.conf` is in the *GRF Reference Guide*.

Configuring a PVC

This example configures a PVC with the following attributes:

- connects to a destination that does not support inverse ARP
- requires high priority quality of service
- is resident on upper physical interface, card in slot 3
- runs in SDH mode
- must be set to destination clock
- is on logical interface 153 (hex=99)
- has a VPI/VCI of 0/32
- runs IP protocol, AAL-5 (default, no matter what is set)
- IP address is 192.0.130.1
- the remote IP address is 192.0.130.111

In configuring a PVC, the IP address of the local ATM interface should be on the same subnet as the remote IP address.

Entries in /etc/gratm.conf

- Service section

  In this section you define the ARP server and a broadcast group (sets up OSPF multicast).

- Traffic Shaping section

  Set traffic shaping name and quality of service parameters.

  ```
  Traffic_Shape name=fast_high peak=155000 sustain=100000 qos=high
  ```

- Signaling section

  Set **protocol=NONE**, PVCs do not require signalling.

  ```
  Signalling card=3 connector=top protocol=NONE mode=SDH clock=Ext
  ```

- Interface section

  Specify interface name and traffic shape name for the logical interface.

  ```
  Interface ga0399 traffic_shape=fast_high
  ```

- PVC section

  Specify these characteristics for this PVC:

  - assigned logical interface name
  - VPI/VCI
  - protocol supported
  - assigned traffic shaping name

  ```
  PVC ga0399 0/32 proto=ip traffic_shape=fast_high
  ```

Entry in /etc/grifconfig.conf

Assign the IP address to the interface name, a netmask is required.

```
#name  address     netmask       broad_dest   arguments
ga0399 192.0.130.1 255.255.255.0
```

If the PVC will run IS-IS, here is the ISO address entry, iso is in the argument field:

```
# <interface-name> <iso-address> <iso-area> - iso
ga030 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

Entries in /etc/grarp.conf

An entry is needed when the destination does not support inverse ARP.

```
#[ifname] hostname phys_addr   [temp]   [pub]   [trail]
#
ga0399 192.0.134.111 0/32
```

Saving the files

After you use the editor to save and close an `/etc` configuration file, write the file to the `/etc` configuration directory. Use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

Verifying the PVC configuration

This section describes commands to review and verify PVC configuration parameters.

Check gratm.conf file entries

This **gratm** command parses the `/etc/gratm.conf` file on the specified media card without performing any configuration actions. It reports errors and file omissions. This report has no errors:

```
# gratm -n ga02
gratm: Accepted traffic shape hshq qos=high for top connector card 2.
gratm: Begin on-the-fly PVC configuration for card 0x2
/usr/nbin/grinch -p 2 2.12.2.3.17.3.1=1
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.1=-1
/usr/nbin/grinch -p 2 2.12.2.3.4.2.5.1=-1
/usr/nbin/grinch -p 2 -A 2.12.2.3.10=1
/usr/nbin/grinch -p 2 -A 2.12.2.3.4.1.5.3=1
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.3.1.1=0
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.3.1.2=155000
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.3.1.3=1
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.32=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.33=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.34=2048
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.35=0
/usr/nbin/grinch -p 2 -A 2.12.2.3.10=128
/usr/nbin/grinch -p 2 -A 2.12.2.3.11.3.12=1
/usr/nbin/grinch -p 2 2.12.2.3.11.3.12.1.1=1
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.31=0
/usr/nbin/grinch -p 2 -A 2.12.2.3.11.3.12.1.2=1
/usr/nbin/grinch -p 2 2.12.2.3.11.3.12.1.2.1.1=10.20.2.237
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.32=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.33=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.34=2048
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.35=0
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.1=00000130 00000000 00000000
    00007fff 00000000 00000000 00000000 00000000 00000002 00000000
    00000000 00000000 00000000 00000001 00025d78 00025d78 00000800
    00000000 00000000
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.1=00000128 0000007f 0000000f
    000001ff 00000000 00000000 00000000 00000000 00000002 00000000
    00000000 00000000 00000000 00000001 00025d78 00025d78 00000800
    00000000 00000000
/usr/nbin/grinch -p 2 2.12.2.3.17.3.1=0
gratm: Sent 0 grinches for card 0x2
```

Here is an error message from **gratm** based on a file with errors:

```
# gratm -n ga0a
gratm: Parse error in "/etc/gratm.conf" file near line 232.
gratm: Input error on 'm' in 'Signalling' section.
# Oct  5 21:25:51 herman gratm: Parse error in "/etc/gratm.conf" file
near line 232.
Oct  5 21:25:51 herman gratm: Input error on 'm' in 'Signalling' section.
```

Verify VPI/VCIs per port

This **maint 13 *port*** command reports the VPI/VCIs that are configured on ***port***.

```
maint 13 0
[RX]
[RX]
[RX] IF  VPI/VCI  TYPE  AAL  ENCAPSULATION
[RX] ------------------------------------------------------------
[RX] 7f  15/511   pvc    5  IPNULL
[RX] 00  0/32767  pvc    5  IPNULL
```

Check ARP entries

Use **maint 8** to check the card's current ARP entries in `/etc/grarp.conf`.

```
GR 2> maint 8
[TX]
[TX] IF  VPI/VCI       IP                   NSAP             STATE
[TX] ------------------------------------------------------------
[TX] 00  0/32767  10.20.2.233               n/a              PERM
[TX] 7f  15/511   10.20.2.237               n/a              PERM
```

Check physical link

Use the **maint 20 *port*** command to verify the port link is up, and verify physical parameters such as mode and timing.

```
GR 2> maint 20 0
GR 2>
[RX]SUNI 0 -mode SDH -timing Internal -loopback [-internal off -line off]
[RX]    -Link up
[RX] TACP-  TSOCI: 00000001  FOVRI: 00000000
[RX] RACP-  OOCDI: 00000001  CHCSI: 00000000  UHCSI: 00000001  FOVRI:
00000000
[RX]        FUDRI: 00000000
[RX] RPOP-  FEBEI: 00000001  BIPEI: 00000001  PYELI: 00000001
     PAISI: 00000000
[RX]        LOPI:  00000001  PSLI:  00000001
[RX] RLOP-  FERFI: 00000001  LAISI: 00000000  BIPEI: 00000001
     FEBEI: 00000001
[RX] RSOP-  BIPEI: 00000001  LOSI:  00000000  LOFI:  00000000
     OOFI:  00000001
[RX]
[RX] Section BIP-8: 00000000  Line BIP-24: 00000040  Line FEBE: 00000050
[RX] Path FEBE: 0000001d Path BIP-8: 0000001e
[RX] Correctable HCS: 00000000  Uncorrectable HCS: 00000000
```

Add/delete PVCs on-the-fly

On ATM OC-3c cards you can add/delete PVCs in the `/etc/gratm.conf` file without rebooting the media card.

There are four steps to add interface `ga03c8` as a PVC on the ATM card in slot 3:

1 Edit `/etc/grifconfig.conf` to reflect the added/deleted PVC:

```
# name   address      netmask        broad_dest      arguments
ga03c8  192.0.130.1   255.255.255.0
```

2 Edit `/etc/gratm.conf` to reflect the added/deleted PVC:

```
# Traffic shaping parameters
Traffic_Shape name=sshq peak=15000 qos=high #slow_speed_high_quality
# Interfaces
Interface ga03c8  traffic_shape=sshq
# PVC's
PVC ga03c8 0/32 proto=ip traffic_shape=sshq
```

3 Use the **gratm -n ga0<*slot*>** command to first check for any errors in `/etc/gratm.conf`:

```
# gratm -n ga03
```

As this command executes, you see numerous messages similar to these:

```
gratm: Accepted traffic shape sshq qos=high for bottom connector card 0.
gratm: Accepted traffic shape sshq qos=high for bottom connector card 1.
gratm: Begin on-the-fly PVC configuration for card 0x3
   /usr/nbin/grinch -p 1 2.12.2.4.17.3.1=1
```

Errors encountered by **gratm** are indicated by a line number where the error is detected. Fix the problem before re-running the **gratm -n** command.

4 Use **gratm ga0<*slot*>** to reconfigure the ATM OC-3c card:

```
# gratm ga03
```

As this command executes, you see numerous messages similar to these:

```
# gratm ga01
gratm: Begin on-the-fly PVC configuration for card 0x3
Oct  2 18:22:57 box1 kernel: ga03c8:  GRF ATM, GRIT address 0:1:0xf0
gratm: Sent 12 grinches for card 0x3
# Oct 2 18:22:57 box1 kernel: ga03c8: GRF ATM, GRIT address 0:1:0xf0
```

Now use the **ifconfig -a** command to see that a new interface is added.

After the ATM OC-3c card is reconfigured, a summary appears in the `grconsole.log` indicating which PVCs were added, which were deleted, and which were updated.

Note: On-the-fly configuration applies only to PVCs. It does not apply to ARP servers or UNI signaling. ARP server and UNI signaling parameters cannot be reconfigured in this way. After ARP server and UNI signaling parameters are configured, the ATM OC-3c card must be reset for new settings to apply.

Rate queue (traffic shape)
Values in a rate queue cannot be changed on the fly. Changes must be made in the `/etc/gratm.conf` file and the ATM media card rebooted.

“Configuring” an SVC

Switched virtual circuits (SVCs) and permanent virtual circuits (PVCs) are configured differently.

On the GRF, SVC configuration actually consists of configuring a set of logical interfaces with sets of characteristics

This example configures two logical interfaces with the following attributes:

– resident on top and bottom physical interfaces, card in slot 3
– on logical interface 20 (hex=14), on logical interface 200 (hex=c8)
– run UNI3.1 protocol and SDH
– are assigned a medium speed path with low priority
– name one or more ARP servers at the specified NSAP addresses

For the example, the GRF can query an ATM ARP server (`arp200`) at NSAP address
`47000555ffe1000000f21513eb0020481513eb00`
for the IP address of a requested destination device (endpoint).

The GRF can also query a second and third ARP server, the address must be unique per ARP server.

Entries in /etc/gratm.conf

- Service section)

  Name an ARP server or servers that will handle SVCs.

```
# ARP Service info
#
Service name=arp200 type=arp \
      addr=47000555ffe1000000f21513eb0020481513eb00
Service name=arp201 type=arp \
      addr=47000555ffe1000000f21513eb0020481513ec00
Service name=arp202 type=arp \
      addr=47000555ffe1000000f21513eb0020481513ed00
```

- Traffic Shaping section

  Define traffic shaping name and quality of service parameters, an SVC assumes the traffic shaping parameters of the logical interface to which it is assigned.

```
# Traffic shaping parameters
Traffic_Shape name=medium_speed_low_quality peak=75000 qos=low
```

- Signalling section

  Set the signalling protocol to UNI 3.1 on both physical interfaces of the card in slot 3, set SDH mode.

```
# Signalling parameters
Signalling card=5 connector=top protocol=UNI3.1 mode=SDH
Signalling card=5 connector=bottom protocol=UNI3.1 mode=SDH
```

- Interface section)

  Identify the logical interfaces (using the `ga0xyz` interface name) to suppoort SVCs.

```
# Interfaces
Interface ga0314  service=arp200   \
             traffic_shape=medium_speed_low_quality

Interface ga03c8  service=arp201   \
             traffic_shape=medium_speed_low_quality
```

Entry in /etc/grifconfig.conf

Assign the IP address to the interface name, a netmask is required.

```
#name  address     netmask       broad_dest   arguments
ga0314 192.160.130.1 255.255.255.0
ga03c8 192.168.10.2 255.255.255.0
```

Entries in /etc/grarp.conf

An entry is needed that maps IP and NSAP addresses if the destination device does not support inverse ARP.

Saving the files

After you use the editor to save and close an `/etc` configuration file, write the file to the `/etc` configuration directory. Use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

SVC creation process

Two ATM devices from different subnets connect to GRF ATM cards through ATM switches. In the example, Device A requests a connection path to Device B.

Assumptions:

- no PVCs are configured for any links
- the following UNIX command to make an entry in the device's route table had previously executed on Device A:
  `route add 222.222.223.0  222.222.222.2`
- the following UNIX command to make an entry in the device's route table had previously executed on Device B:
  `route add 222.222.222.0  222.222.223.2`

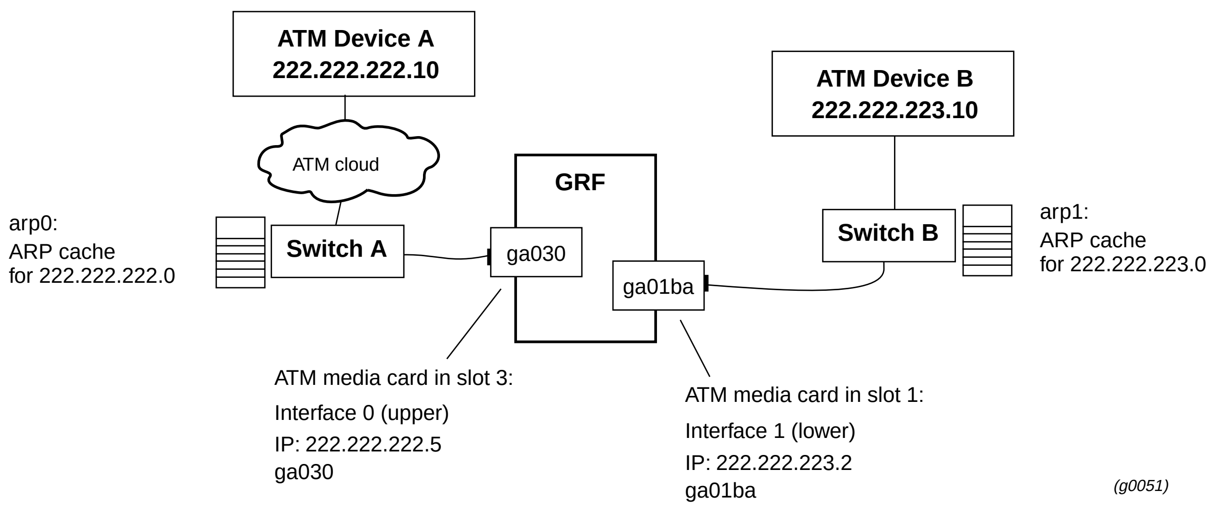


Figure 5-4. GRF role in ATM-ATM connection

Process

Device A:

- Looks up the next hop entry for Device B's destination IP address in its route table.
- Checks its own ARP table for the next hop address for the GRF; since no entry is found, no PVC is in place.
- Requests the GRF's NSAP address from the ARP server **arp0.**
- Using the NSAP address, requests Switch A to set up a connection (virtual path) to the GRF.

Switch A:

- As requested, the switch creates a single full-duplex link between the destination device (the GRF) and the requestor, Device A.

Once the connection is established, packets from Device A flow through Switch A to its ATM connection on the GRF.

The GRF ATM card in slot 3:

– Does a look-up in its route table for the destination (Device B) IP address, finds `222.222.223.10`.
– Finds a subnet entry that happens to be reached via the ATM card in slot 1.
– Requests the NSAP equivalent of the destination IP address from the ARP server (**arp1**) on Switch B.
– Using the NSAP address, requests Switch B to set up a connection (virtual path) to Device B.

Switch B:

– As requested, the switch creates two links, (SVCs) one to the GRF and one to the destination, Device B.

Once the connection is established, packets from the GRF flow through Switch B to Device B.

These are the GRF entries already in `/etc/grifconfig.conf` and `/etc/gratm.conf` that would support the creation of SVCs from the ATM media card in slot 1:

```
# etc/grifconfig.conf
# name  address         netmask         broad_dest        arguments
ga01ba  222.222.223.2   255.255.255.0   222.222.223.10


# /etc/gratm.conf
# ARP Service info
Service name=arp1 type=arp \
              addr=47000580ffe1000000f21513eb0020481513eb00
#
# Traffic shaping parameters
Traffic_Shape name=medium_speed_low_quality  peak=75000 qos=low
#
# Signalling parameters
Signaling card=1 connector=bottom protocol=UNI3.0  mode=SDH
#
# Interfaces
Interface ga01ba  service=arp1 \
                 traffic_shape=medium_speed_low_quality
#
```

Other ATM configuration options

Supply address for ARP service

You need to supply IP-to-physical address mapping information for ARP service ONLY if the remote destination does NOT support inverse ATM ARP. The GRF supports Inverse ATM ARP for determining the IP address of the other end of the VPI/VCI. If the other device does not support Inverse ATM ARP, an ARP entry for the IP and VPI/VCI of the other device must be made in `grarp.conf`.

```
#/etc/grarp.conf
#[ifname] hostname  phys_addr   [temp]   [pub]   [trail]
#
ga0399  192.0.130.111 47000580ffe1000000f21c20e80020481c20e800
```

Changing the transmit clock source

The ATM OC-3c SUNI component has a receive and a transmit clock. The receive clock is always at the SUNI's internal setting. The transmit side clock setting can be toggled between the recovered receive clock (default, the SUNI's own internal clock) and the external oscillator (the clock of the transmitting node).

You can specify the transmit clock permanently in the Signalling section of the `/etc/gratm.conf` file. The example shows how clock is specified for the top interface as internal (the default) and for the bottom interface as external:

```
# Signaling parameters
Signaling card=5 connector=top protocol=UNI3.1  clock=Int
Signaling card=5 connector=bottom protocol=UNI3.1  clock=Ext
```

Transmit clock can be toggled temporarily using the ATM card's **maint 22** command. The setting reverts back to the recovered receive clock (internal) at ATM card reboot and system reset.

Using the **maint 22** ***port value*** command, you can set the top interface's transmit clock to external oscillator. Specify ***value*** as 1:

```
GR 06> maint 22 0 1
```

To set the top interface's transmit clock back to the default (internal, recovered receive clock), specify ***value*** as 0:

```
GR 06> maint 22 0 0
```

These **maint** settings are *temporary*, and revert back to recovered receive clock (0) at ATM card reboot and system reset.

Specifying the IS-IS protocol

Edit the `/etc/gratm.conf` file and specify IS-IS in the protocol field of the interface's PVC statement.

The protocol field enables an interface to run the specified protocol, the options for available protocols are described in `/etc/gratm.conf`. Set the field to `proto=isis`. Here is an example of the PVC statement:

```
pvc ga030 0/40 proto=isis traffic_shape=high_speed_high_quality
```

You also need to create an ISO address entry in `/etc/grifconfig.conf`.

Refer to the *GRF GateD Manual* for information about configuring IS-IS.

Create and assign broadcast groups

The ATM OC-3c media card uses standard broadcast IP group addressing. Broadcast addresses are entered in the Service section of the `/etc/gratm.conf` file. The media card's transmit interface routes broadcast datagrams to each of the members of the broadcast group defined in Service `type=bcast`. Here is an example that also shows broadcast group assignment in the Interfaces section:

```
# Broadcast Service info
Service name=bc0 type=bcast addr=198.174.20.1 addr=198.174.22.1 \
       addr=198.174.21.1
#
#Interfaces
Interface ga090 service=bco traffic_shape=high_speed_high_quality
```

Verify the broadcast groups with **maint 3 3 *port***:

```
maint 3 3 1
[RX] Port 0: LINK UP
[RX] Port 1: LINK UP
[TX] IF  GROUP  BROADCAST GROUP MEMBERS
-----------------------------------------------------------------------
ga092 bc0 198.174.20.1 198.174.22.1 198.174.21.1
```

Bridging

Please refer to the "Transparent Bridging" chapter in this manual for bridging configuration information involving ATM OC-3c media cards.

ATMP

Please refer to the "ATMP Configuration Guide" in this manual for ATMP configuration information involving ATM OC-3c media cards.

Configuring traffic shapes

Peak cell rate, sustained cell rate, and maximum burst size are specified to create a `Traffic_Shape name` in the Traffic Shaping section of the `/etc/gratm.conf` file.

A name can be any string, for example, this shape specifies the best possible service and access to bandwidth resources:

```
Traffic_Shape name=high_speed_high_quality  \
        peak=155000 sustain=155000 burst=2048 qos=high
```

Use a backslash (\) to divide a single long line of characters.

This shape specifies a minimum level of service:

```
Traffic_Shape name=lowest_speed_lowest_quality  \
             peak=10000 burst=64 qos=low
```

Note: Sustained rate defaults to peak cell rate when it is not specified.

You can create as many `Traffic_Shape` names as you need, but you can specify only eight different peak rate queues. At most, there can be four peak rate queues for high QoS, four for low QoS. You cannot borrow from one to increase the other.

– The maximum peak rate is 155520 kilobits/second.

– The maximum sustained rate is 155000 kilobits/second.

– The largest burst size you can specify is 2048 cells.

Peak rate is the only required parameter in a `Traffic_Shape`. If you do not specify a sustained rate, it defaults to the peak rate. If you do not specify a burst size, it also defaults to peak rate. Another optional parameter is Quality of Service (`QoS`). Quality of Service defaults to high priority.

PVCs and logical interfaces are individually assigned a specific `Traffic_Shape name`. An SVC inherits the traffic shape of the logical interface to which it is assigned.

Changing a rate queue

Although a PVC can be added or deleted on-the-fly, values in a rate queue cannot be changed in this way. After you make changes to a rate queue in the `/etc/gratm.conf` file, you must reboot the ATM media card.

Optional: set parameters in the Card profile

Set optional ATM card configuration parameters at the Card profile. Available options are:

– OPTIONAL: specify ICMP throttling settings

– OPTIONAL: change run-time binaries

– OPTIONAL: change dump variables

Media card type, `atm-oc3-v2,` is automatically read into the read-only `media-type` field. Other values shown are defaults. At the top level, you see config and ICMP throttling fields:

```
super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
media-type = atm-oc3-v2
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = <{ 0{off on 10 3} {single off} {"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off}
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

1. Specify ICMP throttling

You can change default ICMP throttling settings in the `icmp-throttling =` field. ICMP throttling messages are described earlier in the chapter or do a `set <field-name>?` for a brief description.

Default values are shown here:

```
super> list icmp
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

Here is how to access the help message for the `echo-reply` field:

```
super> set echo ?
echo-reply:
 The number of ICMP ping responses generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

Change default echo reply and TTL settings with these commands:

```
super> set echo-reply = 4
super> set TTL-timeout = 12
super> write
CARD/8 written
```

You do not have to do a **write** until you have finished all changes in the Card profile. However, you get a warning message if you try to exit a profile without saving your changes.

2. Specify a different executable binary

Card-specific executables can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this ATM OC-3c card only.

```
super> read card 8
card/8 read

super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you want to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_executable_for_ATM OC-3c
super> write
CARD/8 written
```

3. Change default dump settings

Card-specific dump file names can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 8
card/8 read
super> list dump
config = 0
hw-table = < >
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex. Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD/8 written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

To save the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

Optional: change ATM binaries – Load profile

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** ATM cards.

Here is the path, default settings are shown:

```
super> list
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the ATM OC-3c card settings:

```
super> list atm-oc3-v2
type = atm-oc3-v2
rx-config = 0
rx-path = /usr/libexec/portcards/atmq_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/atmq_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

To execute different run-time code on the receive side of the ATM OC-3c card, replace `/usr/libexec/portcards/atmq_rx.run` with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/newatmq_rx.run
super> write
LOAD written
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. You have the option to configure the card's boot sequence so that after booting, the card loads and runs diagnostics before it loads and runs the executable binaries. Set the `enable-boot-seq` field to on and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the "Management Commands and Tools" chapter in this manual for information.

Optional: change ATM dumps – Dump profile

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Default settings are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, default settings are shown:

```
super> read dump
DUMP read

super> list
hw-table = < { hippi 20 var 0 } { rmb 20 var 3} { hssi 20 var 7 }+
dump-vector-table = <{{ 3 rmb "RMB default dump vectors" < { 1 SRAM 2621+
config-spontaneous = off
keep-count = 2
```

The `hw-table` field has settings to specify when dumps are taken and where dumps are stored. Here is the path to examine the ATM OC-3c settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list atm-oc3-v2
media = atm-oc3-v2
config = 20
path = /var/portcards/grdump
vector-index = 5
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP/ written
```

Dump vectors

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for all ATM cards. These fields are read-only and cannot be changed.

Here is the path, **cd** .. back up to the main level if necessary:

```
super> cd ..
super> list dump-vector-table
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" < {1 "atm inst memory" 16+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" < { 1 "fddi/Q CPU0 core mem+
7 = {7 hssi "HSSI default dump vectors" < { 1 "hssi rx SRAM memory" 2097+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" < {1 "Ethernet rx SRA+
9 = {9 dev1 "DEV1 default dump vectors" < { 1 "dev1 rx SRAM memory" 2097+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" < {1 "ATM-12 SDRAM+
11 = {11 sonet-v1 "SONET default dump vectors" < {1 "SONET rx SRAM memor+
```

This sequence shows a portion of the areas in the ATM OC-3c card that are dumped:

```
super> list 5
index = 5
hw-type = atm-oc3-v2
description = "ATM/Q default dump vectors"
segment-table = <{ 1 "atm inst memory" 16777216 4194304}{2 "SAR0-TX cont+

super> list seg
1 = { 1 "atm inst memory" 16777216 4194304 }
2 = { 2 "SAR0-TX control memory" 50462720 131072 }
3 = { 3 "SAR0-RX control memory" 50593792 131072 }
4 = { 4 "SAR1-TX control memory" 50724864 131072 }
5 = { 5 "SAR1-RX control memory" 50855936 131072 }
6 = { 6 "dual port memory" 33554432 32768 }
7 = { 7 "shared memory" 50331648 131072 }

super> list 1
index = 1
description = "atm inst memory"
start = 16777216
length = 4194304

super> list s 7
index = 7
description = "shared memory"
start = 50331648
length = 131072
```

Getting ATM data and statistics

This section describes the use of **maint**, **grarp -a**, **gratm**, and **grstat ip** and **grstat l2** commands to obtain ATM OC-3c card information.

maint commands for ATM OC-3c media cards

maint commands display a range of information about a specific type of media card. Each media card has its own set of **maint** commands. The same **maint** command may work on more than one media card.

The ATM OC-3c card has individual processors for the transmit and receive sides, and two sets of **maint** commands. One set covers the receive (RX) side and include some commands applicable to the card overall. The second set covers the transmit (TX) side. Transmit side counterparts of receive side commands use the same number but are 100-based. For example, the receive side **maint 8** is transmit side **108**.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

```
# grrmb
```

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the ATM media card you are working with. For example, if you are working with a card in slot 2, enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 2
GR 2>
```

To leave the **maint** prompt, enter **`quit`**.

Receive / transmit side maint commands

Use **maint 1** to see the list of **maint** commands for the receive side, use **maint 101** to see the list for the transmit side.

Receive side list

```
# grrmb
GR 2> maint 1
[RX]
GR 2> [RX] Got unknown RX maint command 1
[RX] Perhaps what you are after is on the TX side (try maint 101)
[RX]
[RX]   1:    Display this screen of options for the RX side
            (has TX counterpart)
[RX]   2:    Display RX Version Numbers
```

```
[RX]   3:    Display Active Interfaces
[RX]   4:    Display ATM Media Statistics [ port ]
[RX]   5:    Display Switch Statistics
[RX]   6:    Display Combus Statistics
[RX]   7:    Clear Counters
[RX]   8:    Display ARP Entries
[RX]  10:    Memory & Buffer Usage (has TX counterpart)
[RX]  13:    VPI/VCI Configuration [port] (has TX counterpart)
[RX]  14:    Traffic Stats Per VPI/VCI [ port ]
[RX]  15:    Errors Per VPI/VCI [ port vpi vci ]
[RX]  16:    Show F5 OAM stats for a VC [port vpi vci]
[RX]  20:    Display SUNI Statistics [ port ]
[RX]  21:    Select SUNI Framing Mode [ port SONET=0 SDH=1 ]
[RX]  22:    Select SUNI timing source [ port 0=internal 1=external ]
[RX]  23:    Select SUNI local loopback [ port 0=off 1=on ]
[RX]  24:    Select SUNI line loopback [ port 0=off 1=on ]
[RX]  30:    Switch Test: Setup [ pattern length slots... ]
[RX]  31:    Switch Test: Start [ slots...]
[RX]  32:    Switch Test: Stop [ slots... ]
[RX]  33:    Switch Test: Status
[RX]  37:    Setup Raw Route [ port vpi vci card port vpi vci ]
[RX]  42:    Print FRTLU route table
[RX]  60:    Enable/Disable KERN_TRACE
[RX]  61:    Display KERN Trace [ num_traces ]
[RX]  62:    Display Switch Descriptor Ring [num_entries]
[RX]  63:    Display SAR Receive Descriptor Ring [ port num_entries ]
[RX]  64:    Display Receive State
[RX]  70:    Display ATMP Home Network table
[RX]  73:    Display Mobile Node Tree
[RX]  80:    Laser Control [port] [0 = OFF 1 = ON]
[RX]  45:    List next hop data: [family]
[RX]  50:    Filtering filter list: [detail_level [ID]]
[RX]  51:    Filtering filter list: [detail_level [IF]]
[RX]  52:    Filtering action list: [detail_level [ID]]
[RX]  53:    Filtering action list: [detail_level [IF]]
[RX]  54:    Filtering binding list: [detail_level [ID]]
[RX]  55:    Filtering binding list: [detail_level [IF]]
[RX]  56:    Display filtering statistics: [IF#]
[RX]  57:    Reset filtering statistics: [IF#]
[RX]  58:    Show filter protocol statistics
[RX]          note, IF/ID may be '-1' to indicate all of the given item
[RX]               while detail level is 0|1|2.
```

Transmit side list

```
GR 2> maint 101
[RX]
[TX]  101:    Display this screen of options for the TX side
              (maint 1 for RX side)
[TX]  108:    Display ATMARP Entries
[TX]  109:    Display ATMARP Server Info
[TX]  110:    Memory & Buffer Usage
[TX]  113:    VPI/VCI Configuration [port]
[TX]  118:    Display broadcast groups and their members
[TX]  125:    Display Rate Queues [ port ]
[TX]  126:    Setup Rate Queue [ port queue rate(kb) ]
```

```
[TX]  127:    Teardown Rate Queue [ port queue ]
[TX]  134:    Display QSAAL [ port ]
[TX]  135:    Display Q93B [ port ]
[TX]  136:    Display UME [ port ]
[TX]  139:    Setup ATMARP Entry [ if vpi vci ip ]
[TX]  140:    Teardown ATMARP Entry [ port vpi vci ip ]
[TX]  141:    Display LANARP Entries [ if ]
[TX]  142:    Display bridging VC configuration [ if ]
[TX]  160:    Enable/Disable KERN_TRACE [ 0 | 1 ]
[TX]  162:    Display Switch Descriptor Ring [num_entries]
[TX]  161:    Display KERN Trace [ num entries ]
[TX]   70:    Display ATMP Home Network table
[TX]  116:    Show F5 OAM stats for a VC [port vpi vci]
[TX]  145:    List next hop data: [family]
[TX]  150:    Filtering filter list: [detail_level [ID]]
[TX]  151:    Filtering filter list: [detail_level [IF]]
[TX]  152:    Filtering action list: [detail_level [ID]]
[TX]  153:    Filtering action list: [detail_level [IF]]
[TX]  154:    Filtering binding list: [detail_level [ID]]
[TX]  155:    Filtering binding list: [detail_level [IF]]
[TX]  156:    Display filtering statistics: [IF#]
[TX]  157:    Reset filtering statistics: [IF#]
[TX]  158:    Show filter protocol statistics
[TX]          note, IF/ID may be '-1' to indicate all of the given
[TX]          item while detail level is 0|1|2.
```

Examples of ATM maint displays

Display active interfaces

The **maint 3** command gives you useful options for looking at a variety of active interfaces.

```
GR 2> maint 3
[RX] maint 3 1 0 -- IP config per IF port 0
[RX] maint 3 1 1 -- IP config per IF port 1
[RX] maint 3 2 0 -- VC config per IF port 0
[RX] maint 3 2 1 -- VC config per IF port 1
[RX] maint 3 3 0 -- BROADCAST GROUP config per IF port 0
[RX] maint 3 3 1 -- BROADCAST GROUP config per IF port 1
[RX] maint 3 4 0 -- NSAP config per IF port 0
[RX] maint 3 4 1 -- NSAP config per IF port 1
[RX] maint 3 5 0 -- ARP SERVER config per IF port 0
[RX] maint 3 5 1 -- ARP SERVER config per IF port 1
```

Check IP addresses

You can list the IP addresses configured to the logical interfaces on the specified port:

```
GR 2> maint 3 1 0
[RX] Port 0: LINK UP
[RX] Port 1: LINK DOWN
[RX]
[RX] IF       IP         STATE  |  IF        IP        STATE
-------------------------------------------------------------------
[RX] 00  10.20.2.234       UP      7f  10.20.2.238       UP
```

Check virtual circuits

You can list the VPI/VCIs configured to each logical interface on either ATM port.

```
maint 3 2 0
[RX]
[RX] Port 0: LINK UP
[RX] Port 1: LINK DOWN
[RX]
[RX] IF  VPI/VCI  ENCAPS    IF  VPI/VCI  ENCAPS    IF  VPI/VCI  ENCAPS
------------------------------------------------------------------------
[RX] 7f  15/511   IPNULL    00   0/32767 IPNULL
```

Display ATM media statistics

To look at media information per port, use **maint 4** and the port number, 0 or 1:

```
GR 02> maint 4 1
GR 02> [RX]
[RX]                    RX SARA STATISTICS
------------------------------------------------------------------------
[RX] RX Packets: 00000000000000090848
[RX]                    RECEIVE ERRORS
[RX] PCQ Overflow:              0      LBQ underflow:           0
[RX] Overflows:                 0      Timeouts:                0
[RX] Parity Errors:             0
[RX] Invalid VPI:               0      Invalid VCI:             0
[RX] COM errors:                0      EOM errors:              0
[RX] SB underruns:              0      LB underruns:            0
[RX] PTY errors:                0      EOP errors:              0
[RX] RSE errors:                0      CRC errors:              0
[RX] Raw cells:
[TX]
[TX]                    TX SARA STATISTICS
------------------------------------------------------------------------
[TX] TX Packets: 00000000000068049018
[TX]
[TX]                    TRANSMIT ERRORS
[TX] Bank A Miss:               0      Bank B Miss:             0
[TX] CBR Parity:                0      Descriptor:              0
[TX] Packet parity:             0      CM parity:               0
```

RX Packets   The cumulative count of packets received on this port. This count includes all packets that were successfully reassembled, and packets that were not successfully reassembled due to ATM layer errors such as CRC errors, etc. For this reason, the count of RX packets in **maint 4** will often exceed the sum of per VC counters in **maint 14** ***port***, because the per VC counters only track packets that were sucessfuly reassembled.

PCQ overflow (Packet Complete Queue)   A packet was sucessfully reassembled, but there was no free space in the Packet Complete Queue for the packet descriptor, so the packet was discarded.

LBQ underflow (Large Buffer Queue)

Overflow (buffer overflow) The received PDU is larger than the system buffer and reassembly was terminated, the packet was discarded.

Timeouts The packet did not complete reassembly because not all of the packet cells arrived before the packet timer expired, the packet was discarded.

Parity Errors A count of the number of parity errors that occurred while accessing the SAR's control memory.

Invalid VPI A cell arrived on a VPI which was not configured. The lookup in the SAR's reassembly table did not find an entry for this VPI.

Invalid VCI A cell arrived on a VCI which was not configured. The lookup in the SAR's reassembly table did not find an entry for this VCI.

COM Errors (Continuation Of Message) An out of sequence COM cell was received.

EOM Errors (End Of Message) An out of sequence EOM cell was received.

SB underruns (Small Buffer) A packet (the first cell of the packet) was received from the media, but there were no small buffers available in which to begin reassembly, so the packet was discarded.

LB underrun (Large Buffer) A packet (the first cell of the packet) was received from the media, but there were no large buffers available in which to begin reassembly, so the packet was discarded. (All AAL 5 PDUs go into large buffers.)

PTY errors (Parity error in cell payload) When transferring data from the SONET/SDH framer to the SAR, a parity error occurred.

EOP errors (End Of Packet) The last cell of the packet was not received.

RSE errors (Roll Over Sequence) Some AALs use a sequence number. AAL 5 does not, but AAl 3/4 does, for example. An RSE indicates that a roll over sequence error occurred at packet boundaries. An example of this is the first cell of this packet did not have the sequence number succeeding the most recently received cell on this VC.

CRC errors The CRC computed by the SAR over the data portion of the PDU did not match the CRC stored in the PDU itself, so the packet was discarded. This can happen because of payload corruption, or because of cells being discarded by switches, resulting in the CRC being computed over a portion of the PDU instead of the entire PDU.

Raw cells The number of cells placed on the SAR's Raw Cell Queue, typically OAM F5 and congestion notification cells.

Bank A Miss This indicates that a rate queue in bank A missed getting serviced. This could happen because the rate queues are oversubscribed or because the link interface is not accepting cells from the SAR.

Bank B Miss This indicates that a rate queue in bank B missed getting serviced. This could happen because the rate queues are oversubscribed or because the link interface is not accepting cells from the SAR.

CBR Parity Indicates the presence of a parity error in CBR or AAL5 packet data when segmenting a packet. Further segmentation of this packet is aborted.

Descriptor Generic transmit error: either the packet did not complete segmentation because the VC was flushed, or there was a parity error during semgentation.

Packet parity Indicates that a parity error was detected in AAl3/4 segmentation. Further segmentation of this packet is aborted.

CM parity Indicates that a parity error was detected during a read of SAR control memory.

Display switch statistics

This command returns information about the number of packets to and from the switch.

```
GR 02> maint 5
[RX]                    Switch Statistics
[RX] input:
[RX]         Bytes              Packets       Errors   Overruns
[RX] ---------------------------------------------------------
[RX] 0000000000269137056 0000000000000228474 000000000 000000000
[RX]
[RX] output:
[RX]         Bytes            Packets          Errors   Overruns
[RX] -------------------------------------------------------------
[RX] 0000000000284854392 0000000000000374760 000000000 000000000
[RX]
[RX] Switch Transmit Data Errors:            0
[RX] Switch Transmit Fifo Parity Errors:     0
[RX] Switch Transmit Internal Parity Errors:  0
[RX] Switch Transmit Connection Rejects:     0
[RX] Switch Transmit Unclassified Errors:    0
[RX] Switch Transmit Last Error Status:      0x00000000
[RX] Switch Receive Encoding Errors:         0
[RX] Switch Receive Running Disparity Errors: 0
[RX] Switch Receive Receiver Errors:         0
[RX] Switch Receive Running Checksum Errors:  0
[RX] Switch Receive FIFO Overflow Errors:    0
[RX] Switch Receive Unclassified Errors:     0
[RX] Switch Receive Last Error Status:       0x00000000
```

RX Packets
A count of the number of packets received from the switch.

TX Packets
A count of the number of packets transmitted across the switch.

RX Bytes
The count of bytes received from the switch.

TX Bytes
The count of bytes transmitted across the switch

Memory and buffer statistics

The **maint 10** (receive side) and **maint 110** (transmit side) commands display usage on receive and transmit side memories and buffers. They report free, fragmented, and available units. The transmit side is the same as receive except that it does not have combus buffers and it has transmit data buffers instead of receive.

```
GR 2> maint 10
[RX]
[RX] TYPE    UNIT    TOTAL       FREE       N-FRAGS     LRG-FRAG    AVAIL
---------------------------------------------------------------------------
[RX] IN-MEM  00001  1485708     0901588     0000001     0901588    -------
[RX] SH-MEM  00001  0063352     0000232     0000002     0000216    -------
[RX] SHD-BF  02048  0000010     0000010     -------     -------    -------
[RX] COM-BF  00768  0000128     0000127     -------     -------    -------
[RX] RCV-BF  16384  0000256     0000006     -------     -------    0000248
[RX] VCT-0   -----  0001024     0001022     -------     -------    0000002
[RX] VCT-1   -----  0001024     0001024     -------     -------    0000000
```

TYPE - which memory or buffer type

UNIT - number of bytes per unit

TOTAL - number of units available for TYPE

FREE - number of units actually free

N-FRAGS - number of memory fragments

LRG-FRAG - largest “fragment” of memory that a **malloc** could obtain

AVAIL - available buffers or configured VCs

IN-MEM - instruction memory

SH-MEM - shared memory

SH-BUF - shared buffers

COM-BF - combus buffers, only relevant on receive side

RCV-BF - receive buffers

VCT-0 - number of entries in the Virtual Circuit table for port 0.

VCT-1 - number of entries in the Virtual Circuit table for port 1.

- - - - - value does not apply to this type

VPI/VCI configuration

You can return information about VPI/VCIs on a per port, per side basis:

```
GR 02> maint 13 1
[RX]
[RX] IF  VPI/VCI  TYPE  AAL  ENCAPSULATION
[RX] ------------------------------------------------------------------
[RX] f1  15/32    pvc    5   IP-LLC       pt-pt
[RX] f0  15/511   pvc    5   IPNULL
```

VPI/VCI traffic statistics

You can display VPI/VCI traffic statistics on a per port, per side basis:

```
GR 2> maint 14 0
[RX] RECEIVE:
[RX] IF  VPI/VCI    PACKETS       BYTES    IP DISCARD  UNSUPP LLC
-----------------------------------------------------------------------
[RX] 7f  15/511    0000374512  0271098192  0000000000       n/a
[RX] 00  00/32767  0000000003  0000000288  0000000000       n/a
[TX]
[TX] TRANSMIT:
-----------------------------------------------------------------------
[TX] 7f  15/511    0000221345  0265043424
[TX] 00  00/32767  0000000003  0000000288
```

RX/TX Packets
A count of the sucessfully reassembled packets received on this VC (see the **maint 4** RX packet count for more information).

RX/TX Bytes
The bytes (n*48) received from/transmitted to the media.

IP Discards
A count of IP Packets received on this VC which were subsequently dropped by the IP forwarding engine.

UNSUPP LLC
On those VCs which use LLC/SNAP encapsulation, indicates the count of packets of an LLC/SNAP type not supported by the GRF.

Display rate queues

This command displays information about the rates queues configured per port:

```
GR 2> maint 125 0
[TX] RQ   State      Rate(Kbs)       VPCIs
------------------------------------------------------------------------
[TX] 00  ENABLE       155000       15/511  0/32767
[TX] 01  DISABLE
[TX] 02  DISABLE
[TX] 03  DISABLE
[TX] 04  DISABLE
[TX] 05  DISABLE
[TX] 06  DISABLE
[TX] 07  DISABLE
```

Display F5 OAM statistics per port VPI/VCI

```
GR 2> maint 16 0 15 511
[RX]
[RX] OAM F5 Stats for Port 0 VPI/VCI 15/511
[RX] Receive
------------------------------------------------------------------------
[RX]                            End to End                 Segment
[RX] Fault Management
[RX]    AIS:                             0                        0
[RX]    RDI:                             0                        0
[RX]    Loopback:                        0                        0
[RX]    Continuity:                      0                        0
[RX]
[RX] Performance Management
[RX]    BWD Reporting:                   0                        0
[RX]    FWD Monitoring:                  0                        0
[RX]    Monitoring & Reporting:          0                        0
[RX]
[RX] Activation/Deactivation
[RX]    Continuity:                      0                        0
[RX]    Performance Mon:                 0                        0
[TX]
[TX] OAM F5 Stats for Port 0 VPI/VCI 15/511
[TX] Transmit
------------------------------------------------------------------------
[TX]                            End to End                 Segment
[TX] Fault Management
[TX]    AIS:                             0                        0
[TX]    RDI:                             0                        0
[TX]    Loopback:                        0                        0
[TX]    Continuity:                      0                        0
[TX]
[TX] Performance Management
[TX]    BWD Reporting:                   0                        0
[TX]    FWD Monitoring:                  0                        0
[TX]    Monitoring & Reporting:          0                        0
[TX]
[TX] Activation/Deactivation
[TX]    Continuity:                      0                        0
[TX]    Performance Mon:                 0                        0
```

Display ATMP information

Three **maint** commands return ATMP related information.

```
[RX] H O M E  N E T W O R K   T B L  :
[RX] ================================
[RX] S: Slot, P: Port, Rx: packets Received, BRx: Bytes Received
[RX]                  RTx packets transmitted, BTx: Bytes transmitted
[RX]
[RX]FRT-index  S:P:s0:s1     State     Address   VPN Address  VPN Netmask
[RX]--------   ---------     -----     -------   -----------  -----------
[RX]   6     2:01:888:0  LocCir,HA 10.9.9.9 10.20.2.237 255.255.255.0
```

The columns are defined as follows:

- **FRT-index**:
  Foreign agent Route Table, an arbitrarily-assigned tunnel number, not the tunnel ID, but the number you use in the **maint 73** command to display the tunnel ID
- **S:P:s0:s1 on Frame Relay (HSSI card)**:
  S = slot, P = port, s0 = DLCI number of the dedicated link to the home network (`pvcatmp`), s1 is currently unused and is always 0
- **S:P:s0:s1 on ATM**:
  S = slot, P = port, s0 = ATM VPI, s1 ATM VCI (`vc_atmp`)
- **State**
  indicates configuration and functional status depending upon the type/role of card.
  - On the home agent card (usually an Ethernet card acting as the home agent)

  The foreign agent uses an IP address on this card for tunnel negotiation. A state of `HomeAgent` indicates a home agent is properly configured to this card in `aitmd.conf`.
  - On any HSSI or ATM card on a GRF home agent, `State` can have three values:

  1. `HomeAgent` indicates that the home agent is properly configured in `/etc/aitmd.conf`, but this is not the link to the home network.
  2. `LocalCirc` indicates that this link is not part of ATMP configuration.
  3. `LocCir,HA` = the circuit to the home network is configured on this media card.
- **Address** is the IP address of the associated home agent (`atmp0`).
- **VPN Address** is the private network address the customer assigns to the interface that has the link to a home network, it only appears if entered in `/etc/aitmd.conf` file.
- **VPN Netmask** is the netmask for the VPN address.

Please ignore the “`Rx: packets Received, BRx: Bytes Received`” and “`RTx packets transmitted, BTx: Bytes transmitted`” headers, they no longer apply.

Then use **maint 73 *FRT_index*** command to display tunnel information “toward” the foreign agent, including tunnel IDs:

```
GR 2> maint 73 6
GR 2>
[RX] Mobile node tree list for home network index 6
[RX]  Mobile Node /Mask Flags Foreign Agent Tunnel Id  S:P:s0:s1
[RX]  10.4.0.0    /16   0  => 206.146.164.5 0x00000503 02:01:888:0000
```

The columns are as follows:

- mobile node non-routable IP address
- number of bits in the address netmask
- the route flags column is not currently used
- foreign agent routable IP address
- tunnel ID, in this case, 0x503
- **S:P:s0:s1 on Frame Relay (HSSI card):**
  S = slot, P = port, s0 = DLCI number of the dedicated link to the home network (`pvcatmp`), s1 is currently unused and is always 0
- **S:P:s0:s1 on ATM:** S = slot, P = port, s0 = ATM VPI, s1 ATM VCI (`vc_atmp`)

Toggle single mode laser

Use the **maint 80** command to toggle the single mode laser on or off.

The format is **maint 80** ***port*** **0** | **1** where 0 = off and 1 = on.

```
GR 2> maint 80 1 1
[RX]
[RX] Enable Laser on Port 1

GR 2> maint 80 1 0
[RX]
[RX] Disable Laser on Port 1
```

List next hop data

This **maint** command returns a list of the next hop data. You can also use the **grrt -S -p 1** command to obtain the same information, an example is at the bottom of this page.

```
maint 45
[RX] Location is: 0
[RX] Add:       48 Delete:       11 noNH:        0
[RX]    0: 206.146.160.133 (1   )   2:00  0:fc   RMS
[RX]    1: 0.0.0.0         (1   )   2:00  0:fc   MCAST
[RX]    2: 127.0.0.1       (1   )   2:00  0:00   RMS
[RX]    3: 0.0.0.0         (1   )   2:00  0:fc   UNREACH
[RX]    4: 0.0.0.0         (1   )   2:00  0:00   UNREACH
[RX]    5: 0.0.0.0         (1   )   2:00  0:00   MCAST
[RX]    6: 206.146.160.1   (1   )   2:00  0:fc   RMS
[RX]    7: 206.146.160.151 (1   )   2:00  0:fc   RMS
[RX]    8: 206.146.160.132 (1   )   2:00  0:fc   RMS
                  •
                  •
                  •
[RX]   30: 0.0.0.0         (12  )   2:00  3:05   BCAST
[RX]   31: 0.0.0.0         (12  )   2:00  3:05   LOCAL
[RX]   32: 0.0.0.0         (12  )   2:00  3:05   FWD
[RX]   33: 0.0.0.0         (9   )   2:00  3:07 * BCAST
[RX]   34: 0.0.0.0         (9   )   2:00  3:07 * LOCAL
[RX]   35: 10.20.2.237     (73  )   2:00  2:7f   FWD
[RX]   36: 10.20.1.182     (12  )   2:00  3:05   FWD
[RX] Location is: 1
[RX] Add:        0 Delete:        0 noNH:        0
```

These are the columns of interest:

```
[RX]   36: 10.20.1.182     (12  )   2:00  3:05   FWD
```

Entry #

Source address

Next hop index #

Slot, interface #
of destination media card

grrt next hop information

Here are a few lines of **grrt** output:

```
# grrt -S -p 1
default                          3    0.0.0.0          RMS       UNREACH
0.0.0.0         255.255.255.255  7    0.0.0.0          RMS       DROP
127.0.0.0       255.0.0.0        5    0.0.0.0          RMS       UNREACH
127.0.0.1       255.255.255.255  2    127.0.0.1        RMS       RMS
198.174.11.0    255.255.255.0    6    206.146.160.1    RMS       RMS
203.1.10.156    255.255.255.255  77   0.0.0.0          gf0d2     LOCAL
203.1.10.255    255.255.255.255  76   0.0.0.0          gf0d2     BCAST
203.3.10.0      255.255.255.0    68   0.0.0.0          gf081     FWD
203.3.10.156    255.255.255.255  67   0.0.0.0          gf081     LOCAL
203.3.10.255    255.255.255.255  66   0.0.0.0          gf081     BCAST
```

Display ARP information

The **maint 3 5 0 and maint 3 5 1** commands display the ARP server configuration per port. The first example indicates no servers are assigned to the two ATM interfaces:

```
GR 2> maint 3 5 1
[RX]
[RX] Port 0: LINK UP
[RX] Port 1: LINK DOWN
[TX]
[TX] 0x7f       none
[TX] 0x00       none
```

This example shows the ARP server assigned to interface `ga052e`:

```
GR 2> maint 3 5 0
[RX]
[RX] Port 0: LINK UP
[RX] Port 1: LINK UP
[TX]
[TX] 0x2e       47000555ffe1000000f21513eb0020481513eb00
[TX] 0x00       none
```

The **maint 8** and **maint 108** commands display the ARP table for a specific ATM card. The row of question marks indicates the field is not applicable. In this case, you will see either n/a or the question marks.

```
GR 9> maint 8
[TX]
[TX] IF  VPI/VCI       IP                  NSAP              STATE
[TX]
------------------------------------------------------------------------
[TX] 00  0/100  204.101.11.158  ??????????????????????????????????  112



GR 2> maint 108
[TX]
[TX] IF  VPI/VCI       IP                  NSAP               STATE
[TX]
------------------------------------------------------------------------
[TX] 00  0/32767 10.20.2.233              n/a               PERM
[TX] 7f 15/511   10.20.2.237              n/a               PERM
```

The **maint 141** ***port*** displays LANARP information

```
GR 2> maint 141 0
[TX]
[TX]
[TX]    Arp Table for Interface 0:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
```

The **maint 139** and **maint 140** commands have been replaced by **grarp -a**.

Use grarp -a to display ARP addresses

The **grarp -a** command displays the contents of the GRF system ARP table.

```
# grarp -a
box1.minnet.com (206.146.160.1) at 0:c0:80:89:15:e5
box2.minnet.com (206.146.160.131) at (incomplete)
box3.minnet.com (206.146.160.132) at 0:a0:24:a3:c:36
box4.minnet.com (206.146.160.133) at 0:c0:80:86:14:e2 permanent
box5.minnet.com (206.146.160.202) at 0:e0:1e:5d:a4:7f
ga027f  (73): 10.20.2.237 at VPI=15, VCI=511 permanent
ga020   (66): 10.20.2.233 at VPI=0, VCI=32767 permanent
ge035   (12): 10.20.1.182 at 0:60:70:ac:38:20
ge035   (12): 10.20.1.131 at 0:c0:80:8f:8a:3
#
```

Use grstat ip to look at layer 3 statistics

```
# grstat ip ga02
card 2 (2 interfaces found)
  ipstat totals
       count description
      375149 total packets received
      368375 packets forwarded normally
           3 packets handled by the card
        6771 packets forwarded to the RMS
        6493 multicast packets received
        6493 multicast packets sent to RMS
  ipdrop totals
       count description
```

Use grstat l2 to look at layer 2 statistics

```
# grstat l2 ga02
card 2
  Layer 2 statistics
    physical port 0
               count description
              374899 RX packets
           271135344 RX bytes
              221732 TX packets
           265103616 TX bytes
    physical port 1
               count description
```

Collect data via grdinfo

With a single command, **grdinfo** collects the output from nearly all of the ATM OC-3c **maint** commands and compresses it in a log file. Refer to the “Management Commands and Tools” chapter in this manual for more information.

FDDI Configuration

6

Chapter 6 describes the implementation of FDDI on the GRF and other features supported on the FDDI media card. It includes the information needed to configure FDDI interfaces and parameters in the `/etc/grifconfig.conf` file and CLI profiles.

Chapter 6 contains these topics:

Note: This release supports only version 2 of the FDDI media card, FDDI/Q. The first version of FDDI, sometimes referred to as “FDDI classic,” is supported only in 1.3 and earlier releases.

In this manual, FDDI refers to the FDDI/Q media card. Profiles use `fddi-v2` to describe the FDDI/Q card.

FDDI implementation

Single attach (SAS)

Single attach FDDI interfaces can be either master (M) ports or slave (S) ports. They require a cable with a corresponding master or slave connector. Single attach cables have an M connector at one end and an S connector on the other. With no key installed, both M and S connectors fit the FDDI interface.

A single attach FDDI interface on the GRF is a master port when it directly connects to a workstation. As shown in Figure 6-1, it is a slave port when connected to the master port of a FDDI concentrator. Such concentrators connect, in turn, to the slave ports of single-attach workstations.

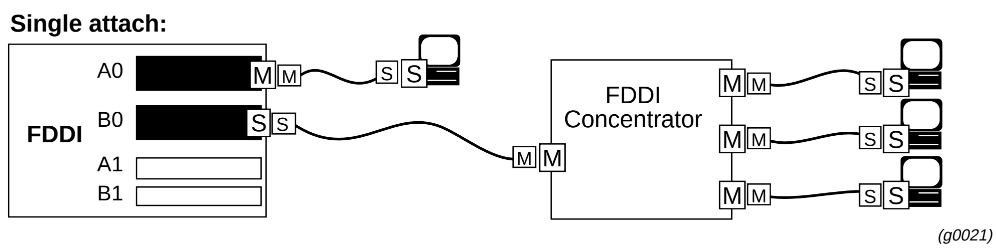


Figure 6-1. Master/slave connector keys for single-attach interfaces

Dual attach (DAS)

Dual attach interfaces connect to form two unbroken counter-rotating rings. Each interface, or station, has both an A and a B port.

Dual attach cables have an A connector on one end and a B connector on the other. As shown in Figure 6-2, the A port connects a station to its downstream neighbor; the B port connects a station to its upstream neighbor.

To create a logical ring, A must connect to B and B must connect to A. Otherwise, the network does not operate as a logical ring, but segments into unconnected subrings.

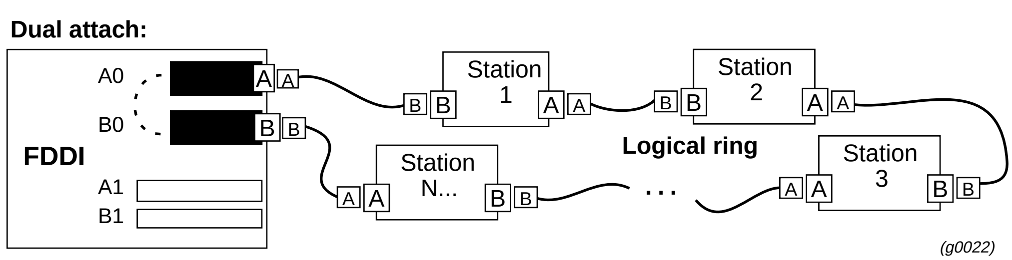


Figure 6-2. A/B connector keys for dual-attach interfaces

Optical bypass switch interface

Optical bypass capability is provided externally. The FDDI face plate has a six-pin DIN connector to directly attach a single bypass switch.

As shown below, two bypass switches can be attached with the Ascend-supplied Y-cable adapter. The Y-cable is required to reconcile control pin assignments between the GRF and the external switch module. Through the Y-cable, an optical bypass switch module attaches to a pair of media interface connectors on the FDDI card.

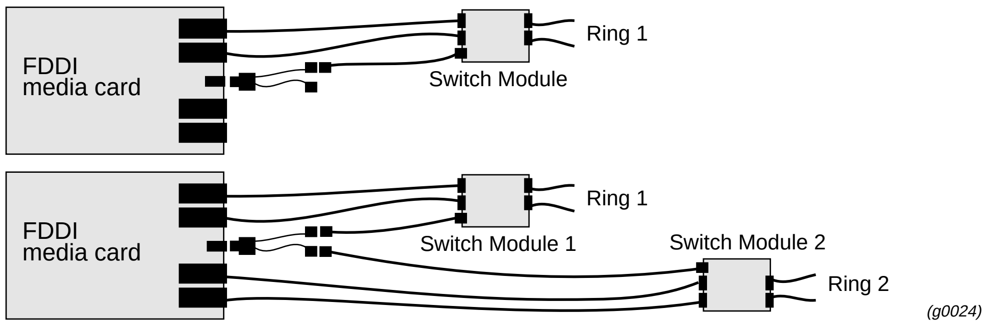


Figure 6-3. Optical bypass switch attachments

A bypass switch allows the GRF to remove itself from the dual ring during a failure or maintenance without causing the ring to “wrap” at upstream and downstream neighbors. Should a GRF failure occur, the bypass switch connects upstream and downstream neighbors on both the primary and secondary rings, and allows the GRF node to remove itself from the ring while still retaining ring continuity.

A node failure without a bypass switch causes the dual ring to “wrap.” A wrapped ring absorbs the secondary ring into the primary ring and no longer has a backup ring.

Support for dual homing

Dual homing provides redundant connectivity between a FDDI media card and a single ring.

Configure the FDDI media card for dual attach, but use two single attach (SAS) cables to connect to two M ports. As shown in Figure 6-4, the M ports can be on either one or two FDDI concentrators on the ring.

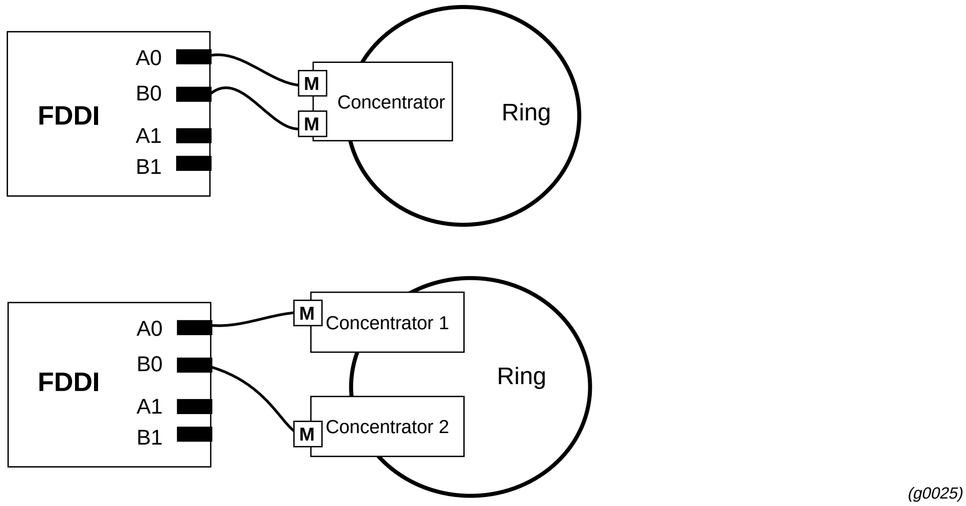


Figure 6-4. Dual homing options

Installing FDDI connector keys

Physical interface (connector) keys are site-installed according to site practice. FDDI media cards are each shipped with a key set for each physical interface.

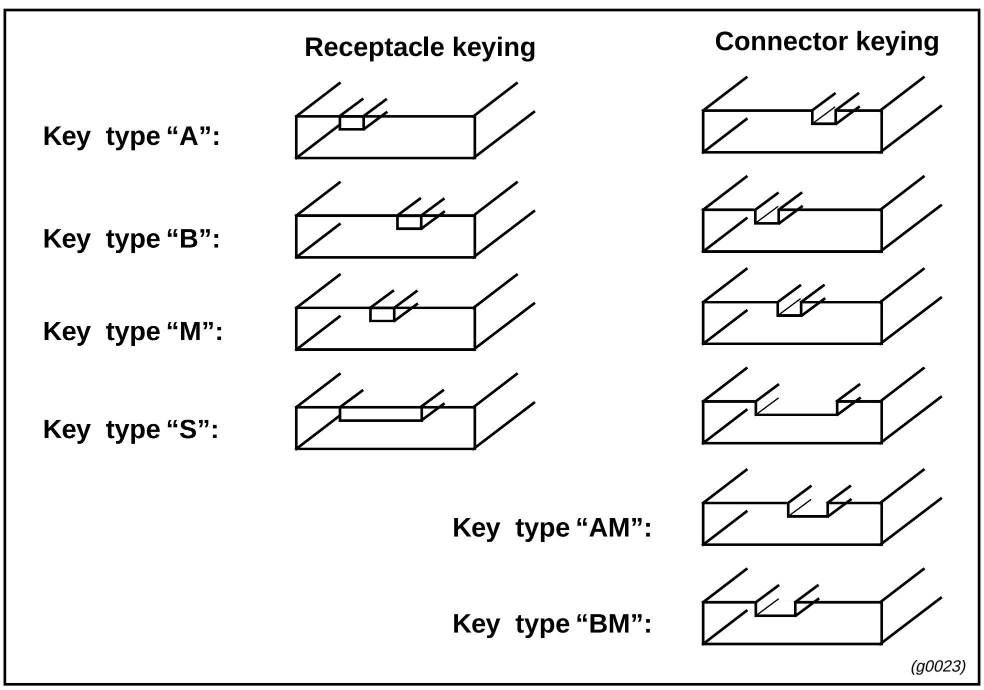


Figure 6-5. Types of FDDI connector keys

Basic functionality

Connector keys are physically installed in an FDDI interface. Once installed, a key limits the type of FDDI cable that can be inserted into that interface. Different cables are matched to single and dual attached interfaces. Cables and interface ports are labeled or “keyed” so they will connect only to a compatible interface type. Figure 6-5 illustrates different types of receptacle and connector keys:

Removing keys from FDDI card interfaces is sometimes difficult. Keys can break, and the FDDI media card must be removed from the chassis to remove a key.

On the FDDI card, the interfaces extend far enough to let the keys pop out without removing the card.

Existing faceplate A and B labels provide built-in “keys” for dual attach configurations. As a single attach, the interface accepts both master and slave cable connectors without affecting configuration.

MTU

The FDDI maximum transmission unit (MTU) size is set at 4352 bytes per packet. You can specify another value in the `grifconfig.conf` file.

FDDI on the GRF

The FDDI card has transmit and receive side processors, CPU0 and CPU1, and two sets of **maint** commands. One set acts on CPU0, a second set, the **maint 70** commands, acts on CPU1. **maint** commands are described near the end of this chapter.

Large route table support

FDDI/Q card software maintains route tables containing up to 150K entries, and hardware support for full table lookups.

Transparent bridging

The GRF implements IEEE 802.1d transparent bridging on GRF Ethernet and FDDI interfaces. A FDDI interface may simultaneously bridge layer-2 frames and route layer-3 packets--that is, forward frames destined to a system attached to another LAN at the MAC layer, but still receive IP packets destined for a remote system attached to a non-broadcast GRF interface and route those packets at the IP layer.

On the FDDI card, frame forwarding is compatible with any station sending and receiving FDDI LLC frames.

IPv4 frames are fragmented as necessary, as when bridging an FDDI frame of more than 1500 bytes to an Ethernet interface. The GRF bridge will attempt to break such a frame into fragments that will fit the sending interface. This is possible if the frame contains an IP datagram; then the GRF may use the fragmentation rules of IP to split the frame. Otherwise, the GRF must drop the frame.

Refer to the *Transparent Bridging* chapter for more information.

Proxy ARP

Proxy ARP is supported on GRF broadcast media, FDDI and Ethernet cards.

Proxy ARP enables a router to answer an ARP request on one of its networks that is actually destined for a host on another of the router’s networks. This leads the sender of the ARP request into thinking that the router is the destination host, when in fact the destination host is "on the other side" of the router. The router acts as a proxy agent for the destination host, relaying packets to it from the other hosts.

Controlled-load (class filtering)

Controlled-Load is supported on the FDDI media card.

The GRF delivers Controlled-Load service to a specific flow by marking its packets precedence field to prevent Selective Packet Discard (SPD). The marking mechanism uses filters to identify the packets belonging to the class of applications for which resources are reserved. Class filters are manually configured by adding them to `/etc/filterd.conf`.

Controlled-Load protects packets that match the filter from being lost. Packets that match the filter are marked so they will not be dropped by SPD. SPD drops packets that are not marked when the number of free buffers gets too low. Dynamic routing packet precedence fields are marked by GateD. The class filter is another way of setting the same precedence bit in the IP packet header.

Refer to the *Integrated Services: Controlled-Load* chapter for information about constructing class filters.

Selective packet discard

Selective packet discard can be enabled on the FDDI card to ensure that dynamic routing packets are transmitted on the media in the presence of a sustained high volume of data packets. During high traffic volumes, data packets are discarded in a rate that favors dynamic routing packets.

Packet discard is regulated by reserving buffers for dynamic routing packets. This gives the operator complete control over the point at which congestion management begins to discard data packets. A user-configured threshold defines the percentage of buffers to reserve for dynamic routing packets.

When the threshold is set to zero, no buffers are reserved for dynamic routing packets and dynamic routing packet discard is disabled. In this case, dynamic routing packets and data packets are treated identically.

When the threshold is set to 100, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the threshold of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the number of free transmit buffers is less than the user-defined threshold of buffers required to be reserved for dynamic routing packets. When the number of free buffers used for switch receive/media transmit falls below the congestion threshold, non-dynamic routing packets are discarded until the congestion condition clears. Because the congestion condition is updated thousands of times per second and busy buffers are rapidly transmitted and returned as free buffers, a congested state ends rapidly after its onset. This prevents prolonged discard of non-dynamic routing packets and ensures the transmission of dynamic routing packets even during periods of heavy network load.

The discard mechanism applies only to the transmit side of the media card, and has no impact on packets received from the media. There is no analogous treatment of packets received from the media. The discard threshold is set to zero by default, and is therefore disabled by default.

The threshold value is unique per media card in the chassis, and is set at the Card profile in the CLI. Ascend recommends the threshold value be set low, to a small value that maximizes the benefit for dynamic routing packets and minimizes the impact on data packets. As the number reserved for dynamic routing packets increases, the number of buffers available for data traffic decreases and dynamic routing packets are a small percentage of all packets when the card is congested, Practice has shown it unnecessary to set the threshold above single digits as it is unlikely that dynamic routing packets account for more than a few percent of all packets.

Checking results

Examine GateD log files to determine the number of dynamic routing packets transmitted and their timestamps. A little arithmetic using the timestamps in the log files for packets transmitted to a neighbor (remember this is a transmit-only feature) should indicate the number of dynamic routing updates per unit time. Compare this number to the cumulative packet counters for switch receive over the same unit of time and you should arrive at the percentage of all transmit packets that are dynamic routing packets. Compare the average number over a few minutes to the number in a worst-case condition during bursts of dynamic routing packets based on periodic updates, and then select a percentage that balances the two.

IS-IS protocol support

IS-IS is a link state interior gateway protocol (IGP) originally developed for routing ISO/CLNP (International Organization for Standardization/Connectionless Network Protocol) packets. In ISO terminology, a router is referred to as an “intermediate system” (IS). IS-IS intra-domain routing is organized hierarchically so that a large domain may be administratively divided into smaller areas using level 1 intermediate systems within areas and level 2 intermediate systems between areas.

This example shows FDDI interface `gf030` configured for IS-IS in the GateD IS-IS statement:

```
isis yes {
      area "49000080";
      systemid "326032603260";
      interface "gf030" metric 10 priority 60;
} ;
```

An ISO address must also be assigned to the FDDI logical interface in `/etc/grifconfig.conf`. This is in addition to the entry for the IP address also assigned in that file. Refer to the *Introduction to IS-IS* chapter for more information.

Here is an example of FDDI entries in `/etc/grifconfig.conf`:

```
#name   address         netwmask       broad_dest arguments
gf030  xxx.xxx.xxx.xxx 255.255.255.0        -     mtu 4100
#interface_name <iso_address> <iso_area> - iso
gf030 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

How FDDI interfaces are named

An FDDI media card has four physical interfaces.

Each interface is named/numbered in four different ways:

– by its physical location on the FDDI card

– by a site-specified SAS–DAS setting name in the Card profile, `single` or `dual`

– by a logical interface number assigned after the SAS/DAS settings are numbered (use in `grifconfig.conf` file)

– by a unique IP address assigned to each logical interface

Figure 6-6 indicates files where various numbers are used to configure the interfaces on a FDDI media card:

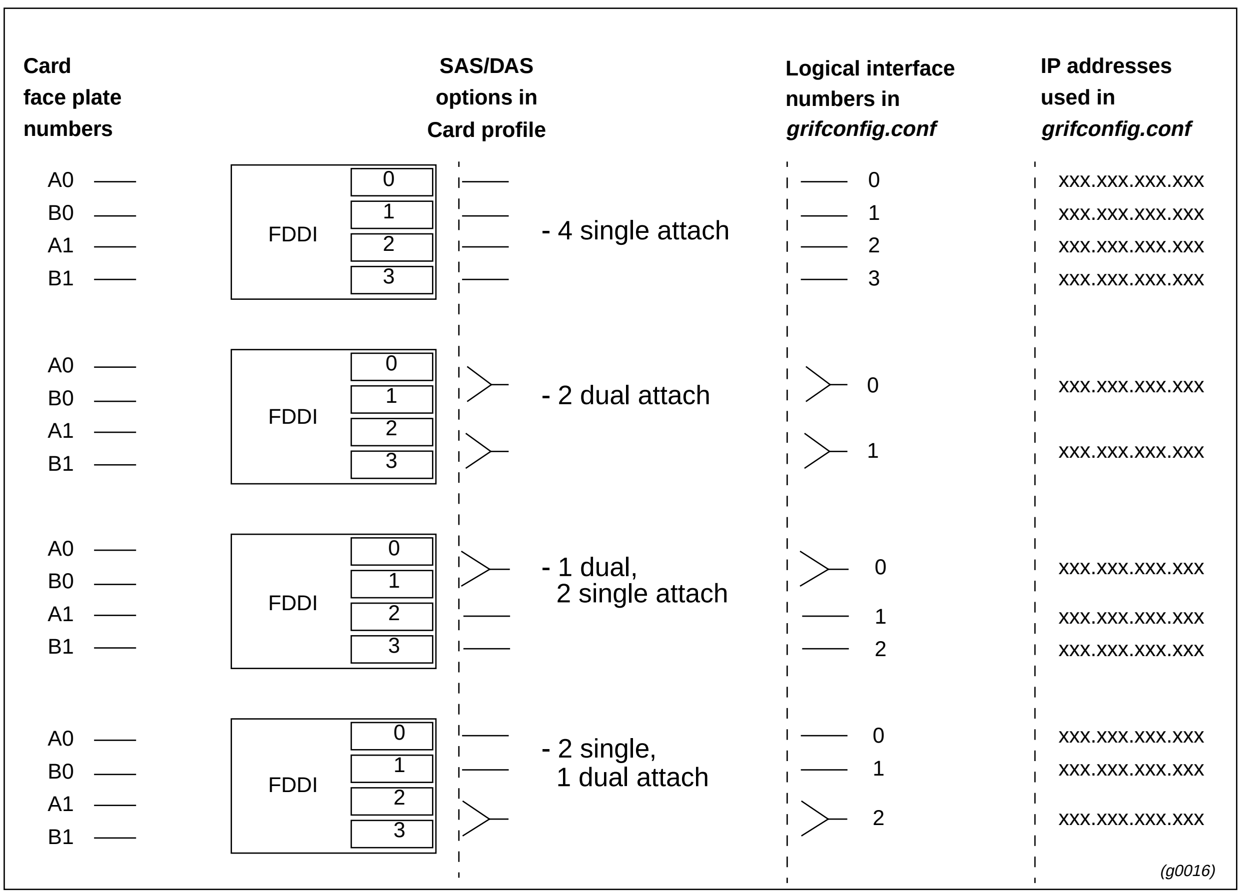


Figure 6-6. Assigning numbers to FDDI interfaces

Physical interface numbers

The physical interface number identifies the specific FDDI fiber optic attachment component according to its location on the media card, 0–3.

Starting at the top of the media card, each physical interface is numbered consecutively, beginning with 0, as shown in Figure 6-7:

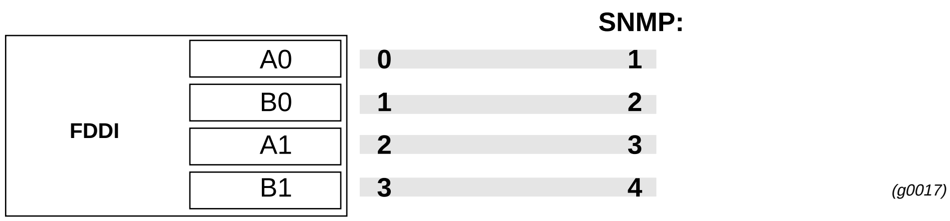


Figure 6-7. Physical interface numbering on FDDI media card

The diagram shows that the physical interface numbering is 0-based, 0–3, and that SNMP numbering is 1-based, 1–4.

For FDDI cards, the fifth character will be 0, 1, 2, or 3 to specify the logical interface on the FDDI card.

Note: The logical interface number may be different from the physical interface on the card, depending on whether the interface is single- or dual-attached. For example, gf020 can specify the top-most connector on the FDDI card in slot 2, or the top two connectors on that card if they are configured dual-attached.

Looking at the FDDI card

The FDDI media card has four physical interfaces. Each interface has a pair of LEDs that show the type of connection (OP) and traffic activity (TRX) at that interface. Refer to Figure 6-8.

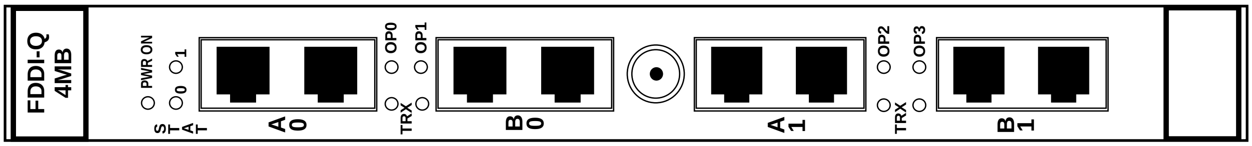


Figure 6-8. FDDI/Q media card faceplate and LEDs

LEDs on the faceplate

Certain LEDs on the FDDI/Q media card can be either amber or green depending upon the type of information they convey at the time. Table 6-1 describes FDDI/Q card LEDs.

Table 6-1. FDDI/Q media card LEDs

LED	Description
PWR ON	This green LED is on when GRF power is on.
STAT 0 STAT 1	The amber / green Status LEDs at the top of each FDDI/Q media card are amber during self-test. When self test completes, the LEDs turn green. The Status LEDs alternate amber during power-on/dumping, and alternate green during power-on/loading. When status is normal: - the green 0 LED on the left blinks ten times a second - the green 1 LED on the right blinks once a second FDDI/Q Status LEDs do not blink error codes.
OP0, OP1, OP2, OP3	The amber / green OP LEDs indicate the type of ring connection made at the particular interface: When OP is off, no viable connection is enabled. When OP is green, a SAS connection is configured. When OP is amber, a DAS connection is configured.
TRX	These green LEDs blink when FDDI/Q traffic is active in either direction at a particular interface (updated each 100 ms).

List of FDDI configuration steps

These are the steps to configure FDDI cards:

1 Assign IP address to each logical interface

Edit `grifconfig.conf` to assign an IP address for each logical interface.

2 Specify FDDI card parameters in the Card profile:

– specify SAS and DAS settings as single or dual

– enable optical bypass on or off

– OPTIONAL: specify ICMP throttling settings

– OPTIONAL: specify selective packet discard threshold

– OPTIONAL: change run-time binaries

– OPTIONAL: change dump variables

These next steps are optional, they describe tasks that are performed infrequently:

3 Change Load profile (optional).

Global executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every FDDI card.

If you want to change the run-time code in one FDDI card (per physical interface), make the change in the Card profile, in the `load` field.

4 Change Dump profile (optional).

Global dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.

If you want to change dump settings for one FDDI card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1. To save files in the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

2. In the command-line interface, use **set** and **write** commands to save a profile.

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configuring a FDDI interface

This section describes how to configure a FDDI interface in the `/etc/grifconfig.conf` file. Use a UNIX editor to make entries in `/etc/grifconfig.conf`.

Each logical FDDI interface is identified in `/etc/grifconfig.conf` as to its:

- interface name, `gf0yz` (names are always lower case)
- Internet address
- netmask
- broadcast/destination address (optional)
- arguments field (optional)

The format for an entry in the `grifconfig.conf` file is:

```
name   address    netmask       broad_dest    arguments
```

Interface name *gf0yz*
Each logical GRF interface is given an interface name `ga0yz` where:

- the “gf” prefix indicates a FDDI interface
- the chassis number is always “0”
- “y” is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
- “z” is the logical interface number in hex, on the FDDI card it is 0, 1,2, and 3.

Note: The logical interface number may be different from the physical interface on the card, depending on whether the interface is single- or dual-attached. For example, gf020 can specify the top-most connector on the FDDI card in slot 2, or the top two connectors on that card if they are configured dual-attached.

Address
Enter the IP or ISO address to be assigned to this interface.

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured.

Broadcast address
Use the broadcast address when you wish to specify other than all 1s as the broadcast address.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO addess is being added to an interface’s IP address. Specify the MTU value as `mtu` *xyz*. Leave the `arguments` field blank if you are not using it.

Example

The entry assigns an IP address for interface 3 on the FDDI card in slot 6. If needed, a dash is used as a placeholder for the broadcast address:

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
gf063 192.0.2.1 255.255.255.0 192.0.2.255
```

If an interface is nonbroadcast (NBMA), do not include a destination address in its `/etc/grifconfig.conf` entry.

Save the /etc file

Save the file with the editor. Then, use **grwrite** to write the file to the `/etc` configuration directory:

```
# grwrite -v
```

Check contents of grifconfig.conf file

After you save the `/etc` directory and reset the FDDI card, use **netstat -in** to display the contents of the `/etc/grifconfig.conf` file and verify that the logical interface is configured with the correct IP address.

Here is the output from a **netstat** command looking at the FDDI interfaces:

```
# netstat -in | grep gf
gf081   4352  <link23>    00:c0:80:06:19:68    4    0     10    0    0
gf081   4352  203.3.10    203.3.10.156         4    0     10    0    0
```

Please refer to the **netstat** man page for information about other **netstat** options.

Check system-level IP configuration

The UNIX **ifconfig *interface*** command returns system level information for the specified interface name, here is the interface for logical interface 0 (`ga020`):

```
# ifconfig gf081
gf081: gritfddi flags=180b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
              MULTICAST> mtu 4352
        inet 203.3.10.156 netmask 0xffffff00 broadcast 203.3.10.255
```

Setting parameters in the Card profile

This section describes how to verify and/or change FDDI parameters in the Card profile. The parameters are presented in this order:

- Set FDDI SAS / DAS parameters per port
- Enable optical bypass on, per port, default is off
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard threshold
- OPTIONAL: set card-specific load variables
- OPTIONAL: set card-specific dump variables

1. Set FDDI parameters

Media card type, `fddi-v2`, is automatically read into the read-only `media-type` field. Other values shown are defaults.

```
super> read card 6
CARD/6 read
super> list
card-num* = 6
media-type = fddi-v2
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = <{ 0{off on 10 3} {single off} {"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

The FDDI parameters are located in the `ports 0` through `ports 3` sections of the top-level Card profile.

```
super> list ports 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 207 }
hssi = { 1 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disab+

super> list fddi
single-dual = single
optical-bypass = off
super>
```

```
super> set single-dual = dual
super> set optical-bypass = on
super> write
CARD/6 written
```

Check the changes you have made and saved. If you are at the `ports` level, use **cd ..** to go “up” a level so you can access the FDDI section:

```
super> cd ..

super> list fddi
single-dual = dual
optical-bypass = on
super>
```

Now do the settings for the three other FDDI ports. Use **cd ..** to move up to the ports level as you need.

Tip: A quick way to set interface 1 in slot 8 as DAS without moving “down” into the profile:

```
super> read card 6
CARD/6 read
super> set port 1 fddi single-dual = dual
super> write
CARD/6 written
```

2. Specify ICMP throttling

You can specify ICMP throttling changes for this FDDI card in these settings. Refer to Chapter 1 for an explanation of each field or do a `set <field-name>?` command for a brief description. Default values are shown:

```
super> read card 6
CARD/6 read
super> list card 6
card-num* = 6
media-type = ether-v2
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < { 0 {off on 10 3} {single off}{"" "" 1 sonet internal-oscillat+
load = { 0 < > 1 3 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

Here is how to access the help information for the `echo-reply` field:

```
super> set echo ?
echo-reply:
```

```
 The number of ICMP ping responses generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

Changeand save the default echo reply and TTL settings with these commands:

```
super> set echo-reply = 8
super> set TTL-timeout = 12
super> write
CARD/6 written
```

You do not have to do a **write** until you have finished all changes in the Card profile. You get a warning message if you try to exit a profile without saving your changes.

3. Specify selective packet discard threshold

Specify a SPD threshold for this FDDI card in the `spd-tx-thresh` field. This field is contained in the `config` section of the Card profile. A discussion of how to determine an SPD threshold is provided in the “Selective packet discard” section at the beginning of this chapter.

```
super> read card 6
CARD/6 read
super> list
card-num* = 6
media-type = ether-v2
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = Frame-Relay
ether-verbose = 0
ports = < {0{ off on 10 3} {single off}{"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 8 10 2147483647 12 10 10 }

super> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0
super> set spd-tx-thresh = 7
super> write
CARD/6 written
```

On reboot, the congestion threshold message should indicate the new setting, as shown below:

```
[2] [TX] Current congestion thresholds, out of 256 available buffers:
[2] [TX] Congestion:  17 (7%) [2] [TX]        Overshoot:  8
```

A discussion of how to determine an SPD threshold is provided in the “Selective packet discard” section on page 6-7. The FDDI **maint 4** command reports discard counts.

4. Specify different executables

Card-specific executables can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this FDDI card only.

```
super> read card 6
card/6 read
super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you want to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_executable_for_fddiq
super> write
CARD/6 written
```

5. Specify different dump settings

Card-specific dump file names can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 6
card/6 read
super> list dump
config = 0
hw-table = < >
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex. Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD/6 written
```

Optional: change FDDI binaries – Load profile

Global values for executable binaries are set in the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** FDDI cards. Here is the path:

```
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the FDDI card settings:

```
super> list fddi-v2
type = fddi-v2
rx-config = 0
rx-path = /usr/libexec/portcards/fddiq-0.run
tx-config = 0
tx-path = /usr/libexec/portcards/fddiq-1.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

As an example, to execute different run-time code on the receive side of the FDDI card, replace `/usr/libexec/portcards/fddiq-0.run` with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/newfddiq_rx.run
super> write
LOAD written
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. You have the option to configure the card's boot sequence so that after booting, the card loads and runs diagnostics before it loads and runs the executable binaries. Set the `enable-boot-seq` field to `on` and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the "Management Commands and Tools" chapter in this manual for information.

Optional: change FDDI dumps– Dump profile

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Defaults are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read
super> list
hw-table = <{hippi 20 /var/portcards/grdump 0} {rmb 20 /var/portc+
dump-vector-table = <{ 3 rmb "RMB default dump vectors" < { 1 SRAM 26214+
config-spontaneous = off
keep-count = 2
```

The `hw-table` field has settings to specify when dumps are taken and where dumps are stored. Here is the path to examine the FDDI settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
etherne+ = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list fddi-v2
media = fddi-v2
config = 20
path = /var/portcards/grdump
vector-index = 6
```

In the `config =` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP/ written
```

Dump vectors

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for all FDDI cards.

Here is the path, first you **cd** .. up to the main level:

```
super> cd ..
super> list . d
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" <{1 "atm inst memory" 167+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" < {1 "fddi/Q CPU0 core memo+
7 = {7 hssi "HSSI default dump vectors" <{1 "hssi rx SRAM memory" 20971+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" < {1 "Ethernet rx SRA+
9 = {9 dev1 "DEV1 default dump vectors" <{1 "dev1 rx SRAM memory" 209715+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" < {1 "ATM-12 SDRAM+
11 = {11 sonet-v1 "SONET default dump vectors" <{1 "SONET rx SRAM memory+
```

The segment tables contain the start and stop address for each area of memory dumped, this changes for each type of media card:

```
super> list 6
index = 6
hw-type = fddi-v2
description = "FDDI/Q default dump vectors"
segment-table = < {1 "fddi/Q CPU0 core memory" 2097152 2097152}{2 "fddi/+
```

This sequence shows one segment of the areas in the FDDI card that are dumped, note that segment table is abbreviated to “s”:

```
super> list s
1 = { 1 "fddi/Q CPU0 core memory" 2097152 2097152 }
2 = { 2 "fddi/Q IPC memory" 8388608 32800 }
3 = { 3 "fddi/Q shared descriptor memory" 16777216 131072 }
4 = { 4 "fddi/Q CPU1 core memory" 4194304 2097152 }

super> list 1
index = 1
description = "fddi/Q CPU0 core memory"
start = 2097152
length = 2097152
```

Monitoring FDDI media cards

Use the **maint** commands to look at packet statistics on the FDDI media card. The **maint** commands operate from the control board and require the GR> prompt. Execute the **grrmb** command to switch prompts.

If you are not sure of the card's slot number, use the **grcard** command to view the location of installed cards.

Canonical output

By default, FDDI reports its MAC addresses in canonical form:

– output at initiation time of GRF FDDI ports is canonical

– output of neighbors is canonical

Using maint commands

The FDDI **maint** command displays a range of information about the media card, including:

– **maint** command options

– version numbers of media card and kernel software

– media card configuration and status

– FDDI media statistics

– switch statistics

– ARP entries

– history trace

– bridging information

Several management operations are performed with **maint**:

– clear statistics counters

– examine SMT MIB variables

– set history trace on/off

– reset interface

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

grrmb

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the FDDI media card you are working with. For example, if you are working with a card in slot 2, enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 2
GR 2>
```

To leave the **maint** prompt, enter **`quit`**.

Receive / transmit side maint commands

Use **maint 1** to see the list of **maint** commands for the receive side, use **maint 70** to see the list for the transmit side.

Receive side list

```
GR 03> maint 1
[CPU0]  1:     Display this screen of Options
[CPU0]  2:     Display Version Numbers
[CPU0]  3:     Display Configuration and Status
[CPU0]  4:     Display FDDI media Statistics
[CPU0]  5:     Display SWITCH Statistics
[CPU0]  6:     Display Combus Statistics
[CPU0]  7:     Clear statistics counters (may mess up SNMP)
[CPU0]  9:     History trace on/off [ 0 | 1]
[CPU0]  10:    Display History Trace
[CPU0]  11:    Examine SMT mib variables [ ID, smt, index]
[CPU0]  12:    Reset Interface [ IF ]
[CPU0]  13:    Set/Show CAM addresses [ IF [ num entry ] ]
[CPU0]  14:    Copy All Multicast LLC Frames [ IF [ 1/0 ]]
[CPU0]  15:    Read MAC_CNTRL_A IF
[CPU0]  16:    Display Multicast Routing Table
[CPU0]  17:    Toggle promiscuous mode per interface [ IF [1/0] ]
[CPU0]  18:    Toggle promiscuous mode all interfaces [1/0]
[CPU0]  30:    Switch Test: Clear Stats (but not setup)
[CPU0]  31:    Switch Test: IP [ patt len slots...]
[CPU0]  32:    Switch Test: Setup [ patt len slots... ]
[CPU0]  33:    Switch Test: Start [ slots...]
[CPU0]  34:    Switch Test: Stop [ slots...]
[CPU0]  35:    Switch Test: Status [ slots...]
[CPU0]  36:    Switch Test: Duration [ slots...]
[CPU0]  37:    Switch Test: Packets-per-second [ slots...]
[CPU0]  38:    Switch Test: Send Single Packet [ slots...]
[CPU0]  39:    Switch Test: Send Single IP Packet [ slots...]
[CPU0]  45:    List next hop data: [family]
[CPU0]  50:    Filtering filter list: [detail_level [ID]]
[CPU0]  51:    Filtering filter list: [detail_level [IF]]
[CPU0]  52:    Filtering action list: [detail_level [ID]]
[CPU0]  53:    Filtering action list: [detail_level [IF]]
[CPU0]  54:    Filtering binding list: [detail_level [ID]]
[CPU0]  55:    Filtering binding list: [detail_level [IF]]
[CPU0]  56:    Display filtering statistics: [IF#]
[CPU0]  57:    Reset filtering statistics: [IF#]
[CPU0]  58:    Show filter protocol statistics
```

```
[CPU0]        note, IF/ID may be '-1' to indicate all of the given item
[CPU0]              while detail level is 0|1|2.
[CPU0]  60:    Print FSI registers and counters.
[CPU0]  61:    Print FSI indirect registers.
[CPU0]  62:    Print PHY registers and counters.
[CPU0]  63:    Print MAC registers and counters.
[CPU0]  64:    Display memory [address #words]
[CPU0]  70:    Send maint command to cpu-1.
```

Transmit side list

CPU1 maintains data for certain functions such as ARP. Enter the **maint 70** command to switch to the set of CPU1 commands.

```
GR 03> maint 70

[CPU1]   1:     Display this screen of maint command options.
[CPU1]   8:     Display ARP entries
[CPU1]   9:     History trace on/off [ 0 | 1]
[CPU1]  10:     Display History Trace
[CPU1]  43:    List next hop data: [family]
[CPU1]  50:    Filtering filter list: [detail_level [ID]]
[CPU1]  51:    Filtering filter list: [detail_level [IF]]
[CPU1]  52:    Filtering action list: [detail_level [ID]]
[CPU1]  53:    Filtering action list: [detail_level [IF]]
[CPU1]  54:    Filtering binding list: [detail_level [ID]]
[CPU1]  55:    Filtering binding list: [detail_level [IF]]
[CPU1]  56:    Display filtering statistics: [IF#]
[CPU1]  57:    Reset filtering statistics: [IF#]
[CPU1]  58:    Show filter protocol statistics
[CPU1]         note, IF/ID may be '-1' to indicate all of the given item
[CPU1]               while detail level is 0|1|2.
[CPU1]  60:     Display memory address #words
[CPU1]  70:     Display cache tags
```

A few examples follow, and show how the CPU of origin is included in the data.

Display port card S/W version

In actual released code, the dates will be different, and some of the version numbers may be different.

Enter: `maint 2`

```
[CPU0]             FDDI Port Card Hardware and Software Revisions:
[CPU0]             ===============================================
[CPU0] HW:
[CPU0]  Power-On Self-Test (POST) result code: 0x0.
[CPU0]  FDDI Media Board HW Rev: 0x7, with 2M Sram.
[CPU0]  FDDI Xilinx Version: 0x0.
[CPU0]  SDC Board HW Rev: 0x9  (SDC2).
[CPU0]          SDC2 Combus Xilinx version: 0x6.
[CPU0]          SDC2 Switch Transmit Xilinx version: 0x5.
[CPU0]          SDC2 Switch Receive Xilinx version: 0x0.
```

```
[CPU0] SW:
[CPU0] FDDI Code Version: A1_4_12R_3, Compiled Wed Sep 23 03:13:46 CDT
[CPU0]      1998, in directory: /nit/A1_4_12R_3/fddiq/common.
[CPU0] IPv4 Library Version: 1.4.12R.3, Compiled on Wed Sep 23 03:08:06
       CDT 1998
[CPU0]  Route Library Version:  1.4.12R.3, Compiled on Wed Sep 23
        03:05:52 CDT 1998.
[CPU0]  IF Library Version: 1.1.0.0, Compiled on Wed Sep 23 03:08:48
         CDT 1998.
```

Verify FDDI configuration

The **maint 3** command returns configuration information for each interface:

```
GRF> 13> maint 3
[CPU0]      FDDI Slot 13 Configuration and Status Summary:
[CPU0]          Interface 0     Interface 1     Interface 2     Interface 3
[CPU0]          ===========     ===========     ===========     ===========
[CPU0] Single/Dual:  Single       Single        Single          Single
[CPU0] Port(s):    0              1               2               3
[CPU0] Station Cfg:   Wrap S       Isolated      Isolated       Isolated
[CPU0] Primary MAC:
[CPU0] Mac Address: 00c080061967  00c080061968  00c080061969  00c08006196a
[CPU0] Upstr Nbor:  00030191944b  0003019194eb  000000000000  00030191d42d
[CPU0] Dnstr Nbor:  00030191944b  0003019194eb  000000000000  00030191d42d
[CPU0] Secondary MAC:
[CPU0] Mac Address:
[CPU0] Upstr Nbor:
[CPU0] Dnstr Nbor:
[CPU0] IP Address:   203.3.14.156 16.128.0.6  203.1.10.156 203.5.2.156
```

List statistics per FDDI interface

Enter: `maint 4`

```
[CPU0]               FDDI statistics
[CPU0] Input:
[CPU0]   IF      Bytes      Packets    Discards    Errors
-------------------------------------------------------------------
[CPU0]   0     000777416   000005549   000000755   0000
[CPU0]   1     000000000   000000000   000000000   0000
[CPU0]   2     000000000   000000000   000000000   0000
[CPU0]   3     1188416834   004256673   000000000   0000
[CPU0] Output:
[CPU0]   IF      Bytes      Packets    Discards    Errors
[CPU0]
-------------------------------------------------------------------
[CPU0]   0     874971296   092320332   000000000   0000
[CPU0]   1     000000000   000000000   000000000   0000
[CPU0]   2     000000000   000000000   000000000   0000
[CPU0]   3     934086079   004145604   000000000   0000
```

List switch interface statistics

Enter: `maint 5`

```
GR 13> maint 5
[CPU0]                    Switch Statistics
[CPU0] input:
[CPU0]         Bytes              Packets          Errors
[CPU0] ---------------------------------------------------------
[CPU0]  000000020535388184    000000000096464900    000000000
[CPU0]
[CPU0] output:
[CPU0]         Bytes              Packets          Errors
[CPU0] ---------------------------------------------------------
[CPU0]  000000014131302659    000000000004237665    000000000
```

List communications bus interface statistics and status

Enter: `maint 6`

```
GR 13> maint 6
[CPU0]
[CPU0] Combus Status:
[CPU0]  Last interrupt status:           0x0
[CPU0] Combus Statistics:
[CPU0]  Message ready interrupts:          1471959
[CPU0]  Truncated input messages:              431
[CPU0]  Grit messages for TX-CPU:                0
[CPU0]  Ip messages Rcvd (non-bypass):           0
[CPU0]  Raw messages:                            0
[CPU0]  ISO messages:                            0
[CPU0]  Grid messages:                     1471959
[CPU0]  Grid echo requests:                  43691
[CPU0]  Port available messages:                 0
[CPU0]  Segmented Packets:                       0
[CPU0]  Segments Sent:                           0
[CPU0] Combus Errors:
[CPU0]  Bus in timeouts:         0      Bus out timeouts:         0
[CPU0]  Out of buffer cond.:     0      Bad packet type:          0
[CPU0]  Dropped IP packets:      0      Bad packet dest:          0
[CPU0]  Receive Msg Errors:      0      Receive Format Errors:    0
[CPU0]  Receive Past End:      431      Received Long Message:  120
```

Clear all statistics

Enter: `maint 7`

```
GR 13> maint 7
All FDDI Statistics Cleared.
All Switch Statistics Cleared.
All Combus Statistics Cleared.
```

Display current ARP table contents

Enter: `maint 70 8`

```
[CPU1]  Arp Table:
[CPU1]  I/F IP Address       Mac Address          Status    TTL
[CPU1]  === ============     =================    ======    ===
[CPU1]  0  203.3.14.200     00:01:02:03:04:05     80000007   600
[CPU1]  1  203.3.10.158     00:c0:80:89:29:d7     03         559
```

Display SMT MIB variables

Enter: `maint 11`

```
GR 13> maint 11 0x2018 3 0
[CPU0] SMT MIB Request made successfully.  Result in grconslog.
[CPU0]
[CPU0] Maint SMT Mib Response:
[CPU0]     Parameter ID: 0x2018, SMT Index: 3, Resource Index: 0
[CPU0]     Value:
[CPU0] 00000001 1B084036
```

Reset individual FDDI interface

Use **maint 12** and the physical interface number (0–3) to reset an individual interface.

Display CAM addresses

Content addressable memory (CAM) contains the information to support reception of multicast datagrams.

Enter: `maint 13`

```
GR 13> [CPU0]     CAM addresses
[CPU0]
[CPU0] Interface 0: +Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0 01:00:5e:00:00:01
[CPU0] Interface 1: -Multicast -Promiscuous -BridgeStrip
[CPU0] Interface 2: +Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0 01:00:5e:00:00:01
[CPU0] Interface 3: +Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0:d:1 flags=0xa042<BROADCAST,RUNNING,LINK1,MULTICAST> mtu 4352
[CPU0] 0:d:2 flags=0xa043<UP,BROADCAST,RUNNING,LINK1,MULTICAST> mtu 4352
[CPU0] 0:d:3 flags=0xa043<UP,BROADCAST,RUNNING,LINK1,MULTICAST> mtu 4352
```

As an option, you can specify the particular physical interface:

```
GR 13> maint 13 gf0d2
```

```
[CPU0]     CAM addresses
[CPU0]
[CPU0] Interface 0: +Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0:d:0 flags=0xb043<UP,BROADCAST,RUNNING,LINK0,LINK1,MULTICAST>
mtu 4352
```

Toggle promiscuous mode

You can toggle promiscuous mode on/off (1/0) for all (**maint 18**) or individual interfaces (**maint 17**):

```
GR 13> maint 18 0
[CPU0] Interface 0: -Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0 8000
[CPU0] Interface 1: -Multicast -Promiscuous -BridgeStrip
[CPU0] 1 0000
[CPU0] Interface 2: -Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 2 0000
[CPU0] Interface 3: -Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 3 0000
[CPU0] All interfaces' promiscuity set to 0

maint 18 1
GR 13> [CPU0]
[CPU0] Interface 0: +Multicast +Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 0 8030
[CPU0] Interface 1: +Multicast +Promiscuous -BridgeStrip
[CPU0] 1 0030
[CPU0] Interface 2: +Multicast +Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 2 0030
[CPU0] Interface 3: +Multicast +Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
[CPU0] 3 0030
[CPU0] All interfaces' promiscuity set to 1
```

Print PHY registers/counters

This command displays internal state information from FDDI PHY hardware. Enter: `maint 62` *`interface`*

```
GR 13> maint 62 0
[CPU0] Dumping PHY registers 0:
[CPU0] ELM(0) regs:
[CPU0]      ELM_CNTRL_A: 0000     ELM_CNTRL_B: 8058       INTR_MASK: 207C
[CPU0]  XMIT_VECTOR: 0001   VECTOR_LENGTH: 0000    LE_THRESHOLD: 00FF
```

```
[CPU0]           A_MAX: FF65         LS_MAX: FFCF         TB_MIN: FF10
[CPU0]           T_OUT: ECED       LC_SHORT: F676        T_SCRUB: FFFF
[CPU0]          NS_MAX: E796  TPC_LOAD_VALUE: 0000  TNE_LOAD_VALUE: 0000
[CPU0]     ELM_STATUS_A: 4B61    ELM_STATUS_B: 7421            TPC: 0000
[CPU0]             TNE: E796        CLK_DIV: 0398  BIST_SIGNATURE: 0000
[CPU0]      RCV_VECTOR: 0001     INTR_EVENT: 1002    VIOL_SYM_CTR: 000D
[CPU0]     MIN_IDLE_CTR: 004B    LINK_ERR_CTR: 0000
[CPU0] PHY info 0
[CPU0] PHY(0) info:
[CPU0] INTR_EVENT: 0044       lct: 00000    pcmbrk: 00000
[CPU0] pcmcode: 00011   pcmena: 00001    tracep: 00000   selftst: 00000
```

Print MAC registers/counters

This command displays internal state information from FDDI MAC hardware. Enter: `maint 63 interface`

```
GR 13> maint 63 0
Dumping MAC registers 0:
[CPU0]  MAC_CNTRL_A: 8010      MAC_CNTRL_B: 0000    INTR_MASK_A: 2344
[CPU0] INTR_MASK_B: C000       INTR_MASK_C: 0000            MSA: A5A5
[CPU0]       MLA_A: 0334             MLA_B: 8000          MLA_C: 00C0
[CPU0]       T_REQ: E119  VX_VALUE_N_T_MAX: 6DE0   INTR_EVENT_C: 0005
[CPU0]   VOID_TIME: 0039          TOKEN_CT: EAC2       FRAME_CT: 001A
 LOST_CT_N_ERROR_CT: 0000     INTR_EVENT_A: 4000   INTR_EVENT_B: 0010
[CPU0]    RX_STATUS: 0020        TX_STATUS: 0830        T_NEG_A: 7A00
[CPU0]      T_NEG_B: 00FE       INFO_REG_A: 7A00     INFO_REG_B: 7777
     BIST_SIGNATURE: 0000         TVX_TIMER: 6D27     TRT_TIMER_A: 7A21
[CPU0]   TRT_TIMER_B: 00FE   THT_TIMER_A: 7A95STCNT_N_THT_TIMER_B:00F
[CPU0] PKT_REQUEST: 67C7          RC_CRC_A: D2D6       RC_CRC_B: 12ED
[CPU0]     TX_CRC_A: FFFF        TX_CRC_B: FFFF

[CPU0] MAC info 0
[CPU0] MAC(0) info data:
[CPU0] mla: 00 c0 80 89 29 06  frame_ct: 00516   lost_ct: 00000
error_ct: 00000
[CPU0]     ringop: 00001    tvxtmr: 00000    latect: 00000
[CPU0]     recvry: 00000  utknrcvd: 00000  rtknrcvd: 00000
[CPU0]   mybeacon: 00000otherbeacon: 00000higherclaim: 00000
[CPU0] lowerclaim: 00000   myclaim: 00000   badtbid: 00000
```

Collect data via grdinfo

With a single command, **grdinfo** collects the output from nearly all of the FDDI **maint** commands and compresses it in a log file. Refer to the “Management Commands and Tools” chapter in this manual for more information.

Use grstat to look at layer 3 statistics

```
# grstat ip gf08
card 8 (4 interfaces found)
  ipstat totals
       count description
    12411564 total packets received
    12411564 packets forwarded normally
  ipdrop totals
       count description
```

The **grstat** layer 2 statistics are not available for FDDI cards.

Use grstat grid to look at card-control board traffic

GRID does internal command messaging. These communications include functions such as route updates, and interface adds and deletes. In **grstat**, the **grid** option returns statistics for messages needed to monitor internal traffic on a card.

```
# grstat grid 8
card 8
  GRID statistics
               count description
                3346 COMBUS messages received
                3346 COMBUS GRID messages received
                2871 COMBUS GRID echo requests
                   9 COMBUS receive long messages
                 424 GRIDAX packets received
                  21 GRIDAX restart
                  72 GRIDAX acks received
                  14 GRIDAX requested acks received
                  86 GRIDAX control packets received
                   9 GRIDAX dropped out of order
                 329 GRIDAX output queued
                 408 GRIDAX packets sent
                  79 GRIDAX acks sent
                  79 GRIDAX control packets sent
```

HIPPI Configuration

7

Chapter 7 describes the implementation of HIPPI on the GRF and other features supported on the HIPPI media card. It includes the information needed to configure HIPPI interfaces and parameters in the `/etc/grifconfig.conf` file and CLI profiles. The first sections are a brief HIPPI tutorial.

Chapter 7 covers these topics:

Introduction to HIPPI

HIPPI poses interesting configuration problems because of the number of ways HIPPI connections can be established. Several addressing schemes can be used depending upon how a site needs to organize and connect equipment to support a range of user needs.

Not only are there several addressing schemes, but a HIPPI media card can be configured to process all of them. The HIPPI media card is capable of handling both HIPPI-SC switching protocol and IP packet routing and, based on I-field indicators, can dynamically alternate between these modes. I-fields on HIPPI host machines are discussed frequently in this section. Hosts must pass on the appropriate information for GRF media cards and other HIPPI devices to operate in the ways you intend.

HIPPI offers many configuration options. The ANSI HIPPI standards and RFCs describe implementation details that support source routing, logical addressing, IP routing, and raw (switch) mode operations.

Note: You can obtain ANSI standards and RFCs via anonymous ftp at the site: `ftp.isi.edu`

Files are in the `/in-notes` directory, with file names of the form `rfcnnnn.txt`. The file `rfc-index.txt` is an index of all RFCs.

Connection processing

The GRF processes connections; it does not process data. It accepts data and establishes a connection point to which it can transfer data.

The HIPPI media card establishes connections and transfers packets. A HIPPI media card processes one HIPPI connection at a time. It does not begin another process until the first connection completes.

There are two types of connections: an IP connection and a raw connection. In internetworking, the main difference between the two connections is that data from a HIPPI host can be transferred to any other IP-capable media via IP routing. Raw mode is HIPPI-to-HIPPI and is only used to transfer data from one HIPPI device to another HIPPI device. The HIPPI I-field tells the media card which type of connection is being requested.

Starting a HIPPI connection

The HIPPI standard requires that a HIPPI connection be established between a HIPPI source and a HIPPI destination before any data is transmitted. Every connection REQUEST signal sent to a HIPPI media card is accompanied by a HIPPI I-field.

The sequence that establishes a HIPPI connection is:

First:

> The HIPPI Source asserts the REQUEST line to a Destination, at the same time placing a 32-bit word, called the I-field, on the data lines.

Second:

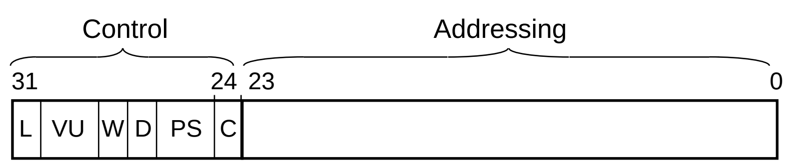
> The HIPPI Destination sees the REQUEST signal, reads the I-field, and accepts the connection by asserting its CONNECT line back to the Source.

Data coming from an external HIPPI I/O channel may be formatted into standard IP packets. Embedded in the front of each IP packet is an IP header. The media card reads the header only if told to do so by information in the HIPPI I-field. If the I-field tells the card to read the IP header, then it is an IP connection.

How the I-field is used

The I-field tells the GRF how to process the connection and where to send the data.

Figure 7-1 shows the basic structure of a HIPPI I-field:



L = Locally administered bit (L = 0)
VU = Vendor unique bits (not used)
W = Double-wide bit (not used)
D = Direction bit
PS = Path selection bits
C = Camp-on bit

(g0026)

Figure 7-1. HIPPI I-field components

Connection control information is in the leftmost 8 bits. Addressing takes up the other 24 bits.

Camp-on bit

The C (camp-on) bit is set on or off, 1 or 0. The HIPPI Source host uses camp-on to tell a HIPPI device (switch or router) to wait until a busy destination becomes available and to keep trying to make the connection.

Path selection bits

The path selection (PS) bits have four settings that tell the HIPPI media card how to read the 12-bit Destination address.

00 Source Routing

When Path Selection is set 00, the HIPPI Source has selected the exact route to the destination. This means the HIPPI host knows the specific path through some number of devices (switches/routers) to get to the endpoint host. In fact, the rightmost bits of the I-field (bits 0–23) contain the physical output slots for each switch/router in the path.

In the example shown in Figure 7-2, host A is to send data to host B through two switches and a GRF. The I-field sent by host A contains the output slot addresses in the order they will be used (starting at bit 0):

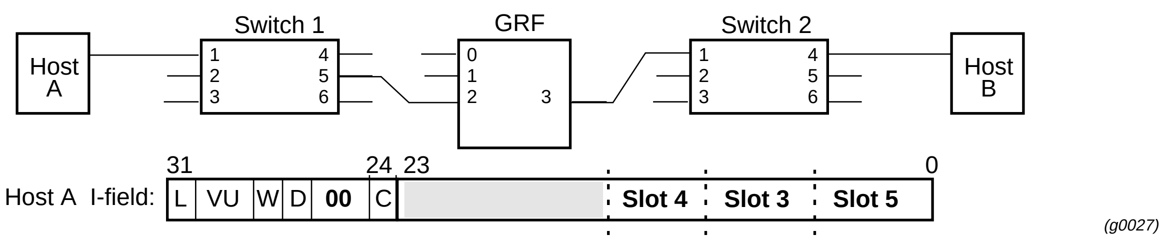


Figure 7-2. I-field for source-directed routing

The switch reads the first slot address starting at the right side of the list (bit 0). This is the output interface it will send the data to.

In source routing, a return path is automatically “built” by the network device at each point of data transfer. Figure 7-3 shows the return path created in the source routing example illustrated in Figure 7-2.

Switch 1 sees the Slot 1 I-field and copies the input interface address beginning at bit 23 of the I-field sent by Host A. Then the GRF copies Slot 2’s input interface address beginning at bit 23, shifting the prior address to the right. Switch 2 copies Slot 3’s input interface address beginning at bit 23, again shifting the prior addresses to the right.

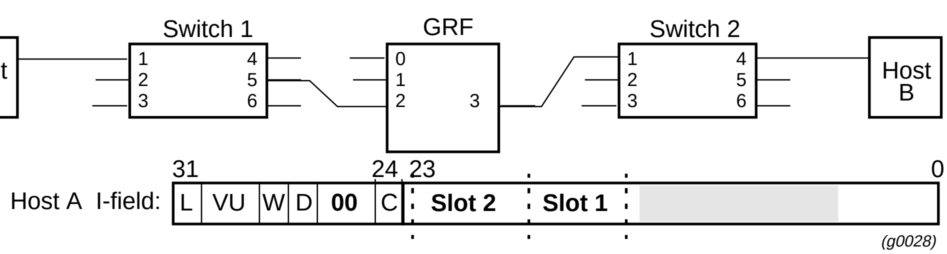


Figure 7-3. Return path created in source routing

Example 1 in this chapter describes how to configure source routing.

01 Logical address

When PS is set to 01, logical address, the host does not know or want to specify the actual physical route to the target endpoint. The host supplies a logical address for the endpoint host. In this case, all switches and the GRF must be programmed to route the connection.

The structure of the I-field is different when logical addressing is used. The 24 bits of destination addressing are divided into two 12-bit fields:

When PS is set to 01, the rightmost 12 bits of the I-field contain the logical address of the target endpoint. As shown in Figure 7-4, the leftmost 12 bits contain the logical address of the Source host; host A.

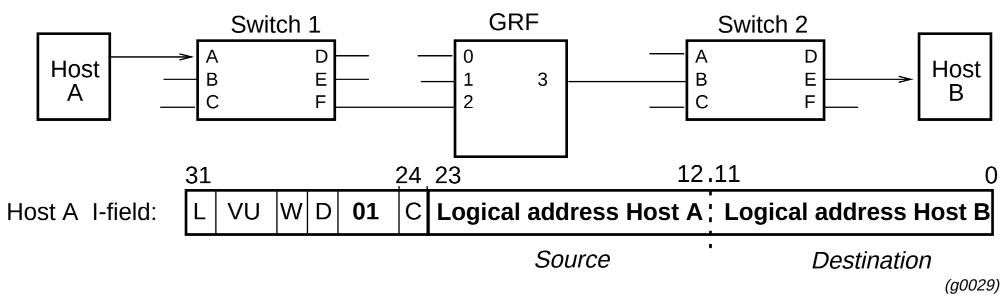


Figure 7-4. I-field for logical addressing (PS is set to 01)

Each switch/router has to look up the destination host logical address in its own tables and decide which of its output slots it will transfer the data to. In the table, there may be several output slots that could be used in the route. PS set to 01 specifies that the first entry in the list of possible output ports must be used. When PS is set to 11, data can be sent to any of the listed output slots. This is described in the *11 Logical address* section.

IP connection

An I-field containing a special logical address and the PS = 01 setting is used to establish an IP connection with a GRF HIPPI card. In the I-field, path selection (PS) bits are set to 01 or 11, and bits 0–11 contain a designated destination logical address (default = `0xfc0`, which is mapped to slot 64).

After the IP connection is established, the data packets arriving at the GRF are routed to the appropriate output slot using the default or a site-specified IP destination logical address. This means that data is transferred using a table based on IP addresses rather than HIPPI addresses. IP routing is discussed later in this chapter. Example 2 describes how to configure logical addresses.

10 Unused

This PS setting is not currently specified for use by the HIPPI-SC standard.

11 Logical address

This section relates to the prior section, “Logical address. “

The PS = 11 setting is the same as the 01 setting except that the switch or GRF can choose an output slot from a list of valid slots for this logical address. With PS = 01, the first port in the list is always used.

When PS is set to logical address, the host does not know the route and instead supplies a logical address for the endpoint host. In this case, all switches and the GRF must be programmed to route the connection.

The structure of the I-field is changed when logical addressing is used:

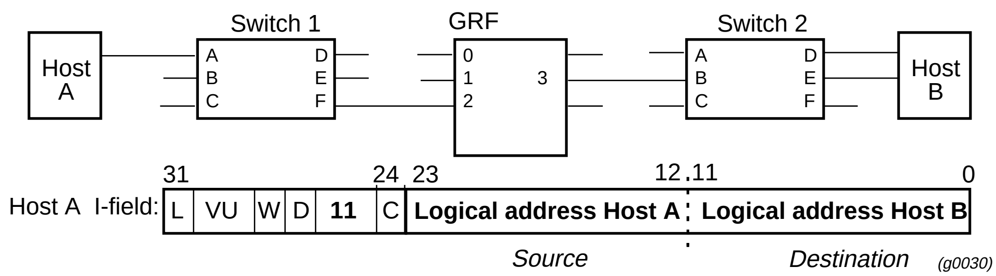


Figure 7-5. I-field for logical addressing (PS is set to 11)

When PS is set to 11, the rightmost 12 bits of the I-field contain the logical address of the target endpoint. The leftmost 12 bits contain the logical address of the Source host, in the example above, host A. Each switch/router has to look up the destination host logical address in its own tables and decide which of its output ports it will transfer the data to. In the table, there may be several output ports that could be used in the route.

Example 2 in this chapter describes how to configure logical addresses.

Direction bit

HIPPI hosts set the direction bit (D) to be either 0 or 1. The bit changes how a switch/router reads the 24 bits of destination address information. The previous illustrations for source routing and logical addressing are shown with the destination address information organized as if the host has set the destination bit to 0:

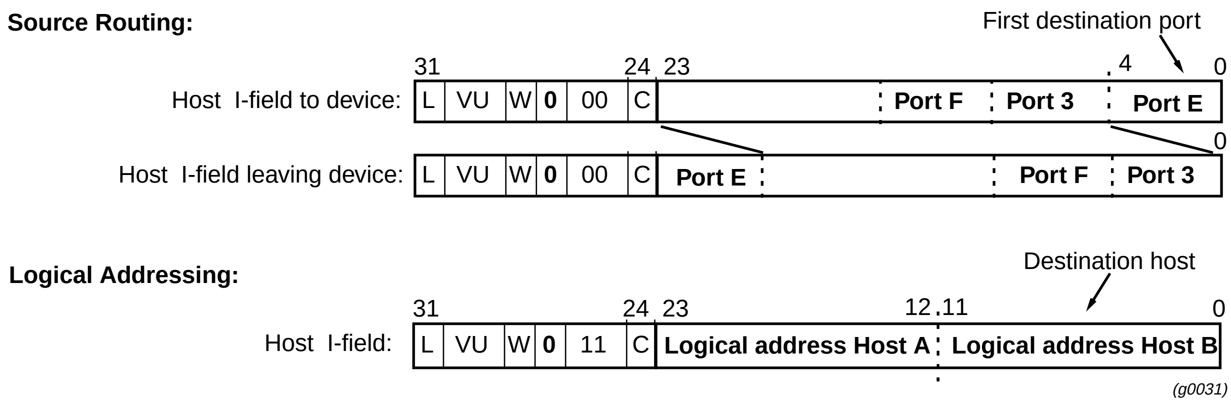


Figure 7-6. Source routing and logical addressing with D = 0

When the destination bit is set to 1, the information in the 24 bits of destination addressing is read starting from the left. The media card bits (in this case, Port E) are shifted to the left.

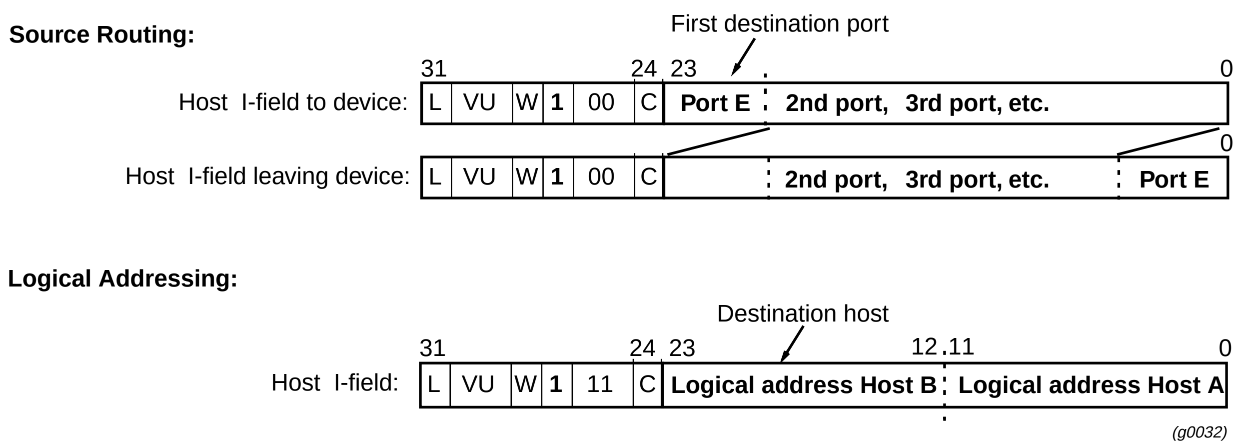


Figure 7-7. Source routing and logical addressing with D = 1

Destination bits can be used by Source and Destination hosts to reply and reverse-transfer data to one another, or as a means to trace where a connection originates.

L, VU, and W bits

The L and VU control bits are not used by the GRF. HIPPI media cards do not support double-wide HIPPI connections; they will reject the connection if the W bit is set. Additional information about the I-field can be found in the HIPPI-SC standard, ANSI X3.222.1993.

Taking stock...

We have described how a HIPPI host sends along an I-field with its request to be connected to a HIPPI media card. We have gone into some detail about the addressing information contained in the I-field and how the HIPPI media card uses the addressing information to transfer data.

We started out with the fact that the GRF processes two kinds of HIPPI connections:

– IP routing

– raw (HIPPI-to-HIPPI)

You might have noticed that the I-field very neatly supports logical addressing and source-specified routes between what appear to be two HIPPI hosts connected by any combination of devices that read I-fields, but are, in essence, HIPPI hosts talking to one another.

So far, we have talked about the GRF operating in a HIPPI I-field world, supporting only raw mode switching.

Beyond HIPPI

How does the GRF process connections that come from HIPPI hosts but which carry data destined to be used by nodes out on an FDDI ring? Or vice versa?

This is the kind of processing for which the GRF is designed.

This is why, in addition to raw mode switching, the HIPPI media card also does IP routing, that is, packet routing using the Internet Protocol and Internet addresses. Using IP routing, HIPPI data can be sent through the GRF, out to any other attached media, and on to any IP destination.

IP routing

In an IP connection, data coming from a HIPPI I/O channel is formatted into standard Internet Protocol (IP) packets. Embedded in the front of each IP datagram is the IP header. The media card reads the header only if told to do so by information in the HIPPI I-field indicating that this is an IP connection.

This header contains the Internet address of the host sending the datagram and the Internet address of the target IP-media host for whom the datagram is intended. This target host can be attached to any media that supports IP, or be reached via that attached media.

Because the GRF is a router, it creates and updates an IP routing table that describes paths to destination addresses. This is the basis of IP routing.

Each GRF media card has a copy of this IP routing table. When processing an IP connection, a HIPPI media card "opens" the datagram's header, reads the address of the target host, and determines which GRF media card to transfer the IP datagram to.

More information about IP routing via HIPPI is available in RFC 1374, *IP and ARP on HIPPI*. IP headers are described in RFC 791.

What is an IP datagram ?

In IP routing, the “currency” of internetwork data exchange is the datagram. The IP header functions as an envelope that can “carry” or “wrap around” the currency of specific media: FDDI frames, HIPPI packets, or ATM cells. In an IP datagram, frames, packets, and cells are freely routed via the IP protocol.

IP routing and the I-field

A HIPPI host’s I-field table can be used to direct the GRF HIPPI media card to do IP routing.

In the I-field, path selection (PS) bits are set to 01 or 11 and a designated destination logical address (default = `0xfc0`) is in bits 0–11. The mapping of this address to slot 64 in `grlamap.conf` indicates to the receiving media card that it is an IP connection and to read the IP header.

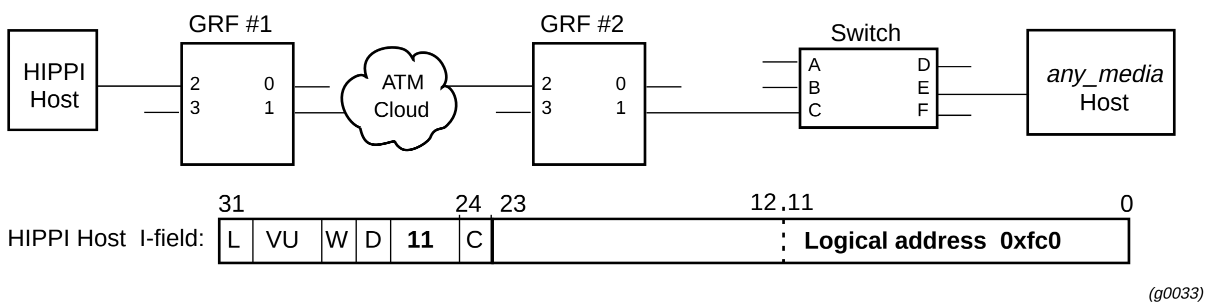


Figure 7-8. I-field 0xfc0 entry that triggers IP routing

The logical address for `0xfc0` is preset as a default in the logical address table. This logical address maps to a non-existent slot (64).

HIPPI cards are programmed so that when they look up an address that points to slot 64, they accept the connection and extract the destination IP address from the first datagram’s header.

A site can set any logical address or addresses to designate an IP connection by editing the `grlamap.conf` file. The only requirement is that the site-specified address must map to destination slot 64. As noted, address `0xfc0` does not need to be added. It is preset in the `grlamap.conf` file.

Using the IP address

The receiving HIPPI card looks up the destination IP address in the routing table, finds the corresponding GRF output media card, and forwards the datagram across the switch.

The output media card just forwards the data *unless* it is a HIPPI card connected to a HIPPI switch. In this case, the output HIPPI card needs an I-field to forward when it requests a HIPPI connection to the switch.

You need to supply the I-field by editing the `/etc/grarp.conf` file using the **grarp** command.

grarp supports address resolution for all GRF media. The command maintains a mapping of IP addresses to physical addresses in the `/etc/grarp.conf` configuration file and displays the file's contents. HIPPI ARP entries are defined in `/etc/grarp.conf`.

In addition to the information presented here, a **grarp** man page is also available.

Examples 3 and 4 describe how to configure IP routing step-by-step.

HIPPI in a bridging environment

HIPPI does not bridge. On the GRF, you can route IP to a bridge group from a HIPPI routing domain, but there is no encapsulated bridging across a HIPPI connection.

MTU

The HIPPI maximum transmission unit (MTU) is 65280 bytes. A different MTU can be specified in the`/etc/grifconfig.conf` file, in the **arguments** field. Refer to the *GRF Reference Guide* for the format of`/etc/grifconfig.conf` entries.

ARP

HIPPI ARP tables are manually configured for remote devices connected to GRF HIPPI interfaces. The `/etc/grarp.conf file` maps an IP address to a 32-bit HIPPI I-field address that uniquely defines a remote HIPPI host.

HIPPI standards and RFCs via ftp

HIPPI standards are publicly available by anonymous ftp at the site: `ftp.network.com`.

Files are in the `/hippi` directory. To obtain the HIPPI-SC standard, make your request using:

```
ftp> get hippi-sc_2.7.ps
```

Enter:

```
ftp ftp.network.com
Connected to ftp.network.com.
Name:    anonymous
331 Guest login ok, send ident as password.
Password:    (enter your email address here)
230 Guest login ok, access restrictions apply.
ftp> cd hippi
ftp> get hippi-sc_2.7.ps
ftp> exit
```

RFCs 1374 (*IP–ATM*) and 1483 (*IP and ARP–HIPPI*) are also available in ASCII text format by anonymous ftp at the site:

```
ftp.isi.edu
```

Files are in the `/in-notes` directory, with file names of the form `rfcnnnn.txt`. The file `rfc-index.txt` is an index of all RFCs.

HIPPI configuration options

This section uses examples to show how to set up various configurations by programming a HIPPI media card and, when necessary, a HIPPI host I-field.

The first three examples are HIPPI-to-HIPPI configurations:

Example 1:
when you know and want to specify the exact path from host to target endpoint:
use source routing

Example 2:
when you do not know or want to specify the exact path but do have a logical address for the target endpoint: *use logical addressing*

Example 3:
when you do not know the exact path but do have the IP address for the target endpoint:
use IP routing

A fourth example shows how to configure IP routing for HIPPI-to-other media connectivity.

Example 4:
when you do not know the exact path but do have the IP address for the target endpoint:
use IP routing

Two special cases are described:

Special case 1:
configuration options for IPI-3 routing

Special case 2:
how to configure the IBM HIPPI connection option, H0 HIPPI

Note: A site might assign logical addresses in decimal format. Decimal format is often converted into hex shorthand. Some file entries use binary equivalents, the I-field, for example.

Configuration files and commands use entries in a variety of formats. Follow the examples shown in the next sections. Examples are given in appropriate formats.

Example 1: Source routing – host selects the path

In the example here, HIPPI hosts exchange data using source routes over switches and a GRF router. There is one configuration step.

In each HIPPI host, set up the output slots in the I-field table to create a source-specified route. A host name or host IP address must be associated with a HIPPI I-field. No programming is required for the GRF HIPPI media card.

The HIPPI media card reads the leading output slot number in the I-field and transfers the data to the card in that slot. Boards in HIPPI switches that conform to the HIPPI-SC standard also read and use the I-fields in the same way.

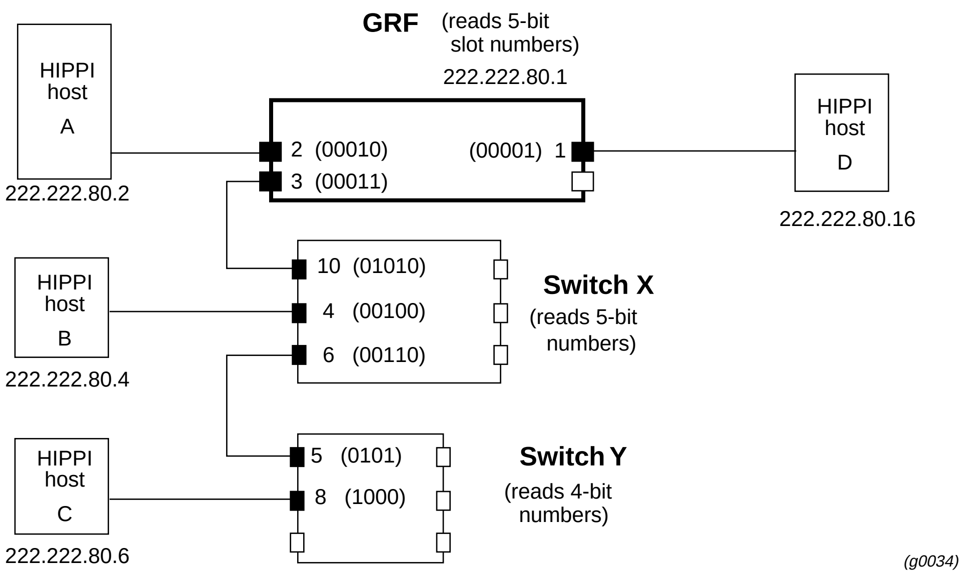


Figure 7-9. Planning diagram for source routing example

I-fields provide a HIPPI sending device with the number of the slot it needs to transfer the data to after a connection is established with the receiving device.

Remember that the originating HIPPI host is only the *first* sending device and the endpoint system is the *last* receiving device. Along the connecting route, each device is first a receiving device and then a sending device.

Collect host information

Each HIPPI device has an IP address. Determine the slot numbers for the GRF and any connected switches. Collect this information, then edit your host(s) I-field table. This manual does not document how to edit specific host I-field tables.

Set up host I-field table

Here is the list of I-fields for each host in the source routing example. The following assumptions are made:

- PS bits are 00 (selects source routing)
- D bit is 0 (leading slot number is rightmost)
- camp-on bit is site-determined
- 24-bit address field is filled in with 0s to left of slot numbers
- no spaces between slot numbers (the list does only for clarity)
- preface slot numbers with a zero for devices that read 5-bits

```
                     31                24 23                                            0
HIPPI host I-field:  | L | VU | W | 0 | 00 | C |    (fill)    : Slot # : Slot # : Slot # |

From Host A
     to Host B:      00000000000000      00100    0011
                     fill with 0s to 24 bits  4       3
     to Host C:      0000000000 1000     00110    0011
                                   8        6       3
     to Host D:      0000000000000000000          0001
                                                    1
From Host B
     to Host A:      00000000000000       0010   01010
                                            2      10
     to Host C:      000000000000000      1000   00110  <– output slot numbers
                                                           (in binary)
                                            8       6
     to Host D:      00000000000000       0001 01010
                                            1     10
From Host C
     to Host A:      0000000000 0010      01010   0101
                                   2        10       5
     to Host B:      000000000000000      00100   0101
                                              4      5
     to Host D:      0000000000  0001      01010  0101
                                    1        10      5
From Host D
     to Host A:      0000000000000000000         01001
                                                    2
     to Host B:      00000000000000      00100    0011
                                            4       3
     to Host C:      0000000000 1000     00110    0011
                                   8        6       3
                                                                       (g0035)
```

Figure 7-10. I-field list for source routing example

This completes the configuration process for source routing.

Example 2: Using logical addresses

When logical addresses are used, the HIPPI host supplies a logical address in the I-field for the endpoint HIPPI host. In this case, you program the switches and the GRF with lists of physical slots for each logical address.

The structure of the I-field is different when logical addressing is used. The 24 bits of destination addressing are divided into two 12-bit fields:

Each switch/router has to look up the destination host logical address in its own tables and decide which of its output slots it will transfer the data to.

In the table, there can be several output slots that could be used in the route. Path selection bits (PS) set to 01 specifies that the first entry in the list of possible output ports must be used. PS set to 11 specifies that any output slot in the list can be used.

There are two steps to configure logical addressing:

1 In each HIPPI host, insert 12-bit logical addresses in the I-field table for both the source host and the destination endpoint.

2 In the GRF, edit the logical address file, `/etc/grlamap.conf`.
In this file enter the 12-bit logical addresses for the destination endpoints, and also specify which GRF output slots these addresses map to. Then execute the **grlamap** command to initialize the logical address file and distribute it to each media card.
(Note that the **grlamap** command automatically executes each time a HIPPI media card boots.)

You also must program any connected HIPPI switches using their command sets.

Here is the planning diagram for the logical addressing example:

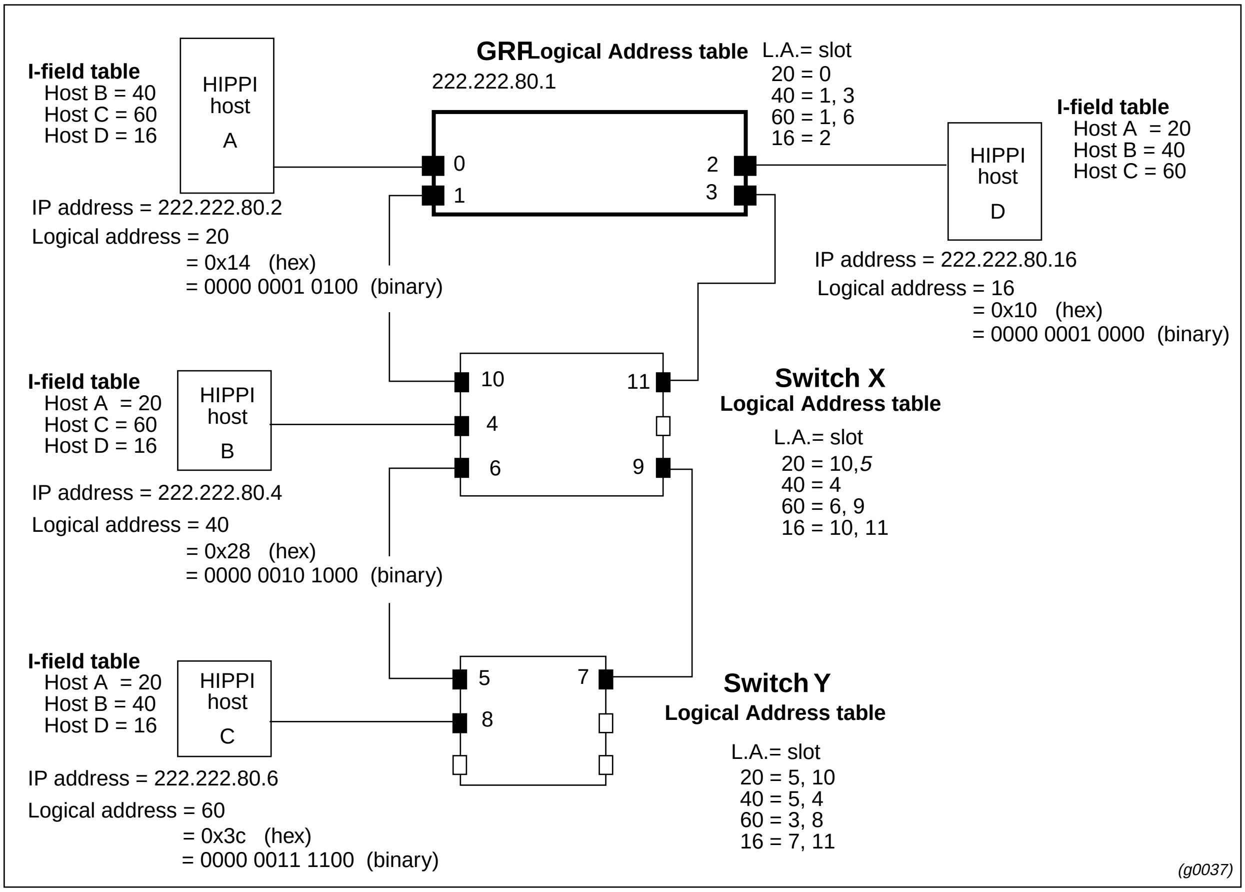


Figure 7-11. Planning diagram for logical addressing

Logical addressing configuration example

In the planning diagram, four hosts connect to each other through two HIPPI switches and four GRF HIPPI media cards.

An I-field table is provided for each host. In each host I-field table there are logical addresses for all destination hosts, A through D.

A representative logical address table is shown for each switch and the GRF. These tables map a destination host's logical address with the output slot number(s).

The I-fields provide each HIPPI sending device with the logical address of the receiving device they need to transfer the data to after a connection is established with the receiving device.

Remember that the originating HIPPI host is only the *first* sending device and the endpoint system is the *last* receiving device. Along the connecting route each device, for example, a switch, is first a receiving device and then a sending device.

Set up host I-field logical addresses

Set up the host I-field tables. These are I-field values that relate to the example:

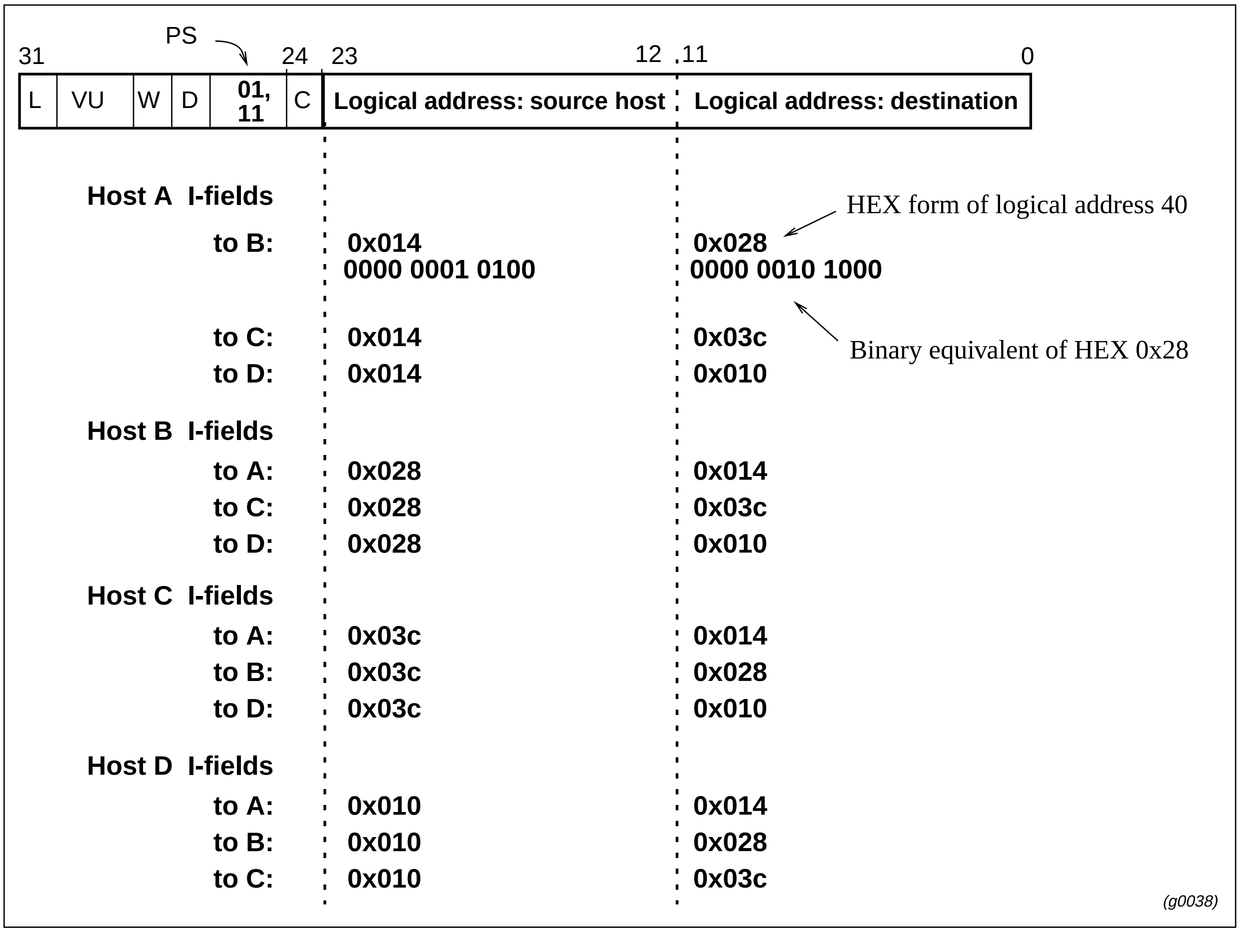


Figure 7-12. Sample host I-field table for logical addressing

The diagram shows hex equivalents of decimal logical address numbers. The GRF accepts either format in the I-field tables.

A site might assign logical addresses in decimal as is shown in the planning diagram. The decimal is often converted into hex shorthand.

Edit the logical address file – /etc/grlamap.conf

In this second step, edit the `/etc/grlamap.conf` file. Use a UNIX editor to open the file and insert the needed entries.

The format at each entry of `/etc/grlamap.conf` is:

```
portcard  logical_address  dest_portcard
```

where:

`portcard` is the slot number of the media card being configured.

`logical_address` is the logical address of the destination device.

`dest_portcard` can be up to four destination slot numbers to which the logical address will be mapped.

Referring back to the logical addressing example, these are the entries you would add to the `/etc/grlamap.conf` file to set up the correct GRF logical address table:

```
portcard     logical_addr     dest_portcard
0          0x28             3,1
0          0x3c             1,3
0          0x10             2
1          0x14             0
1          0x28             3,1
1          0x10             2
1          0x3c             1,3
2          0x14              0
2          0x28             3,1
2          0x3c             1,3
3          0x10             2
3          0x14             0
3          0x28             3,1
3          0x3c             1,3
```

Downloading new mappings

The **grlamap** mappings are downloaded to the HIPPI media card in two ways:

- automatically at media card boot
- by using the **grlamap** command

Execute grlamap

This command reads logical address information from the `/etc/grlamap.conf` file and uses the **grinch** command to download the configuration information to the appropriate media card. The following command downloads the new mappings to a specific media card:

```
#  grlamap -p <slot number>
```

Use `all` in place of the slot number to download to all media cards. **grlamap** has a number of options. Refer to the **grlamap** man page or the *GRF Reference Guide* for more information.

This completes the GRF configuration process for logical addressing.

Example 3: IP routing – HIPPI-to-HIPPI across a switch

This example discusses IP routing between two HIPPI hosts across the GRF and a HIPPI switch. In the planning diagram shown in Figure 7-13:

– the GRF functions as a high performance LAN backbone

– the GRF has two HIPPI cards in slots 1 and 0

– HIPPI host A transfers data to HIPPI host C across the GRF and through a HIPPI switch

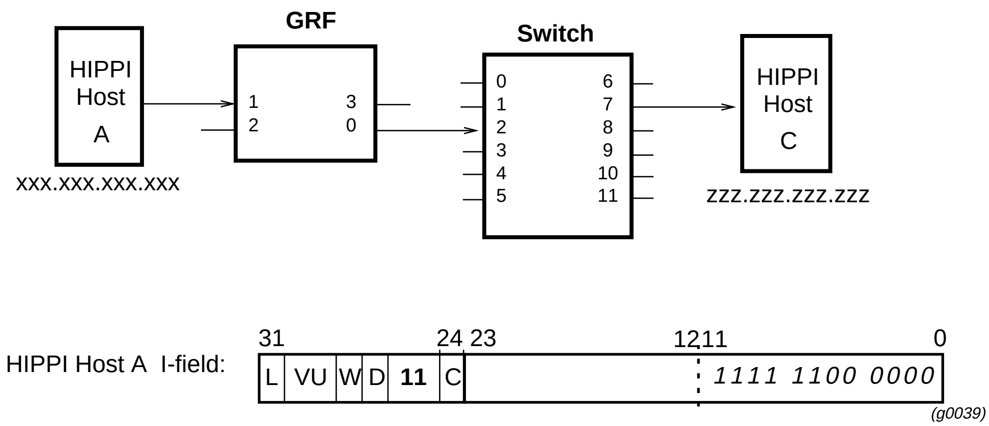


Figure 7-13. Planning diagram for HIPPI-HIPPI IP routing with switch

HIPPI-to-HIPPI IP routing process

The originating HIPPI host establishes IP routing with a special address in the I-field it forwards to the GRF.

With IP routing established, the receiving HIPPI media card (slot 1) opens the IP header at the front of a datagram, reads the target host's IP address, and looks it up in its route table.

The route table tells the receiving card which output HIPPI card (slot 0) to transfer the data to.

Configuration steps

To configure IP routing in this example, there are three steps:

1 Set up the host's I-field to tell the GRF that an IP connection is desired.

In the host's I-field table, enter either the site-designated logical address for IP routing, or the GRF default. The GRF default address is `0xfc0`.

As the host requests a connection and the HIPPI card reads the designated logical address (`0xfc0`) in the I-field, the HIPPI card will automatically process the connection in IP routing mode.

2 Skip this step if you used the default `0xfc0` address in the host's I-field table in Step 1.

Otherwise, enter the address you designated for IP routing in the GRF's `/etc/grlamap.conf` file. Set the destination slot as 64.

3 Update the GRF's IP address table with the **garp** command.

Add one entry for each IP address the GRF needs to know. This usually means an entry for each host and target device connected to the GRF.

The next several pages take you through the steps.

Set up host I-field table to establish IP routing

The I-field for host A can carry the host's own logical address (source) in bits 12–23. This source address is optional since the GRF does not use it.

The destination address in bits 0 – 11 is important. Host A's I-field must carry either the site-designated logical address for IP routing or the GRF default (`0xfc0`).

Here is the format of the I-field for host A with the following assumptions:

- PS bits are 01 or 11
- D bit is 0 (destination address is rightmost)
- camp-on bit is site-determined
- GRF default address is used to specify IP routing

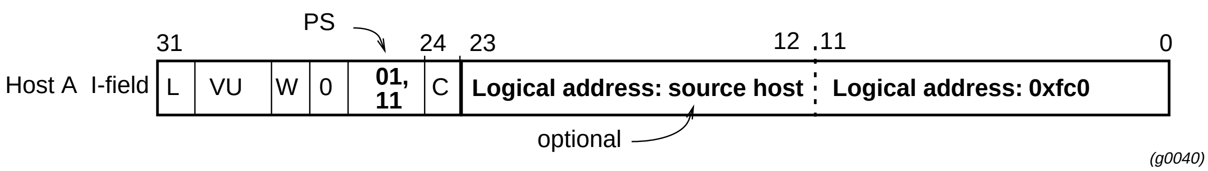


Figure 7-14. I-field for HIPPI-HIPPI IP routing example

Set site-specified address for IP routing

If you used the GRF default `0xfc0` address, skip this step.

To designate a site-specified address for IP routing, edit the `/etc/grlamap.conf` file and then run the **grlamap** command. Open the file and use a UNIX editor to insert the values needed.

The format at each entry of `/etc/grlamap.conf` is:

```
portcard   logical_address   dest_portcard
```

where:

`portcard` is the slot number of the media card to which the HIPPI host (host A) connects.

`logical_address` is the site-designated 12-bit address for IP routing.

`dest_portcard` must be set to 64.

Based on example 3, these entries are made to `/etc/grlamap.conf`:

```
portcard      logical_addr      dest_portcard
1       <site-specified address>    64
```

Downloading new mappings

The `/etc/grlamap.conf` mappings are downloaded to the HIPPI card in two ways:

– automatically at media card boot

– by using the **grlamap** command

Execute grlamap command

This command reads logical address information from the `/etc/grlamap.conf` file and uses the **grinch** command to download the information to the appropriate media card. The command given below downloads the new mappings to a specific media card:

```
# grlamap -p <slot number>
```

For our example, use this command to download the new information to media card 1:

```
# grlamap -p 1
```

Use `all` in place of the slot number to download to all media cards.

grlamap has a number of options. Refer to its man page or the *GRF Reference Guide* for more information.

Map output IP address to output I-field – grarp command

In this step, set up a path through and out of the GRF to a HIPPI switch.

Use the **grarp** command to tell the input HIPPI card which output GRF card to send the data to, and provide an I-field for the output HIPPI card to use when requesting a connection to the switch. The IP address of the destination host is used to “link” both types of information — output media card number and destination I-field.

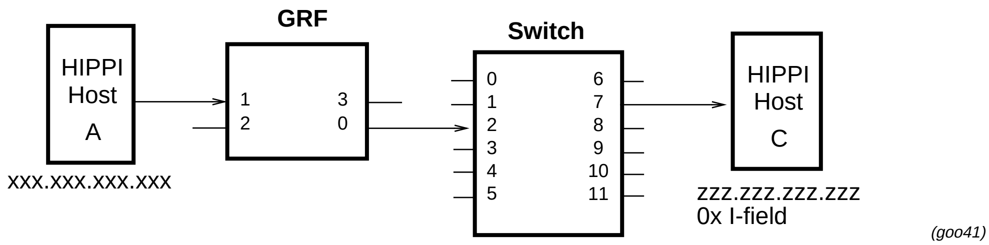


Figure 7-15. Mapping an IP address to a destination I-field

Link destination IP address to output media card

After the input HIPPI card reads the host-supplied I-field, that I-field is discarded and the host-GRF connection is established. The input HIPPI card reads the destination host IP address from the packet header and looks up that address in its route table. The table tells the input card which output card to transfer the data to.

Link destination IP address to forwarding I-field

As part of requesting a connection to a HIPPI switch on the path to the destination host, the output media card must send an I-field. This is part of the basic HIPPI connection request process.

In Example 3, GRF output card 0 has to supply an I-field as part of its request to connect to the HIPPI switch. The GRF only forwards an I-field when a HIPPI switch is between the GRF and the target endpoint HIPPI host.

Execute grarp command(s)

This step sets up a table that collects and relates the destination host IP address, the output media card number, and the HIPPI I-field illustrated in Figure 7-15.

To enter addresses in the IP address table, execute one **grarp** command for each destination IP address the GRF needs to know about.

To configure the GRF as shown in the figure, enter one **grarp** command for host C and another for host A.

The format of the **grarp** command is:

```
grarp -s   hostname   phys_addr   -i  <ifname>
```

where

hostname specifies the HIPPI host by name or by number using Internet dot notation.

phys_addr is the site-specified I-field containing a logical or source address for the destination host.
(used by the GRF output card to request a connection with an intervening switch)

ifname is the GRF interface name in the format `gx0yz`.

Here are the commands needed to configure the GRF:

```
# grarp -s zzz.zzz.zzz.zzz  <0x i-field_hostC>  -i  gh000
# grarp -s xxx.xxx.xxx.xxx  <0x i-field_hostA>  -i  gh010
```

This manual cannot tell you how to devise the I-field to assign to host C; the I-field will be site-specific. A site can choose to assign a logical address for the target host or to use a source route address.

This completes the HIPPI-HIPPI IP routing configuration process.

Example 4: IP routing – HIPPI-to-IP media

IP routing is often used when the target host is on a remote network. Setting up a configuration for IP routing is the same whether data is to transfer between two HIPPI hosts or between HIPPI and other media.

This example discusses IP routing between two hosts (one of which is HIPPI) across two GRF routers and a WAN. Figure 7-16 contains the example's planning diagram. Both GRFs function as high-performance LAN backbones, and connect over ATM or FDDI.

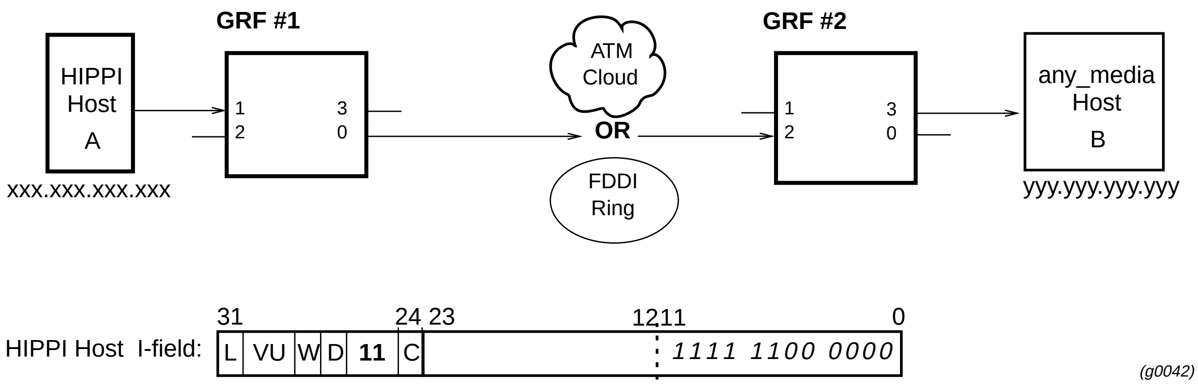


Figure 7-16. Planning diagram for HIPPI IP routing

GRF #1 has one HIPPI card connecting to host A and one ATM or FDDI card connecting to the WAN. Host A transfers data to host B. GRF #2 has one media card connecting to host B and one ATM or FDDI card connecting to the WAN.

The example shows how to configure GRF #1 and GRF #2 to support host A-to-host B transfers.

Host A-to-B IP transfers

IP routing is established by values in the originating host's I-field.

The IP datagrams from host A carry host B's IP addresses in their headers. The receiving GRF HIPPI card opens the header, reads the target host's IP address, and looks it up in its route table. The route table tells that media card just which output media card to transfer the data to, in this case the ATM or FDDI card in slot 0.

On the other side of the WAN, the receiving ATM or FDDI card in slot 2 repeats the process of reading and looking up the IP address in the route table, and finds out that the datagram should be transferred to output slot 3.

Configuring GRF #1

1 Set up the host's I-field to tell the GRF that an IP connection is desired.

In the host's I-field table, enter either the site-designated logical address for IP routing or the GRF default. The GRF default address is `0xfc0`.

When the host requests a connection and the HIPPI media card reads the `0xfc0` address in the I-field, the media card will automatically process the connection in IP routing mode.

2 Skip this step if you used the default `0xfc0` address in the host's I-field table in Step 1.

Otherwise, enter the address you designated for IP routing in the GRF's `/etc/grlamap.conf` file. Set the destination slot as 64.

Execute the **grlamap** command:

```
# grlamap -p 1
```

3 If the media card connecting to the WAN is ATM, set up a permanent virtual circuit (PVC) for the ATM card. Refer to the *ATM Configuration* chapter for information.

If the media card connecting to the WAN is FDDI, it will accept and forward the IP datagrams with no further programming.

Configuring GRF #2

If the media card connecting to the WAN is ATM, set up a permanent virtual circuit (PVC) for the ATM card. Refer to the *ATM Configuration* chapter for information.

If the media card connecting to the WAN is FDDI, it will accept and forward the IP datagrams with no further programming.

The next section takes you through the steps.

Set up host I-field table

The I-field for host A can carry the host's own logical address (source) in bits 12–23. This source address is optional since the GRF does not use it. The destination address in bits 0 – 11 is important. Host A I-field bits 0 – 11 must carry either the site-designated logical address for IP routing or the GRF default (`0xfc0`).

Here is the format of the I-field for host A with the following assumptions:

– PS bits are 01 or 11

– D bit is 0 (destination address is rightmost)

– camp-on bit is site-determined

– GRF default address is used to specify IP routing

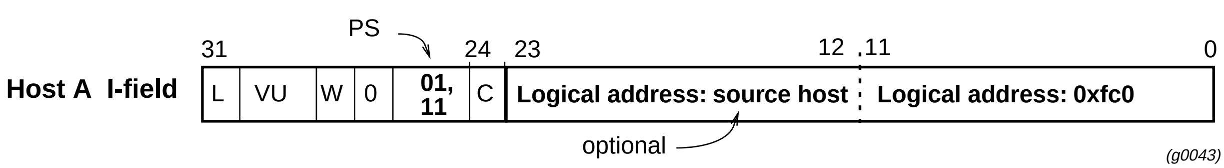


Figure 7-17. I-field for IP routing example

Set I-field for IP routing

If you used the GRF default `0xfc0` address, skip this step.

To designate a site-specified address for IP routing, edit the `/etc/grlamap.conf` file and then run the **grlamap** command.

Use a UNIX editor to open the file and insert the values needed.

The format at each entry of `/etc/grlamap.conf` is:

```
portcard  logical_address  dest_portcard
```

where:

`portcard` is the slot number of the GRF #1 media card to which the HIPPI host (host A) connects

`logical_address` is the site-designated 12-bit address for IP routing

`dest_portcard` must be set to 64

Based on Example 4, this is the entry you make to `/etc/grlamap.conf`:

```
portcard     logical_addr     dest_portcard
1       <site-specified address>     64
```

Now execute **grlamap** to load the table.

Execute grlamap command

This command reads logical address information from the `/etc/grlamap.conf` file and uses the **grinch** command to download the configuration information to the appropriate media card.

```
# grlamap -p 1
```

This command does the basic downloading for media card 1.

Configure WAN media card

If the media card is ATM, reserve a permanent virtual circuit (PVC). Refer to the ATM OC-3c configuration chapter for information.

If the media card is FDDI, install the card and configure its SAS/DAS attachments. Refer to the FDDI configuration chapter for information. .

Special case 1: HIPPI IPI-3 routing

IPI-3 is a protocol used primarily by large storage devices that have a HIPPI I/O channel. These storage devices can be cabled directly to a supercomputer's HIPPI channel, but increasingly the devices are configured as shown in Figure 7-18 as a shared resource on a high-performance network. Since the IPI-3 peripheral protocol operates on the GRF's HIPPI card as raw HIPPI-SC, the protocol is essentially transparent to the media card.

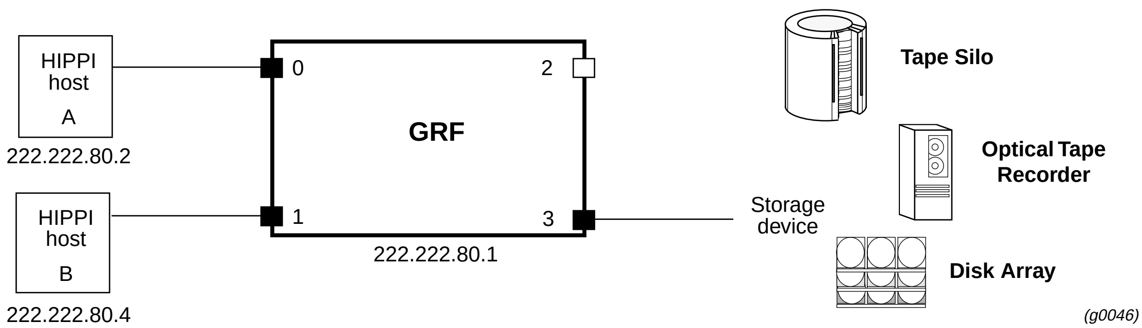


Figure 7-18. Planning diagram for HIPPI IPI-3 configuration

Choose whether to use source routing or logical addressing to configure these devices and then follow the steps given in the appropriate example. Treat the storage device as a HIPPI host.

Special case 2: IBM H0 HIPPI option

This option supports direct connection to IBM 3090 mainframes set up with the IBM H0 HIPPI interface. The H0 HIPPI option enables such IBM hosts to interconnect with other hosts via the GRF.

The IBM H0 HIPPI interface was developed in accordance with early HIPPI draft standards. The standards have since changed, leaving the IBM H0 HIPPI interface with certain characteristics:

– the I-field is always 0 (there is no user control of the I-field transmitted by the IBM H0 HIPPI)

– the IBM H0 HIPPI sends packets that are multiples of 4096 bytes in length, and must receive the same

– performance is substantially improved if the HIPPI connection remains open

Media card functions

On a GRF HIPPI card, the IBM_H0 mode works as follows:

– the HIPPI media card assumes all packets from the IBM H0 device contain IP datagrams to be routed, and ignores the I-field received from the IBM H0

– the HIPPI connection from the media card to the IBM H0 device is not dropped between packets

– the media card adds zero padding as needed to the end of output packets going to an IBM H0 device to make them a multiple of 4096 bytes in length

Two standard HIPPI features help to support this interface. The HIPPI media card allows any number of packets to be sent to it in a single HIPPI connection. Before forwarding them to another GRF media card, the HIPPI media card strips any padding from input IP packets.

Enabling H0 mode

The IBM_H0 mode is enabled at the Card profile in the `ports` / `hippi` field. The default is IBM_H0 mode disabled. Set `IBM-HO-mode` to `enabled`.

Here is the path:

```
card-num* = 10
media-type = hippi
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = Frame-Relay
ether-verbose = 0
ports = < {0{off on 10 3} {single off} {"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }

super> list ports 0 hippi
```

```
debug-level = 1
ready-count = 32
testmode = no-mode
test-duration = 999999
test-pkt-size = 4
test-pattern = incremental
test-check-data = 5
max-Ifields = 300
out-timeout = 10
switch-timeout = 10
default-Ifield = 03:00:0f:c0
IBM-HO-mode = disabled
hold-connection = disabled
out-timeout-mode = 0
disable-raw = disabled
mcast-addr = 03:00:0f:e0
tunnel-table = { disabled 0 0 32 0 0 0 0 0 0 0 0 }
local-sw-addr = 00
arp-addr = 00

super> set IBM-HO-mode = enabled
super> write
CARD/10 written
super> grreset 10
super>
```

When you change any setting, you must reset the HIPPI card or reboot the system to start the new mode.

If you are not sure of which options are available for a field, you can get a brief description in this way:

```
super> set IBM-HO-mode ?
IBM-HO-mode:
 enable/disable IBM H0 Mode.
Boolean field, 'disabled' or 'enabled'
```

Looking at the HIPPI card

The HIPPI media card provides a single full-duplex interface. The card has one receive (from destination) interface and one transmit (to source) interface. The upper interface, A, is the RCV or destination interface. The lower interface, B, is the SRC or source interface. Figure 7-19 shows a HIPPI faceplate and LEDs.

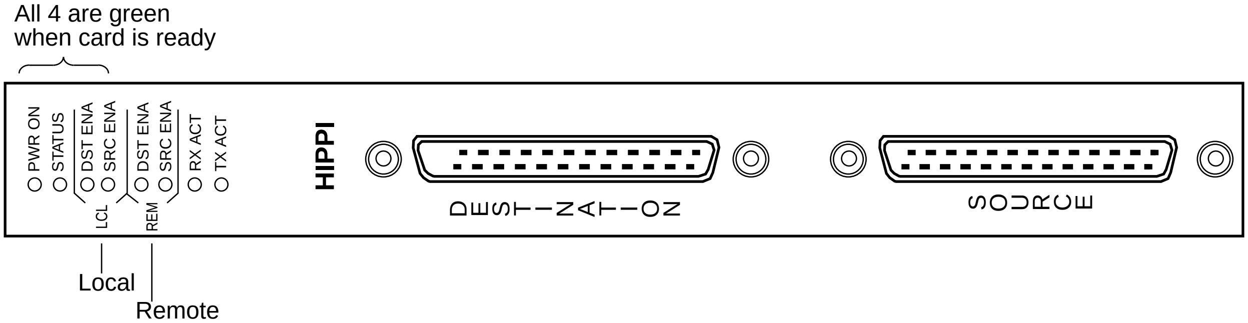


Figure 7-19. HIPPI media card faceplate and LEDs

LEDs on the faceplate

Refer to Table 7-1 for a description of HIPPI card LEDs.

Table 7-1. HIPPI media card LEDs

LED	Description
Power	This green LED is on when GRF power is on.
Status	When self-test completes, this green LED turns on and remains steadily on during normal operations. The Status LED blinks when an error condition is detected.
DST ENA (local)	This green LED is on when the input destination interface is asserting the interconnect signal and is ready for operation.
SRC ENA (local)	This green LED is on when the output source interface is asserting the interconnect signal and is ready for operation.
DST ENA (remote)	HIPPI directly connects to a HIPPI host or to a network device. This green LED is on when the remote destination interface is asserting the interconnect signal and is ready for operation.
SRC ENA (remote)	HIPPI directly connects to a HIPPI host or to a network device. This green LED is on when the remote source interface is asserting the interconnect signal and is ready for operation.
RX ACT	This green LED indicates data is being received at the input interface, the blink rate depends on the traffic load.
TX ACT	This green LED indicates data is being sent from at the output interface, the blink rate depends on the traffic load.

List of HIPPI configuration steps

These are the steps to configure HIPPI cards.

1 Assign an IP address to the logical interface in `/etc/grifconfig.conf`.

2 Specify HIPPI card parameters in the Card profile:

- Check I-field shift in the System profile, by default, the I-field shift is set to 5 bits
- Review HIPPI application / debug settings, check HIPPI host time-out value
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

These next steps are optional, they describe tasks that are performed infrequently:

3 Change Load profile (optional).
Global executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every HIPPI card.

If you want to change the run-time code in one HIPPI card, make the change in the Card profile, in the `load` section.

4 Change Dump profile (optional).
Global dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.
If you want to change dump settings for one HIPPI card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1. To save files in the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

2. In the command-line interface, use **set** and **write** commands to save a profile.

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configuring a HIPPI interface

This section describes how to configure a HIPPI interface in the `/etc/grifconfig.conf` file. Use a UNIX editor to make entries in this file.

Each logical HIPPI interface is identified as to its:

- interface name, `gh0yz` (names are always lower case)
- Internet address
- netmask
- broadcast/destination address (optional)
- arguments field (optional)

The format for an entry in the `/etc/grifconfig.conf` file is:

```
name   address    netmask       broad_dest    arguments
```

Interface name *gohyz*
Each logical GRF interface is given an interface name `gh0yz` where:

- the "gh" prefix indicates a SONET interface
- the chassis number is always "0"
- "y" is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
- "z" is the logical interface number in hex, on the HIPPI card it is 0

There is a single logical interface per HIPPI media card.

Address
Enter the IP address to be assigned to this interface.

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured.

Broadcast or destination address
When you configure a logical interface on a point-to-point media, enter the destination IP address in the `broad_dest` address field. If you do not specify a broadcast address, you create a non-broadcast, multi access (NBMA) interface.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO addess is being added to an interface's IP address. Specify the MTU value as `mtu` *xyz*. Leave the `arguments` field blank if you are not using it.

Example

The entry assigns an IP address for logical interface 0 on the HIPPI card in slot 6. If needed, a dash is used as a placeholder for the broadcast address:

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
gh060 192.0.2.1 255.255.255.0 192.0.2.255
```

If an interface is nonbroadcast (NBMA), do not include a destination address in its `/etc/grifconfig.conf` entry.

Save the /etc file

Save the file with the editor. Then, use **grwrite** to write the file to the `/etc` configuration directory:

```
# grwrite -v
```

Check contents of /etc/grifconfig.conf file

After you save the `/etc` directory and reset the media card, use **netstat -in** to display the contents of the `/etc/grifconfig.conf` file and verify that the logical interface is configured with the correct IP address.

Here is the output from a **netstat** command looking at the HIPPI interfaces:

```
# netstat -in | grep gh
gh0a0   65280 <link24>                   13     0     36     0     0
gh0a0   65280 203.3.11    203.3.11.156   13     0     36     0     0
```

Please refer to the **netstat** man page for information about other **netstat** options.

Check system-level IP configuration

The UNIX **ifconfig *interface*** command returns system level information for the specified interface name, here is the display for logical interface 0 (`gh0a0`):

```
# ifconfig gh0a0
gh0a0: grithippi flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 65280
        inet 203.3.11.156 netmask 0xffffff00 broadcast 203.3.11.255
```

Setting parameters in the Card profile

This section describes how to verify and/or change HIPPI parameters in the Card profile. The parameters are presented in this order:

– Check I-field shift in the System profile, by default, the I-field shift is set to 5 bits
– Review HIPPI application / debug settings, check HIPPI host time-out value
– OPTIONAL: specify ICMP throttling settings
– OPTIONAL: change run-time binaries
– OPTIONAL: change dump variables

1. Check I-field shift setting

The I-field shift setting is in the System profile. By default, it is set to 5. The other option is to set it to 4, depending upon hardware requirements.

```
super> read system
SYSTEM read
super> list
os-level = 1.4.12
hostname = gomez.site.com
chassis = GRF1600
ip-address = 206.146.160.156
netmask = 0.0.0.0
default-route = 0.0.0.0
hippi-ifield-shift = 5
enable-congest = disabled
num-slots = 16
rmb-load-path = /usr/libexec/portcards/rm.run
rmb-dump-config = 4
physical-memory = 64
hardware-revision = "Not Available"
chassis-revision = 1
xilinx-revision = 8
num-fans = 2
num-pwr-supply = 1
Forward_Directed_Bcast_Pkts = disabled

super> set hippi-ifield-shift = 4
super> write
CARD/10 written
```

2. Review HIPPI settings

The GRF functions as more than a switch. Its switching capability is accompanied by routing procedures, buffering, speed matching, and other features. One result can be a lower than expected switch response. For this reason, we suggest that you adjust the connecting HIPPI host time-out values to 10 milliseconds or more.

The setting is at the Card profile in the `ports` / `hippi` field:

```
super> read card 10
```

```
CARD/10 read
super> list ports 0 hippi
debug-level = 1
ready-count = 32
testmode = no-mode
test-duration = 999999
test-pkt-size = 4
test-pattern = incremental
test-check-data = 5
max-Ifields = 300
out-timeout = 10
switch-timeout = 10
default-Ifield = 03:00:0f:c0
IBM-HO-mode = disabled
hold-connection = disabled
out-timeout-mode = 0
disable-raw = disabled
mcast-addr = 03:00:0f:e0
tunnel-table = { disabled 0 0 32 0 0 0 0 0 0 0 0 }
local-sw-addr = 00
arp-addr = 00

super> set out-timeout = 20
super> write
CARD/10 written
```

Descriptions of HIPPI parameters

Here are descriptions of the HIPPI parameters listed above and the options for each:

`debug-level = 1`

- 0–3, number of messages sent to logger

`ready-count = 32`

- 1–63, HIPPI ready count

`testmode = no-mode`

Settings for test mode:

no-mode: no test running, default
hippi-source: sourcing HIPPI data
loopback: loopback, a single board mimics a cable
switch-test: switch test
agency: agency test mode
abort: test aborted HIPPI connection
ip-packet: spit out one IP packet over HIPPI
immunity: like agency but with error checking

`test-duration = 999999`

- test duration in seconds, 0 or non-zero

`test-pkt-size = 4`

- size of test packet in HIPPI bursts

`test-pattern = incremental`

Sets HIPPI test pattern, options are:

alt-walking - alternates walking 1 bit and walking 0 bits
all-ones - all 1 bits
repeat - repeat a pattern of 00000000 01010101 02020202 03030303
incremental - incremental pattern of 01010101 02020202 to ffffffff
alternate - alternate buffers of random pattern and aaaaaaaa/55555555

`test-check-data = 5`

Sets rate of test packets to be verified, every nth packet, 1 – 10

`max-Ifields = 300`

- currently not used

`out-timeout = 10`

- sets number of tenths of a second until output time-out, 0 or non-zero

`switch-timeout = 10`

- sets number of tenths of a second until switch time-out, 0 or non-zero

`IBM-HO-mode = disabled`

- enables/disables IBM H0 mode

`hold-connection = disabled`

`- enables/disables HIPPI hold connection`

When disabled, a new connection per IP packet is needed. When enabled, a connection is held until an error occurs.

`out-timeout-mode = 0`

Settings are 0 or 1:

0 = default time-out checks for output buffer fed to FIFO,

1 = default check for non-decreasing number of buffers queued for output to FIFO

`disable-raw = disabled`

- enables/disables HIPPI raw mode transfers, if disabled, only IP mode is valid

`mcast-addr = 03:00:0f:e0`

- sets switch address of the HIPPI multicast server, this is the I-field HIPPI uses to send multicast packets, a 4-byte hex field

`tunnel-table =`

- option not supported

`local-sw-addr = 00`

- sets HIPPI switch address when utilizing a HIPPI ARP server, a 1-byte hex field

`arp-addr = 00`

- HIPPI switch address of the ARP server, 1-byte hex field

3. Optional: Specify ICMP throttling

ICMP throttling settings are in the `icmp-throttling` section of the Card profile.

```
super> read card 10
CARD/10 read
super> list icmp
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
super>
```

Here is how to access the help information for the `echo-reply` field:

```
super> set echo ?
echo-reply:
 The number of ICMP ping responses generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

Change default echo reply and TTL settings with this series of commands:

```
super> set echo-reply = 4
super> set TTL-timeout = 12
super> write
CARD/10 written
```

You do not have to do a **write** until you have finished all changes in the Card profile. You get a warning message if you try to exit a profile without saving your changes.

4. Specify a different executable binary

Card-specific executables can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this HIPPI card only.

```
super> read card 10
card/10 read

super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you want to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_executable_for_hippi
super> write
CARD/10 written
```

5. Change default dump settings

Card-specific dump file names can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 10
card/10 read
super> list dump
config = 0
hw-table = < >
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD/10 written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

To save the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

```
# grreset <slot_number>
```

Optional: change HIPPI binaries – Load profile

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** HIPPI cards.

Here is the path, defaults are shown:

```
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the HIPPI card settings:

```
super> list hippi
type = hippi
rx-config = 0
rx-path = ""
tx-config = 0
tx-path = N/A
enable-boot-seq = on
mode = 0
iterations = 1
boot-seq-table = <{1/usr/libexec/portcards/xlxload.run N/A}{2 /usr/lib+

super> list boot
1 = { 1 /usr/libexec/portcards/xlxload.run N/A }
2 = { 2 /usr/libexec/portcards/runload.run N/A }
3 = { 3 /usr/libexec/portcards/hippi.run N/A }
```

At this level you see the file names of the specific HIPPI binaries that run by default in all HIPPI cards. The same fields are provided in the Card profile so you can run other executables in a specific HIPPI card.

```
super> list 1
index = 1
hw-type = hippi
rx-path = /usr/libexec/portcards/xlxload.run
tx-path = N/A

super> cd ..
super> list boot 2
index = 2
hw-type = hippi
rx-path = /usr/libexec/portcards/runload.run
tx-path = N/A

super> cd ..
super> list boot 3
```

```
index = 3
hw-type = hippi
rx-path = /usr/libexec/portcards/hippi.run
tx-path = N/A
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. You have the option to configure the card's boot sequence so that after booting, the card loads and runs diagnostics before it loads and runs the executable binaries. Set the `enable-boot-seq` field to `on` and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the "Management Commands and Tools" chapter in this manual for information.

Optional: change HIPPI dumps – Dump profile

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Defaults are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 /var/portcards/grdump 0} {rmb 20 /var/portc+
dump-vector-table = <{ 3 rmb "RMB default dump vectors" < { 1 SRAM 26214+
config-spontaneous = off
keep-count = 2
```

The `hw-table` field has settings to specify when dumps are taken and where dumps are stored. Here is the path to examine the HIPPI settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
etherne+ = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list hippi
media = hippi
config = 20
path = "/var/portcards/grdump 0"
vector-index = 0
super>
```

In the `config =` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

- The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.
- The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP/ written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

Use the **grwrite** command to save the `/etc` configuration directory:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

```
# grreset <slot_number>
```

Collect data via grdinfo

With a single command, **grdinfo** collects the sometimes lengthy output from the HIPPI **maint 132** command and compresses it in a log file. Refer to the “Management Commands and Tools” chapter in this manual for more information.

Monitoring HIPPI media cards

Use the **maint** commands to look at packet statistics on the SONET media card.

The **maint** commands operate on the control board and require the GR> prompt. Execute the **grrmb** command to switch prompts.

If you are not sure of the card's slot number, use the **grcard** command to view the location of installed cards.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

```
# grrmb
```

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the HIPPI media card you are working with. For example, if you are working with a card in slot 10, enter:

```
GR 66> port 10
```

This message is returned along with the changed prompt:

```
Current port card is 10
GR 10>
```

To leave the **maint** prompt, enter **`quit`**.

List of HIPPI maint commands

To obtain a list of **maint** commands, type:

```
GR 10> maint 1
GR 10> HIPPI maint commands:
 45:    List next hop data: [family]
 50:    Filtering filter list: [detail_level [ID]]
 51:    Filtering filter list: [detail_level [IF]]
 52:    Filtering action list: [detail_level [ID]]
 53:    Filtering action list: [detail_level [IF]]
 54:    Filtering binding list: [detail_level [ID]]
 55:    Filtering binding list: [detail_level [IF]]
 56:    Display filtering statistics: [IF#]
 57:    Reset filtering statistics: [IF#]
 58:    Show filter protocol statistics
        note, IF/ID may be '-1' to indicate all of the given item
              while detail level is 0|1|2.
  128 0: Print build date/time
  128 1: Print IEEE Address
  129 [n1[,n2]]:   Print HIPPI statistics
             n1 = first object number
             n2 = second object number
  130 [n [m]]:   Dump trace buffers & tables
       1, 2 = CP, MP trace buffers
```

```
          3 = Coprocessor status block
          4, 5 = MP Input, Output list pointers
          9 = switch/COM bus status structure
          10, 11 = hippiin, hippiout structures
          13 = IP routing statistics
          14 = Hardware control counters
          15 = Hardware control/status registers
          16 = Switch control registers
          17, 18 = HIPPI Destination, Source control registers
     131 <n>:  Set test mode <n>:
        3:   Switch core test
        4:   Agency certification noise generator
        5 <ifield>:  HIPPI aborted connection test
        6 <IPaddr> [length] [times] [Ifield] Send IP test packets
     132:   Dump trace buffers symbolically
     133:   Print IP statistics
     134 0|1  Clear/Set HIPPI loopback mode
     135 <level> Enable debug printouts
     136:   PANIC
     137:   Print the dump vector
     138:   Snapshot CPU registers
     139 <flags> [<ern1> [<ern2>] ] error message print/panic control
     140 <rate> <num> printf rate test
     141  print switch error counts
     151 HIPPI tunnel statistics
     152 {17|19} HIPPI tunnel loopback on|off
     153 HIPPI tunnel debug on|off
     154 HIPPI tunnel reset (0=send|1=receive)
     155 Send a GRID ECHO message to port <n>
     156 Show ARP table entries
     157 Print error message counters
     158 Set local HIPPI switch logical address
     200 (1|2|3) <n> 1-change forward loop, 2-change readies,
           3-change cpiloop_limit
```

Print IP statistics

Enter: `maint 133`

```
GR 03> HIPPI IP statistics:
13894 total HIPPI IP connections
13894 total packets received
15 packets forwarded
1380 packets not forwardable
Interface IP statistics:
ipstat[0].dropped = 13879
ipstat[0].forwarded = 15
ICMP statistics:
icmpstat[0].echo_req_returned = 13894
12482 total packets transmitted
```

Print IEEE address

Enter : `maint 128`

```
GR 03> maint 128
GR 03> IEEE Address = 00.C0.80.00.00.40 (1 address)
```

Dump trace buffers

Enter: `maint 130 16`

```
GR 03> maint 130  16
GR 03> Switch control registers (sdc2.h)
        0x008a0000:     0000000c 00000004 00000009 00000041
        0x008a0010:     00918000 1c000096 0060085d 00600000
        0x008a0020:     00600010 00600028 00600000 006000c8
        0x008a0030:     0060007d 006000c3 006000cf 0060080f
```

Print IP routing statistics

Enter: `maint 130 13`

```
GR 03> maint 130 13
GR 03> IP routing statistics (hippi_ip.h)
        0x00a08105:     00000000 00000005 00003646 00003646
        0x00a08115:     0000000f 00000000 00000000 00000000
```

Dump trace buffers symbolically

Enter: `maint 132`

```
GR 3> maint 132
GR 3>
GR 3> 0.0 CP 0002 f820501e 0090c070 Switch Output DMA start
  72549.3 CP 0001 40009c00 00000000 HIPPI Input DMA complete
     42.0 CP 0001 b801a01f 00000000 HIPPI Input DMA complete
    176.3 CP 0002 f801a01e 0090c076 Switch Output DMA start
     13.2 CP 0003 f802b01e 00000000 Switch Input DMA complete
     72.3 CP 0004 4004b000 00000000 HIPPI Output DMA start
     32.5 CP 0004 b802b01f 00000000 HIPPI Output DMA start
 926607.6 CP 0003 f8273c1f 0090c07c Switch Input DMA complete
     66.6 CP 0004 4020dc00 00000000 HIPPI Output DMA start
     37.7 CP 0004 b8273c1f 00000000 HIPPI Output DMA start
    651.8 CP 0001 40232400 0090c082 HIPPI Input DMA complete
     67.6 CP 0001 b8256c1f 00000000 HIPPI Input DMA complete
     98.8 CP 0002 f8256c1e 00000000 Switch Output DMA start
  32141.3 CP 0001 40073c00 00000000 HIPPI Input DMA complete
     35.1 CP 0001 b825481f 0090c088 HIPPI Input DMA complete
    170.4 CP 0002 f825481e 00000000 Switch Output DMA start
     13.1 CP 0003 f802b41e 00000000 Switch Input DMA complete
     71.9 CP 0004 4004b400 0090c08e HIPPI Output DMA start
     47.8 CP 0004 b802b41f 00000000 HIPPI Output DMA start
 966275.3 CP 0003 f827401f 00000000 Switch Input DMA complete
                         •
                         •
                         •
     11.6 MP 0019 00012390 00000020 Accepting connection
      8.7 CP 0001 b8071c1f 00000000 HIPPI Input DMA complete
     47.6 MP 006a c9010289 00000106 Sending ARP request
      9.5 MP 000c 00000106 00000000 Connecting to IP next hop
      0.3 MP 00a4 b802d81f c9010289 fast_output succeeded
 756166.3 MP 0044 00000001 00000002 Maint command 132
```

Print switch error counts

Enter: `maint 141`

```
GR 10> maint 141
GR 10> Backplane Switch Statistics:
0 rejects
0 transmit data errors
0 transmit FIFO data errors
0 transmit internal errors
0 total transmit errors
0 receive coding errors
0 receive disparity errors
0 receiv errors
0 receive checksum errors
0 total receive errors
COM Bus Statistics:
0 receive parity errors
0 receive timeouts
0 skipped messages
0 receive format errors
0 reads past EOM
0 premature EOMs
1833 messages received
```

Show ARP table entries with maint 156

```
GR 10> maint 156
                   ARP entries
         IP                 MAC         TTL  I-field   flags
------------------------------------------------------------
   203.003.011.158   00 00 00 00 00 00  600  03000fc0  permanent
   201.001.002.130   00 00 00 00 00 00  600  03000105  permanent
   201.001.001.134   00 00 00 00 00 00  600  03000101  permanent
   201.001.001.136   00 00 00 00 00 00  600  03000100  permanent
   201.001.002.137   00 00 00 00 00 00  600  00000106  permanent
```

Use grarp -a to look at ARP table

```
# grarp -a
ga010   (15): 211.10.11.134 at VPI=0, VCI=100 permanent
gh050   (25): 201.1.100.130 at 03000110 permanent
gh0d0   (34): 201.1.2.130 at 03000105 permanent
gh0d0   (34): 201.1.1.134 at 03000101 permanent
gh0d0   (34): 201.1.1.136 at 03000100 permanent
gh0d0   (34): 201.1.2.137 at 00000106 permanent
```

HSSI Configuration

8

Chapter 8 describes the implementation of HSSI on the GRF and other features supported on the HSSI media card. It includes the information needed to configure HSSI interfaces and parameters in the `/etc/grifconfig.conf` file and CLI profiles.

The chapter also provides information needed to configure framing protocols on the HSSI media card. Three framing protocols are supported over HSSI: Frame Relay, Point-to-Point Protocol (PPP), and HDLC.

The HSSI media card provides two independent physical interfaces. Both full-duplex interfaces are capable of transferring data at a rate of 52 megabits per second in each direction.

Chapter 8 covers these topics:

Profiles use `hssi` as the name of the HSSI media card.

HSSI (High Speed Serial Interface) implementation

The GRF HSSI implementation is compliant with the *HSSI Design Specification* written by John T. Chapman and Mitri Halabi, revision 2.11, dated March 16, 1990, and Addendum Issue #1, dated January 23, 1991.

HSSI is currently being ratified by the American Standards Institute. The physical layer specification will be EIA/TIA-613 and the electrical layer specification will be EIA/TIA-612.

The GRF HSSI media card provides two full duplex attachments. The CCITT-standard interfaces support up to 52 megabits per second performance per attachment. HSSI media card software supports Frame Relay, Cisco HDLC, and PPP protocols.

HSSI on the GRF

This section describes features of the HSSI media card implemented on the GRF.

Physical interfaces

A HSSI media card supports two physical interfaces (connectors).

As shown in Figure 8-1, a physical interface supports either 1 or 128 logical interfaces, depending upon which protocol is running on the HSSI card.

HSSI card:	**Frame Relay (256 / card)**	**PPP**	**Cisco HDLC**
Physical interface 0 (top)	128 logical interfaces *Numbered 0 – 7f*	1 logical interface *Numbered 0*	1 logical interface *Numbered 0*
Physical interface 1 (bottom)	128 logical interfaces *Numbered 80 – ff*	1 logical interface *Numbered 1*	1 logical interface *Numbered 1*

(g0053)

Figure 8-1. Logical interfaces supported per HSSI physical interface

Logical interfaces

A logical interface is configured by its entry in the `grifconfig.conf` file where it is assigned an IP address and netmask. A logical interface is uniquely identified by its HSSI interface name.

A logical interface is configured by its entry in the `/etc/grifconfig.conf` file where it is uniquely identified by a HSSI interface name (gs0yx) and is assigned an IP address and netmask.

The number of logical interfaces configurable on the HSSI media card depends upon which protocol is running.

A HSSI card that is to run the PPP or HDLC protocol requires two entries into the `/etc/grifconfig.conf` file because these protocols can support one logical interface on each of the physical interfaces.

A HSSI card that is to run Frame Relay will have as many as 256 entries since Frame Relay supports 128 logical interfaces per physical interface.

Framing protocols supported

The HSSI card runs the same protocol on both interfaces. If you change the protocol name in the Card profile, the old link is lost when you do a **write** command. The new protocol will be run after you reset the HSSI card.

Frame Relay

Frame Relay services provide a subset of the Data Link Layer and Physical Layer services, supporting the IETF encapsulation protocol and encapsulation of ARP frames.

The HSSI interface provides a User-to-Network-Interface (UNI) interface (DTE functionality), with an initial capacity of 256 logical interfaces per media card.

The Frame Relay MTU is set at 4352 bytes.

For interoperability, the following vendor documents are primary guides for defining the Frame Relay protocol:

- Frame Relay Physical Layer and Link Layer
  (including the subset of ANSI T1.602 LAPD protocol), documented in the US Sprint *Frame Relay Service Interface Specification* (Document #5136.03)
- the ANSI local management protocol developed and approved by ANSI,
  part of *T1.617, Annex-D*
- the CCITT local in-channel signaling protocol,
  part of *Q.933 ANNEX-A*

High-level Data Link Control protocol (HDLC)

Cisco HDLC is the name given to Cisco’s default protocol over HSSI interfaces. Proper operation of this protocol is verified through interoperability testing done using a GRF connected to a Cisco 7000 router.

The default HDLC MTU is 4352 bytes, it can be changed in the `grifconfig.conf` file.

Point-to-Point Protocol (PPP)

The Point-to-Point Protocol (PPP) implementation conforms to IETF RFCs 1661 and RFC 1662.

This release supports the following standard PPP options:

–	maximum receive unit	(LCP option 1)
–	quality protocol	(LCP option 4)
–	magic number	(LCP option 5)
–	IP address	(IPCP option 3)

The default PPP MTU is 1500 bytes, it can be changed in the `grifconfig.conf` file.

Note: The current implementation supports link quality monitoring, but does not yet support a link quality policy to take action when the link quality is inadequate.

IS-IS protocol support

IS-IS is a link state interior gateway protocol (IGP) originally developed for routing ISO/CLNP (International Organization for Standardization/Connectionless Network Protocol) packets. In ISO terminology, a router is referred to as an "intermediate system" (IS). IS-IS intra-domain routing is organized hierarchically so that a large domain may be administratively divided into smaller areas using level 1 intermediate systems within areas and level 2 intermediate systems between areas.

The GRF HSSI card supports IS-IS over HSSI Frame Relay.

This example shows HSSI interface `gs030` configured for IS-IS in the GateD IS-IS statement:

```
isis yes {
      area "49000080";
      systemid "326032603260";
      interface "gs030" metric 10 priority 60;
} ;
```

An ISO address must also be assigned to the HSSI logical interface in `/etc/grifconfig.conf`. This is in addition to the entry for the IP address also assigned in that file. Refer to the *Introduction to IS-IS* chapter for more information.

Here is an example of HSSI IP and IS-IS entries in `/etc/grifconfig.conf`:

```
#name   address         netwmask       broad_dest arguments
gs030  xxx.xxx.xxx.xxx 255.255.255.0        -     mtu 4352
#interface_name <iso_address> <iso_area> - iso
gs030 49.0000.80.3260.3260.3260.00 49.0000.80  -   iso
```

Large route table support

The HSSI media card supports a route table with 150K entries. The card has the 4MB of memory required for large route tables and also has the /Q level of hardware support for expanded route table look up.

ICMP throttling

The Internet Control Message Protocol (ICMP) is a message control and error-reporting protocol between a host and a gateway to the Internet. ICMP uses IP datagrams, and the messages are processed by the TCP/IP software. ICMP throttling is a way of limiting the number of messages generated per GRF card.

You can specify how many of several types of ICMP messages can be generated by the HSSI media card per one-tenth second. These are the message types:

– number of replies to echo requests
– number of "cannot deliver packet" replies (unreachable)
– redirect messages, number is not limited
– number of time-to-live replies
– number of parameter problem (packet discard) messages

- number of time of day time stamp replies to send

Specify ICMP throttling parameters in the Card profile.

On-the-fly PVC configuration

Frame Relay supports on-the-fly configuration of links and PVCs without requiring the media card to be reset. The **grfr** command has options to add and delete, enable and disable, and modify links and PVCs.

Please refer to the “Configuring Frame Relay” chapter for more information.

Selective packet discard

Selective packet discard can be enabled on the HSSI card to ensure that dynamic routing packets are transmitted on the media in the presence of a sustained high volume of data packets. During high traffic volumes, data packets are discarded in a rate that favors dynamic routing packets.

Packet discard is regulated by reserving buffers for dynamic routing packets. This gives the operator complete control over the point at which congestion management begins to discard data packets. A user-configured threshold defines the percentage of buffers to reserve for dynamic routing packets.

When the threshold is set to zero, no buffers are reserved for dynamic routing packets and dynamic routing packet discard is disabled. In this case, dynamic routing packets and data packets are treated identically.

When the threshold is set to 100, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the threshold of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the number of free transmit buffers is less than the user-defined threshold of buffers required to be reserved for dynamic routing packets. When the number of free buffers used for switch receive/media transmit falls below the congestion threshold, non-dynamic routing packets are discarded until the congestion condition clears. Because the congestion condition is updated thousands of times per second and busy buffers are rapidly transmitted and returned as free buffers, a congested state ends rapidly after its onset. This prevents prolonged discard of non-dynamic routing packets and ensures the transmission of dynamic routing packets even during periods of heavy network load.

The discard mechanism applies only to the transmit side of the media card, and has no impact on packets received from the media. There is no analogous treatment of packets received from the media. The discard threshold is set to zero by default, and is therefore disabled by default.

The threshold value is unique per media card in the chassis, and is set at the Card profile in the CLI. Ascend recommends the threshold value be set low, to a small value that maximizes the benefit for dynamic routing packets and minimizes the impact on data packets. As the number reserved for dynamic routing packets increases, the number of buffers available for data traffic decreases and dynamic routing packets are a small percentage of all packets when the card is congested, Practice has shown it unnecessary to set the threshold above single digits as it is unlikely that dynamic routing packets account for more than a few percent of all packets.

Refer to the "Setting selective packet discard threshold" section on later in this chapter.

Checking results

Examine GateD log files to determine the number of dynamic routing packets transmitted and their timestamps. A little arithmetic using the timestamps in the log files for packets transmitted to a neighbor (remember this is a transmit-only feature) should indicate the number of dynamic routing updates per unit time. Compare this number to the cumulative packet counters for switch receive over the same unit of time and you should arrive at the percentage of all transmit packets that are dynamic routing packets. Compare the average number over a few minutes to the number in a worst-case condition during bursts of dynamic routing packets based on periodic updates, and then select a percentage that balances the two.

Precedence handling

Precedence handling prioritizes delivery of dynamic routing update packets, even when the transmitting media card is congested. To ensure that dynamic routing update packets and other high priority packets are not dropped, the GRF uses precedence features to avoid this instability:

The GRF dynamic routing agent sets a precedence value in the internal packet header of the dynamic routing update packets it generates, which communicates to the media card a high-priority status for the packet.

The media card maintains a user-configurable threshold of transmit buffers that always remain available for high-priority traffic, ensuring that dynamic routing update packets are forwarded during congested conditions.

Precedence field

With selective packet discard enabled, the available buffer pool is managed as two pools, one for those with the "precedence field" set (high priority) and one for low priority data. Therefore, as the packets are taken off the switch, the buffer pools can be set up so that high priority packets will always find a buffer available, and the low priority packets will be dropped.

The precedence field is set in the IP packet header in one of two ways:

- by GateD on dynamic routing packets
- by filters configured to set this field on incoming data that matches any filter definition

Most dynamic routing packets sourced by the GRF have the precedence field set. This results in priority handling on the outbound (transmit) side of the media card in that a buffer is always made available for these packets as the data is read off the switch or communications bus. The media card starts discarding "low priority" packets before it completely runs out of buffers.

Controlled-load (class filtering)

Controlled-Load is supported on GRF media cards that support Selective Packet Discard, this includes the HSSI media card.

The GRF delivers Controlled-Load service to a specific flow by marking its packets precedence field to prevent Selective Packet Discard (SPD). The marking mechanism uses filters to identify the packets belonging to the class of applications for which resources are reserved. Class filters are manually configured by adding them to `/etc/filterd.conf`.

Controlled-Load protects packets that match the filter from being lost. Packets that match the filter are marked so they will not be dropped by SPD. SPD drops packets that are not marked when the number of free buffers gets too low. Dynamic routing packet precedence fields are marked by GateD. The class filter is another way of setting the same precedence bit in the IP packet header. Refer to the *Integrated Services: Controlled-Load* chapter in this manual for information about constructing class filters.

ATMP

The HSSI card supports the Ascend Tunnel Management Protocol (ATMP). ATMP is a layer 3 UDP/IP-based protocol that provides a cross-WAN (Internet or other) tunnel mechanism using standard Generic Routing Encapsulation between two Ascend units. ATMP is described in RFC 2107.

The ATMP tunnel protocol creates and tears down the tunnel between a foreign agent and a GRF home agent. The GRF connection to a home network is made across a PVC from a HSSI card. The home network router connects to the GRF ATM PVC through an ATM VC. The ATM circuits are created and assigned ATMP parameters in `/etc/gratm.conf`. Please refer to the "ATMP Configuration Guide" chapter for information about ATMP functions and configuration.

Null modem cabling

When a null-modem cable is used, each DTE must be set to enable internal clock generation. The clock value is shipped preset to 0, and needs to be changed if using a null-modem cable. The change is done by setting the `source-clock =` parameter at the Card profile, in the `ports` / `hssi` field. Go to the ports 0 or 1 section and bring up the HSSI fields. Here is the path:

```
super> read card 8
CARD/8 read
super> list card 8
```

Now list the specific HSSI port you want to set, 0 or 1, a short cut is used:

```
super> list ports 0 hssi
source-clock = 0
CRC-type = 16-bit
```

The `source-clock = 0` default is shown, change the setting to 1 to put internal clock generation on:

```
super> set source-clock = 1
super> write
CARD/8 written
```

Looking at the HSSI card

The GRF HSSI media card provides two full-duplex attachments and requires a pair of copper cables/connector ends as described in the *HSSI High Speed Serial Interface Design Specification* (March 1990).

Figure 8-2 shows the HSSI faceplate and LEDs. At the top of the HSSI face plate are five LEDs that indicate card status. Each HSSI interface has two sets of LEDs that indicate link and packet information. Each interface has a connector for attaching an encryption modem.

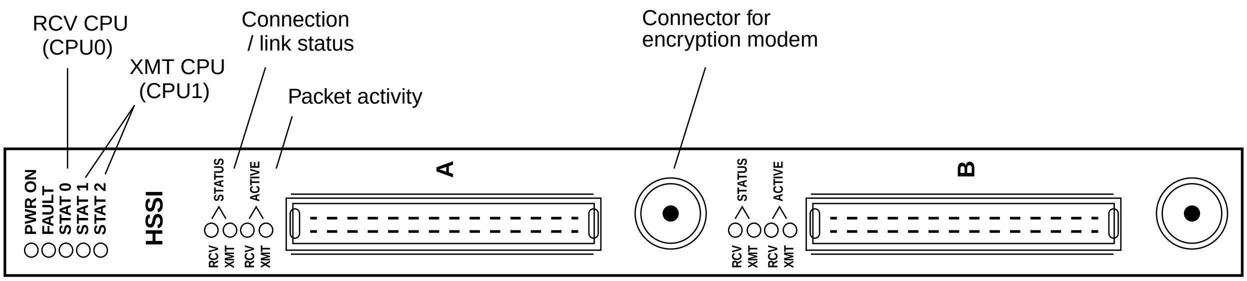


Figure 8-2. HSSI media card faceplate and LEDs

LEDs on the faceplate

Refer to Table 8-1 for a description of HSSI card LEDs.

Table 8-1. HSSI media card LEDs

LED	Description
Power	This green LED is on when GRF power is on.
Fault	This amber LED turns on and remains on if an error condition is detected. The Fault and STAT 0 LEDs alternate during self-test and while the HSSI card is loading. If the HSSI card is dumping, these two flash in unison.
STAT 0 STAT 1	During normal running time, these green LEDs blink together in a heartbeat pattern, one for each CPU on the card.
STAT 2	This green LED is inactive during normal running time.
RCV / XMT Status	These green LEDs indicate the status or viability of the HSSI connection for interface A or interface B.
RCV / XMT Active	These green LEDs indicate the frequency of packet traffic across an interface.

Configuration file and profile overview

These are the steps to configure HSSI interfaces and protocols:

1 Assign IP address to each logical interface
Edit `/etc/grifconfig.conf` to assign an IP address for each logical HSSI interface.

2 Specify HSSI card parameters in the Card profile:

- specify a framing protocol
- specify internal clock generation
- specify cyclic redundancy check (CRC)
- specify HDLC settings
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard threshold
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

3 Configure the framing protocol
Cisco HDLC - Steps 1 and 2 complete the configuration, reset the card.
Frame Relay - After steps 1 and 2, set Frame Relay and PVC parameters in the `/etc/grfr.conf` configuration file, reset the card.
Point-to-Point Protocol - After steps 1 and 2, set PPP parameters in the `/etc/grppp.conf` configuration file, reset the card

These next steps are optional, they describe tasks that are performed infrequently:

4 Change Load profile (optional).
Global executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every HSSI card.
If you want to change the run-time code in one HSSI card, make the change in the Card profile, in the `load` section.

5 Change Dump profile (optional).
Global dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.
To change dump settings for one HSSI card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1. To save files in the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

2. In the command-line interface, use **set** and **write** commands to save a profile.

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configuring a HSSI interface

This section describes how to configure a HSSI interface in the `/etc/grifconfig.conf` file. Use a UNIX editor to make entries in `/etc/grifconfig.conf`.

A HSSI card that is to run the PPP or HDLC protocol requires two entries into the `grifconfig.conf` file since these protocols support one logical interface on each of the physical interfaces.

A HSSI card that is to run Frame Relay will have as many as 256 entries since Frame Relay supports 128 logical interfaces per physical interface.

Edit `grifconfig.conf` to identify each logical HSSI interface by assigning:

- interface name, `gs0yz` (names are always lower case)
- Internet address
- netmask
- broadcast/destination address (optional)
- arguments field (optional)

The format for an entry in the `grifconfig.conf` file is:

```
name   address    netmask      broad_dest    arguments
```

Interface name *gs0yz*
Each logical GRF interface is given an interface name `ga0yz` where:

- the “gs” prefix indicates a HSSI interface
- the chassis number is always “0”
- “y” is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
- “z” is the logical interface number in hex

Note: Interface names are case sensitive. Always use lower case letters when defining interface names.

Address
Enter the IP or ISO address to be assigned to this interface

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured.

Broadcast address
Use the broadcast address when you wish to specify other than all 1s as the broadcast address.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO addess is being added to an interface's IP address. Specify the MTU value as `mtu` *xyz*. Leave the `arguments` field blank if you are not using it.

Example

The entry assigns an IP address for logical interface 0 on the HSSI card in slot 6. If needed, a dash is used as a placeholder for the broadcast address:

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
gs060 192.0.2.1 255.255.255.0 192.0.2.255
```

If an interface is nonbroadcast (NBMA), do not include a destination address in its `/etc/grifconfig.conf` entry.

Save the /etc file

Save the file with the editor. Then, use **grwrite** to write the file to the `/etc` configuration directory:

```
# grwrite -v
```

Check contents of grifconfig.conf file

After you save the `/etc` directory and reset the media card, use **netstat -in** to display the contents of the `/etc/grifconfig.conf` file and verify that the logical interface is configured with the correct IP address.

Here is the output from a **netstat** command looking at the HSSI interfaces:

```
# netstat -in | grep gs
gs050   4352  <link10>                     35972    0   53640    0    0
gs050   4352  207.1.11   207.1.11.156    35972    0   53640    0    0
gs0b0   1500  <link26>                         0    0       0    0    0
gs0b1   1500  <link27>                       100    0       0    0    0
gs0b1   1500  207.1.12   207.1.12.156      100    0       0    0    0
```

Please refer to the **netstat** man page for information about other **netstat** options.

Check system-level IP configuration

The UNIX **ifconfig** ***interface*** command returns system level information for the specified interface name, here is the interface for logical interface 0 (`gs0b1`):

```
# ifconfig gs0b1
gs0b1: grithssi flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 1500
        inet 207.1.12.156 netmask 0xffffff00 broadcast 207.1.12.255
```

Setting parameters in the Card profile

This section describes how to verify and/or change HSSI parameters in the Card profile. The parameters are presented in this order:

- set framing protocol: `cisco-hdlc`, `ppp`, `frame-relay` (default is Frame Relay)
- HSSI hardware settings:
  specify source clock: 0 and 1, with 1 equal to null modem) (default is 0)
  specify CRC type: the CRC type to match connecting endpoint,
  options are 16-bit, 32-bit, and 0 (default is 16-bit)
- OPTIONAL: specify Cisco HDLC settings
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard threshold
- OPTIONAL: set card-specific load variables
- OPTIONAL: set card-specific dump variables

1. Set framing protocol

At the Card profile top level, you can set the framing protocol. Values are:

- `Cisco-HDLC`
- `PPP`
- `Frame-Relay`

When you read and list the Card profile for this HSSI mecia card, you will see that media card type, `hssi`, is automatically read into the read-only `media-type` field. Other values shown are defaults.

By default, the `hssi-frame-protocol` field is set to `Frame-Relay`. If the card is to run another protocol, you must change it to `PPP` or `Cisco-HDLC`.

```
super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
media-type = hssi
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = <{ 0{off on 10 3} {single off} {"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off}
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

To change the framing protocol to PPP and save your change:

```
super> set hssi-frame-protocol = ppp
super> write
CARD/8 written
```

You do not have to do a **write** until you have finished all changes in the Card profile. You get a warning message if you try to exit a profile without saving your changes.

2. Set source clock and CRC

You specify clock and CRC settings for HSSI interfaces 0 and 1 in the `ports` section:

```
super> list ports 1
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 }
hssi = { 0 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disab+
```

Go into the `hssi` field to set source clock and CRC values.

- The clock value is preset to 0, and needs to be changed if using a null-modem cable.
- The CRC value is preset to `16-bit`, other settings are `32-bit` and `no-CRC`.
  Set the CRC type to match the device on the other end of the wire. A 32-bit CRC is generally recommended when the MTU is over 4096. The Cisco default is 16-bit CRC.

```
super> list hssi
source-clock = 0
CRC-type = 16-bit

super> set source-clock = 1
super> set CRC-type = no-CRC
super> write
CARD/8 written
```

Here is a shortcut you also could use to get to HSSI settings in interface 1 from the top level of the Card profile:

```
super> list ports 1 hssi
```

Tip: A quick way to set only the CRC on interface 0 in slot 8:

```
super> read card 8
CARD/8 read
super> set port 0 hssi crc-type = 32-bit
super> write
CARD/8 written
```

Tip: Use `set <field_name>?` *to display the available values.*

```
set CRC-type?
CRC-type:
 The type of CRC used: 16-bit, 32-bit, or none.
 Enumerated field, values:
 no-CRC:  Don't use a CRC
 16-bit:  Use 16-bit CRC ( the usual value with Frame Relay )
 32-bit:  Use 32-bit CRC
```

3. Specify Cisco HDLC settings if running HDLC

If the card is to run HDLC, verify the HDLC settings are correct. The Cisco HDLC parameters are located in the `ports 0` or `ports 1` section of the Card profile. If you are at the `ports` level, use **cd ..** to go "up" a level so you can access the HDLC fields:

```
super> cd ..

super> list ports 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 2 sdh recovered-clock 0 200 }
hssi = { 1 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disab+

super> list cisco
debug = off
keepalive-enabled = on
keepalive-interval = 10
keepalive-error-thresh = 3
```

The Cisco HDLC settings are:

- Debug turns on diagnostic messages about the Cisco-HDLC keepalive activity, messages are written to the `gr.console` log. The default is off, no diagnostic messages are collected.
- Keepalive activity can be turned off, the default is on.
- The default keepalive interval setting specifies how often the HSSI interface sends keepalive messages, the default is every 10 seconds. Remember to specify the `keepalive-interval` setting in milliseconds.
- The keepalive error threshold specifies how many keepalive messages can go unanswered before the HSSI interface marks the connection as down, three is the default.

If necessary, change the default keepalive settings and save the changes:

```
super> set keepalive-interval = 25
super> set keepalive-error-thresh = 6
super> write
CARD/8 written
```

4. Specify ICMP throttling

You can specify ICMP throttling parameters for this HSSI card in the `icmp-throttling =` field. Refer to Chapter 1 for an explanation of each field or do a `set <field-name>?` for a brief description. Here is how to find out about one parameter,the `echo-reply` field:

```
super> set echo ?
echo-reply:
 The number of ICMP ping responses generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

Read and list the Card profile for the media card you are configuring, then list the ICMP section:

```
super> read card 8
CARD/8 read
super> list

super> list icmp
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

Change the default ICMP throttling setting with this series of commands:

```
super> set echo-reply = 8
super> set TTL-timeout = 12
super> write
CARD/8 written
```

5. Specify selective packet discard threshold

Specify a SPD threshold for this HSSI card in the `spd-tx-thresh` field. This field is contained in the `config` section of the Card profile.

```
super> read card 8
CARD/8 read
super> list
card-num* = 8
media-type = hssi
debug-level = 0
hssi-frame-protocol = PPP
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < {0{ off on 10 3} {single off}{"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 8 10 2147483647 12 10 10 }

super> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0

super> set spd-tx-thresh = 5
super> write
CARD/8 written
```

A discussion of how to determine an SPD threshold is provided in the “Selective packet discard” section earlier in this chapter. The HSSI **maint 4** command reports discard counts.

On reboot, the congestion threshold message should indicate the new setting, as shown below:

```
[2] [TX] Current congestion thresholds, out of 256 available buffers:
[2] [TX] Congestion:  17 (5%) [2] [TX]        Overshoot:  8
```

SPD statistics

Use the **maint 4** command to look at the number of packets each transmit side drops.

```
[RX] Port 0:
[RX] Odd Length TX Packets:  16924
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:       83
[RX]    TX Dropped Ckt Down:   0
[RX] Port 1:
[RX] Odd Length TX Packets:  13816
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Dropped Ckt Down:   0
```

6. Specify a different executable binary

Card-specific executable binaries can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this HSSI card only.

```
super> read card 8
card/8 read

super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you want to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_executable_for_hssi
super> write
CARD/8 written
```

7. Change default dump settings

Card-specific dump file names can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 8
card/8 read

super> list dump
config = 0
hw-table = < >
```

```
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex. Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD8/ written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

To save the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

```
# grreset <slot_number>
```

Optional: change HSSI binaries – Load profile

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** HSSI cards.

Here is the path, defaults are shown:

```
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the HSSI card settings:

```
super> list hssi
type = hssi
rx-config = 0
rx-path = /usr/libexec/portcards/hssi_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/hssi_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

To execute different run-time code on the receive side of the HSSI card, replace **`/usr/libexec/portcards/hssi_rx.run`** with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/newhssi_rx.run
super> write
LOAD written
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. In the Load profile, you have the option to change the boot sequence for all the cards of one type of media so that, after booting, those cards load and run diagnostics before they load and run the executable binaries. Set the `enable-boot-seq` field to on and use **write** to save the change:

binaries. Set the `enable-boot-seq` field to on and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the "Management Commands and Tools" chapter in this manual for information.

Optional: change HSSI dumps – Dump profile

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Defaults are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 /var/portcards/grdump 0} {rmb 20 /var/portc+
dump-vector-table = <{ 3 rmb "RMB default dump vectors" < { 1 SRAM 26214+
config-spontaneous = off
keep-count = 2
```

The `hw-table` field has settings for when dumps are taken and where dumps are stored. Here is the path to examine the HSSI settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list hw hssi
media = hssi
config = 20
path = /var/portcards/grdump
vector-index = 2
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP written
```

Dump vectors

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for all HSSI cards. These fields are read-only and cannot be changed.

Here is the path, first you **cd** .. up to the main level:

```
super> cd ..
super> list . d
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" <{1 "atm inst memory" 167+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" <{1 "fddi/Q CPU0 core memor+
7 = {7 hssi "HSSI default dump vectors" <{1 "hssi rx SRAM memory" 209715+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" <{1 "Ethernet rx SRAM+
9 = {9 dev1 "DEV1 default dump vectors" <{1 "dev1 rx SRAM memory" 209715+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" <{1 "ATM-12 SDRAM +
11 = {11 sonet-v1 "SONET default dump vectors" <{1 "SONET rx SRAM memory+
```

This sequence shows a portion of the areas in the HSSI card that are dumped:

```
super> list 7
index = 7
hw-type = hssi
description = "HSSI default dump vectors"
segment-table =<{1 "hssi rx SRAM memory" 2097152 4194304}{2 "hssi share+

super> list s
1 = { 1 "hssi rx SRAM memory" 2097152 4194304 }
2 = { 2 "hssi shared SRAM memory" 131072 32768 }
3 = { 3 "hssi tx SRAM memory" 69206016 2097152 }

super> list 1
index = 1
description = "hssi rx SRAM memory"
start = 2097152
length = 4194304

super> cd ..
super> list s 2
index = 2
description = "hssi shared SRAM memory"
start = 131072
length = 32768
```

Configuring HDLC on HSSI

Setting up the HSSI card to run HDLC requires four configuration tasks:

- Specify logical interface in `/etc/grifconfig.conf.`
- Set framing protocol in Card profile to Cisco-HDLC.
- Check HSSI settings in Card profile and change them as needed.
- Check Cisco-HDLC settings in Card profile and change them as needed.

These tasks are described in the preceding configuration sections, “Configuring a HSSI interface” on page 8-11 and “Setting parameters in the Card profile” on page 8-13. Please use those sections to set up HDLC on the HSSI interface.

When you are finished, reset the card.

Configuring Frame Relay on HSSI

Setting up the HSSI card to run Frame Relay requires three configuration tasks:

– Specify logical interface in `/etc/grifconfig.conf.`

– Set framing protocol in Card profile to Frame-Relay.

– Check HSSI settings in Card profile and change them as needed.

These tasks are described in the preceding configuration sections, "Configuring a HSSI interface" on page 8-11 and "Setting parameters in the Card profile" on page 8-13. Please use those sections to set up HDLC on the HSSI interface.

When you are finished, reset the card.

What to do next...

Please use the "Configuring Frame Relay" chapter in this manual to configure PVCs on the HSSI interfaces you have created in `/etc/grifconfig.conf`. Frame Relay configuration is done in the `/etc/grfr.conf` file. A copy of the file is in Appendix A of the *GRF Reference Guide*.

Configuring PPP on HSSI

Setting up the HSSI card to run the Point to Point (PPP) requires four configuration tasks:

- Specify logical interface in `/etc/grifconfig.conf`.
- Make sure the framing protocol field in the Card profile is set to PPP.
- Check HSSI settings in Card profile and change them as needed.
- Create PPP interface in `/etc/grppp.conf`.

The first three tasks are described in the preceding configuration sections, "Configuring a HSSI interface" on page 8-11 and "Setting parameters in the Card profile" on page 8-13. Please use those sections to set up PPP on the HSSI interface.

Finally, assign PPP to an interface in the `/etc/grppp.conf` configuration file.

Configuring the PPP interface in grppp.conf

The fourth task is to create the PPP interface in `/etc/grppp.conf` and assign the required PPP parameters. This configuration step binds PPP to the HSSI interface.

Here are the steps:

1 Open the UNIX shell:

```
super> sh
# cd /etc
# vi grppp.conf
```

Use a UNIX editor to edit `/etc/grppp.conf`. A copy of the file is provided on page 8-28.

Note: To make immediate, temporary changes to the PPP configuration, use the **grppp** command, refer to the **grppp** man page for more information. Temporary settings done with **grppp** are lost when the HSSI card is reset or the GRF is rebooted. Make permanent changes in the configuration file.

2 Set up a PPP interface, this setting binds PPP to the interface.

In `/etc/grppp.conf`, a comment cannot be on the same line as an interface configuration. Keep comments separate, on their own line. A line may either be a configuration line or a comment line, not both.

Identify the interface using the `gs0yz` name:

```
# configure HSSI i/f in slot 6
interface gs060
   enable negotiation trace     #writes traces into /var/log/gr.console
   enable ipcp                  #allow IP traffic over PPP
   enable osinlcp               #allow osi traffic over PPP
```

The three "enable" parameters that follow the interface entry are frequently used. These are actually **grppp** commands.

Other **grppp** commands can be entered in the configuration file. Most of these commands override default values and should be used with caution. These are described in the next steps.

The function of each command is provided here. Refer to the **grppp** man page for more information about each.

3 Optional: Specify optional automata parameters.

`set maximum configuration request count = INTEGER`
- Sets number of unanswered configuration requests allowed (default is 10).

`set maximum failure count = INTEGER`
- Sets number of connection non-acknowledgments taken. (default is 5)

`set maximum terminate count = INTEGER`
- Limits number of termination requests sent. (default is 2)

`set restart timer interval = INTEGER`
- Times sending of configuration and termination requests. (default is 3000 milliseconds)

4 Optional: Specify Link Control Protocol (LCP) parameters.

`enable lcp magic number`
- Enabled only to detect looped-back networks.

`set lcp keepalive interval = INTEGER`
- Time allowed between packets, the default of 0 milliseconds disables keepalive feature.

`set lcp keepalive packet threshold = INTEGER`
- Limits number of echo packets unanswered before link is closed. (default is 5 packets)

`set lcp mru = INTEGER`
- Defines maximum packet size. (default is 1500 octets)

5 Optional: Set Link Quality Reporting (LQR) parameters .

`enable lqr`
- Turns on collection of link quality reporting statistics. (default is disabled)

`set lqr timer interval = INTEGER`
- Sets time period between LQR messages sent by one endpoint to peer, begins the exchange of statistics between endpoints, specified in 1/100 seconds. (default is 0)

6 After you have entered the appropriate parameters, save the file with the UNIX editor. Then use the **grwrite** command to write the file to the `/etc` directory:

```
# grwrite -v
```

Using grppp commands

Use the **grppp** status commands to display PPP objects and configuration values.

These are the **grppp** status commands:

```
show configuration
show negotiation trace status
show maximum configuration request count
show maximum failure count
show maximum terminate count
show restart timer interval
show lcp keepalive interval
show lcp keepalive packet threshold
show lcp mru
show lcp status
show lqr timer interval
show lqr status
show ipcp status
```

At the UNIX prompt you enter the **grppp** command and the prompt changes:

```
# grppp
>
```

At the > prompt enter interface and the interface name, the prompt changes again:

```
>interface gs060
gs060>
```

Commands are entered in lower case, short forms of words can be used. Use **quit** to exit the **grppp** prompt.

Looking at a PPP configuration

Here is the output from a **grppp show config** command:

```
# grppp
>interface gs060
gs060> show config
  General Configuration:
    Maximum configure request count: 10
    Maximum request failure count: 5
    Maximum terminate request count: 2
    Negotiation tracing is enabled
    Restart timer interval: 3000 milliseconds
  LCP Configuration:
    Magic number is disabled
    Initial MRU: 1500
    Keepalive interval: 0 milleseconds, disabled
    Keepalive packet threshold: 5
  LQR Configuration:
    LQR is disabled
    Timer interval: 0 milleseconds
  IPCP Configuration:
    enable IPCP
  OSINLCP Configuration:
```

```
    disable OSINLCP
gs060>
```

Here is the output from a **show lcp status** command:

```
gs060> show lcp st
  LCP Status:
    Bad addresses: 0
    Bad controls: 0
    Packets too long: 0
    Bad FCSs: 0
    Local MRU: 1500
    Remote MRU: 1500
  LCP Configuration:
    Magic number is disabled
    Initial MRU: 1500
    Keepalive interval: 0 milleseconds, disabled
    Keepalive packet threshold: 5
gs060>
```

Contents of grppp.conf file

Here is a copy of the `/etc/grppp.conf` file.

```
# Netstar $Id: grppp.conf,v 1.4 1997/03/25 16:54:45 suseela Exp $
#
# Template grppp.conf file.
#
#
# This file is used to set the initial configuration of PPP interfaces.
#
# When a media card configured for PPP is reset, grinchd executes grppp
# to process this file.   The following subset of grppp commands may be
# used in the grppp.conf file.  Most of these commands are used to
# overide default values, and should be used with caution.  Refer to
# the grppp man page for a full explanation of these commands.
#
#       interface INTERFACE_NAME
#       enable negotiation trace
#       set maximum configuration request count = INTEGER
#       set maximum failure count = INTEGER
#       set maximum terminate count = INTEGER
#       set restart timer interval = INTEGER
#       enable lcp magic number
#       set lcp keepalive interval = INTEGER
#       set lcp keepalive packet threshold = INTEGER
#       set lcp mru = INTEGER
#       enable lqr
#       set lqr timer interval = INTEGER
#       enable ipcp
#       enable osinlcp
#
# The example below shows the most commonly used grppp commands used in
# a grppp.conf file.

#
# Example Gigarouter PPP initial configuration
#
# interface gs0b0                       # Card 11, port 0
#    enable negotiation trace           # copy negotiaton traces to
#                                          #  /var/log/gr.console
#    enable ipcp                        # allow IP traffic over PPP
#
# interface gs0b1                       # Card 11, port 1
#    enable ipcp                        # allow IP traffic over PPP
#    enable osinlcp                     # allow osi traffic over PPP
#
```

Monitoring HSSI media cards

Use the **maint** commands to look at packet statistics on the HSSI media card.

The **maint** commands operate on the control board and require the GR> prompt. Execute the **grrmb** command to switch prompts.

If you are not sure of the card's slot number, use the **grcard** command to view the location of installed cards.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

```
# grrmb
```

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the HSSI media card you are working with. For example, if you are working with a card in slot 8, enter:

```
GR 66> port 8
```

This message is returned along with the changed prompt:

```
Current port card is 8
GR 8>
```

To leave the **maint** prompt, enter **quit**.

Display maint commands

To view the list of HSSI card **maint** commands, enter:

```
GR 8> maint 1
[RX]  1:     Display this screen of Options
[RX]  2:     Display Version Numbers
[RX]  3:     Display Configuration and Status
[RX]  4:     Display Media Statistics
[RX]  5:     Display SWITCH Statistics
[RX]  6:     Display Combus Statistics
[RX]  7:     Clear statistics counters (may mess up SNMP)
[RX]  8:     Display ARP Table
[RX]  9:     History trace on/off [ 0 | 1]
[RX]  10:    Display History Trace
[RX]  11:    Display IPC Stats
[RX]  12:    Display HW Registers
[RX]  16:    Display Multicast Routing Table
[RX]  22:    Display RX Packet-Per-Second Rates [# sec avg]
[RX]  30:    Switch Test: Clear Stats (but not setup)
[RX]  32:    Switch Test: Setup [ patt len slots... ]
[RX]  33:    Switch Test: Start [ slots...]
[RX]  34:    Switch Test: Stop [ slots...]
[RX]  35:    Switch Test: Status [ slots...]
[RX]  38:    Switch Test: Send One [ slots...]
```

```
[RX]  45:    List next hop data: [family]
[RX]  50:    Filtering filter list: [detail_level [ID]]
[RX]  51:    Filtering filter list: [detail_level [IF]]
[RX]  52:    Filtering action list: [detail_level [ID]]
[RX]  53:    Filtering action list: [detail_level [IF]]
[RX]  54:    Filtering binding list: [detail_level [ID]]
[RX]  55:    Filtering binding list: [detail_level [IF]]
[RX]  56:    Display filtering statistics: [IF#]
[RX]  57:    Reset filtering statistics: [IF#]
[RX]  58:    Show filter protocol statistics
[RX]         note, IF/ID may be '-1' to indicate all of the given item
[RX]               while detail level is 0|1|2.
[RX]  70:    Display ATMP Home Network table
[RX]  73:    Display Mobile Node Tree n
[RX]  80:    Frame Relay Arp Debug.
```

Read S/W and H/W revisions

Use **maint 2** to read the revision levels of the operating software and media card hardware.

```
GR 8> maint 2
[RX]
[RX]             HSSI Port Card Hardware and Software Revisions:
[RX]             ===============================================
[RX]
[RX] HW:
[RX]    Power-On Self-Test (POST) result code: 0x0.
[RX]    HSSI Media Board HW Rev: 0x8, with 4M Sram.
[RX]    HSSI Xilinx Version: 0x0.
[RX]    SDC Board HW Rev: 0xe  (SDC2).
[RX]            SDC2 Combus Xilinx version: 0x6.
[RX]            SDC2 Switch Transmit Xilinx version: 0x5.
[RX]            SDC2 Switch Receive Xilinx version: 0x0.
[RX]
[RX] SW:
[RX]  HSSI Code Version: A1_4_3, Compiled Mon Nov 17 18:35:22 CST 1997,
[RX]            in directory: /nit/A1_4_3/hssi/rx.
[RX] IF Library Version: 1.1.0.0, Compiled Mon Nov 17 18:29:41 CST 1997.
```

Configuration and status

Use **maint 3** to display current protocol configuration and status:

```
GR 8> maint 3
GR 8> [RX]
[RX] HSSI Configuration and Status.
[RX]    Framing Protocol: Frame Relay.
[RX]            Port 0 LMI Type:
[RX]            Port 1 LMI Type:
[RX]    Free Memory: 846720
[RX]    Line States:
[RX]            Port 0: Up.
[RX]            Port 1: Up.
```

Display media statistics

maint 4 returns statistics on the amount of data transferred and packets discarded:

```
GR 8> maint 4
GR 8> [RX]
[RX]
[RX]                 Media Statistics
[RX]
[RX] Port       Bytes           Packets          Errors       Discards
[RX] --------------------------------------------------------------
[RX]  0 000000000050979765  000000000000432042    000000000    000000000
[RX]  1 000000000066766723  000000000000144902    000000000    000000000
[RX]
[RX] Port 0
[RX]    RX CRC Errors:          0
[RX]    RX ABORT Errors:        0
[RX]    Discard No DLCI:        0
[RX]    Discard No Buffer:      0
[RX] Port 1
[RX]    RX CRC Errors:          0
[RX]    RX ABORT Errors:        0
[RX]    Discard No DLCI:        0
[RX]    Discard No Buffer:      0
[RX]
[RX] output:
[RX] Port       Bytes               Packets          Discards
[RX] ---------------------------------------------------------
[RX]  0 000000000001294184    000000000000233442    000000000
[RX]  1 000000000054470578    000000000000492390    000000000
[RX]
[RX] Port 0:
[RX] Odd Length TX Packets:  16924
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Dropped Ckt Down:   0
[RX] Port 1:
[RX] Odd Length TX Packets:  13816
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Dropped Ckt Down:   0
```

Display switch statistics:

Use **maint 5** to display statistics for the GRF switch:

```
GR 8> maint 5
GR 8> [RX]
[RX]
[RX]                    Switch Statistics
[RX] input:
[RX]         Bytes               Packets           Errors
[RX] ---------------------------------------------------------
[RX]  0000000001592268    000000000000016581    000000000
```

```
[RX]
[RX] output:
[RX]         Bytes              Packets          Errors
[RX] ---------------------------------------------------------
[RX]  00000000002918212    000000000000016581    000000000
[RX]
[RX] Switch Transmit Data Errors:             0
[RX] Switch Transmit Fifo Parity Errors:      0
[RX] Switch Transmit Internal Parity Errors:  0
[RX] Switch Transmit Connection Rejects:      0
[RX] Switch Receive Encoding Errors:          0
[RX] Switch Receive Running Disparity Errors: 0
[RX] Switch Receive Receiver Errors:          0
[RX] Switch Receive Running Checksum Errors:  0
```

Clear status info

Use **maint 7** to clear the collected statistics:

```
GR 8> maint 7
[RX] All Media Statistics Cleared.
[RX] All Switch Statistics Cleared.
[RX] All Combus Statistics Cleared.
```

Display PVC status

maint 8 provides status and information about each configured PVC:

```
GR 8> maint 8
[RX] Frame-Relay PVC Status:
[RX] ('*' = address obtained via inverse arp)
[RX] ('+' = Enabled for ISIS)
[RX] name              port  dlci  state  protocol address
[RX] ================= ====  ====  ====== ================
[RX] north_region       0    99    UP  222.222.60.60
[RX] name_1             0   100    UP  222.222.60.100
[RX] name_2             0   101    UP  222.222.60.101
[RX] name_3             0   102    UP  222.222.60.102
[RX] name_4             0   103    UP  222.222.60.103
[RX] name_5             0   199    UP  222.222.60.199
[RX] name_6             0   200    UP  222.222.61.1
[RX]                    0   201    UP  222.222.62.1
[RX]                    0   202    UP  222.222.63.1
[RX]                    0   203    UP  222.222.64.1
[RX] south_region       1   300    UP  222.222.180.108
[RX] name_1             1   301    UP  222.222.180.109
[RX] name_2             1   302   DOWN 222.222.180.110
[RX]                    1   303   DOWN 222.222.180.111
[RX]                    1   304   DOWN 222.222.180.112
[RX]                    1   400   DOWN 222.222.181.113
[RX]                    1   401   DOWN 222.222.182.114
[RX]                    1   402   DOWN 222.222.183.115
[RX]                    1   403   DOWN 222.222.184.116
[RX] name_3             1   404   DOWN 222.222.185.117
[RX] name_4             1   405    UP  222.222.180.118
[RX] name_5             1   600    UP  222.222.180.105
```

List next hop data

This **maint** command returns a list of the next hop data. You can also use the **grrt -S -p 1** command to obtain the same information, an example is at the bottom of this page.

```
maint 45
[RX] Location is: 0
[RX] Add:       48 Delete:       11 noNH:        0
[RX]    0: 206.146.160.133 (1   )   2:00  0:fc   RMS
[RX]    1: 0.0.0.0         (1   )   2:00  0:fc   MCAST
[RX]    2: 127.0.0.1       (1   )   2:00  0:00   RMS
[RX]    3: 0.0.0.0         (1   )   2:00  0:fc   UNREACH
[RX]    4: 0.0.0.0         (1   )   2:00  0:00   UNREACH
[RX]    5: 0.0.0.0         (1   )   2:00  0:00   MCAST
[RX]    6: 206.146.160.1   (1   )   2:00  0:fc   RMS
[RX]    7: 206.146.160.151 (1   )   2:00  0:fc   RMS
[RX]    8: 206.146.160.132 (1   )   2:00  0:fc   RMS
                 •
                 •
                 •
[RX]   30: 0.0.0.0         (12  )   2:00  3:05   BCAST
[RX]   31: 0.0.0.0         (12  )   2:00  3:05   LOCAL
[RX]   32: 0.0.0.0         (12  )   2:00  3:05   FWD
[RX]   33: 0.0.0.0         (9   )   2:00  3:07 * BCAST
[RX]   34: 0.0.0.0         (9   )   2:00  3:07 * LOCAL
[RX]   35: 10.20.2.237     (73  )   2:00  2:7f   FWD
[RX]   36: 10.20.1.182     (12  )   2:00  3:05   FWD
[RX] Location is: 1
[RX] Add:        0 Delete:        0 noNH:        0
```

These are the columns of interest:

```
[RX]   36: 10.20.1.182     (12  )   2:00  3:05   FWD
```

Entry #

Source address

Next hop index #

Slot, interface #
of destination media card

grrt next hop information

Here are a few lines of **grrt** output:

```
# grrt -S -p 1
default                          3    0.0.0.0         RMS       UNREACH
0.0.0.0         255.255.255.255  7    0.0.0.0         RMS       DROP
127.0.0.0       255.0.0.0        5    0.0.0.0         RMS       UNREACH
127.0.0.1       255.255.255.255  2    127.0.0.1       RMS       RMS
198.174.11.0    255.255.255.0    6    206.146.160.1   RMS       RMS
203.1.10.156    255.255.255.255  77   0.0.0.0         gf0d2     LOCAL
203.1.10.255    255.255.255.255  76   0.0.0.0         gf0d2     BCAST
203.3.10.0      255.255.255.0    68   0.0.0.0         gf081     FWD
203.3.10.156    255.255.255.255  67   0.0.0.0         gf081     LOCAL
203.3.10.255    255.255.255.255  66   0.0.0.0         gf081     BCAST
```

List of filters

maint 50 returns the list of filters by filter ID:

```
GR 8> maint 50
GR 8> filterID     type   status  access
       00000911   ctable (loaded)    0002
       00000912   ctable (loaded)    0004
       00000913   ctable (loaded)    0002
       00000918   ctable (loaded)    0002
```

Display filtering statistics

maint 56 returns a set of filtering statistics:

```
GR 8> maint 58
[RX] Inum   loc packets  [filtered   sniffed    logged   classed]
[RX]    0  IPin       0          0         0         0         0
[RX]    0  IPme       0          0         0         0         0
[RX]
[RX] : tcpdump packets discarded because of throttle: 0
```

Refer to the *IP Packet Filtering* chapter for more information about these **maint** commands.

Enable/disable ARP debug mode

Use the **maint 80** command to check if ARP debug mode is on or off:

```
GR 8> maint 80
[RX]
[RX] Arp Debug Mode is now off.
[RX] Use 'maint 80 1' to turn on.
```

Then you can use **maint 80 1** to turn it on or **maint 80 0** to turn it off.

Display ATMP information

Three **maint** commands return ATMP related information.

```
[RX] H O M E  N E T W O R K   T B L  :
[RX] =================================
[RX] S: Slot, P: Port, Rx: packets Received, BRx: Bytes Received
[RX]                  RTx packets transmitted, BTx: Bytes transmitted
[RX]
[RX]FRT-index  S:P:s0:s1     State     Address   VPN Address  VPN Netmask
[RX]--------   ---------     -----     -------   -----------  -----------
[RX]   6     2:01:888:0  LocCir,HA 10.9.9.9 10.20.2.237 255.255.255.0
```

The columns are defined as follows:

- **FRT-index**:
  Foreign agent Route Table, an arbitrarily-assigned tunnel number, not the tunnel ID, but the number you use in the **maint 73** command to display the tunnel ID
- **S:P:s0:s1 on Frame Relay (HSSI card)**:
  S = slot, P = port, s0 = DLCI number of the dedicated link to the home network (`pvcatmp`), s1 is currently unused and is always 0
- **S:P:s0:s1 on ATM**:
  S = slot, P = port, s0 = ATM VPI, s1 ATM VCI (`vc_atmp`)
- **State**
  indicates configuration and functional status depending upon the type/role of card.
  - On the home agent card (usually an Ethernet card acting as the home agent)

  The foreign agent uses an IP address on this card for tunnel negotiation. A state of `HomeAgent` indicates a home agent is properly configured to this card in `aitmd.conf`.
  - On any HSSI or ATM card on a GRF home agent, `State` can have three values:

  1. `HomeAgent` indicates that the home agent is properly configured in `/etc/aitmd.conf`, but this is not the link to the home network.

  2. `LocalCirc` indicates that this link is not part of ATMP configuration.

  3. `LocCir,HA` = the circuit to the home network is configured on this media card.
- **Address** is the IP address of the associated home agent (`atmp0`).
- **VPN Address** is the private network address the customer assigns to the interface that has the link to a home network, it only appears if entered in `/etc/aitmd.conf` file.
- **VPN Netmask** is the netmask for the VPN address.

Please ignore the "`Rx: packets Received, BRx: Bytes Received`" and "`RTx packets transmitted, BTx: Bytes transmitted`" headers, they no longer apply.

Then use **maint 73 *FRT_index*** command to display tunnel information "toward" the foreign agent, including tunnel IDs:

```
GR 2> maint 73 6
GR 2>
[RX] Mobile node tree list for home network index 6
[RX]  Mobile Node /Mask Flags Foreign Agent Tunnel Id  S:P:s0:s1
[RX]  10.4.0.0    /16   0  => 206.146.164.5 0x00000503 02:01:888:0000
```

The columns are as follows:

- mobile node non-routable IP address
- number of bits in the address netmask
- the route flags column is not currently used
- foreign agent routable IP address
- tunnel ID, in this case, 0x503
- **S:P:s0:s1 on Frame Relay (HSSI card):**
  S = slot, P = port, s0 = DLCI number of the dedicated link to the home network (`pvcatmp`), s1 is currently unused and is always 0
- **S:P:s0:s1 on ATM:** S = slot, P = port, s0 = ATM VPI, s1 ATM VCI (`vc_atmp`)

Collect data via grdinfo

With a single command, **grdinfo** collects the output from nearly all of the HSSI **maint** commands and compresses it in a log file. Refer to the "Management Commands and Tools" chapter in this manual for more information.

Use grstat ip to look at layer 3 statistics

```
# grstat ip ga02
card 2 (2 interfaces found)
  ipstat totals
       count description
      375149 total packets received
      368375 packets forwarded normally
           3 packets handled by the card
        6771 packets forwarded to the RMS
        6493 multicast packets received
        6493 multicast packets sent to RMS
  ipdrop totals
       count description
```

Ethernet Configuration

9

Chapter 9 describes the implementation of Ethernet on the GRF and other features supported on the Ethernet media card. It includes the information needed to configure Ethernet interfaces and parameters in the `/etc/grifconfig.conf` file and CLI profiles. This media is also known as “Fast Ethernet”.

Two types of Ethernet cards are available, one with eight physical interfaces and one with four. Configuration is the same for each type. Each independent physical interface is capable of connecting at speeds of 10 or 100 megabits per second.

Chapter 9 covers these topics:

On Ethernet cards, the fifth character is 0, 1, 2, 3, 4, 5, 6 or 7 to specify the eight physical interfaces on the card.

Examples: `ge067` specifies the bottom physical connector on the Ethernet card in slot 6, `ge000` specifies the topmost connector on the Ethernet card in slot 0.

Ethernet implementation

The GRF Ethernet standards implementation complies with the following RFCs:

RFC 791	Internet Protocol
RFC 792	Internet Control Message Protocol
RFC 826	Ethernet Address Resolution Protocol
RFC 894	Standard for the transmission of IP datagrams over Ethernet networks
RFC 1191	Path MTU Discovery

CSMA/CD (flow control)

Collision sensing capability is based upon the standard MAC-level CSMA/CD algorithm.

Autosensing and autonegotiation

Autosensing and autonegotiation support the Ethernet media card's capability to first determine connection options and then to operate at an optimal level.

When an Ethernet interface autosenses the 10 Mbs or 100 Mbs signal rate coming from another Ethernet device, the interface begins to operate at the detected rate.

If an interface is configured for autonegotiation, it can flexibly renegotiate the link at any time. In autonegotiation, a process occurs in which two endpoints of an Ethernet connection exchange signal and duplex status through a series of handshakes. Together, the interfaces arrive at the highest level of operation between them.

A situation can occur in which the GRF Ethernet card is set to negotiate but the connecting switch does not negotiate. The Ethernet card detects line rate but does not detect duplex, for example. If the GRF and the switch are running in different modes, for example, if the GRF is running the line in HDX and the switch is running FDX, a high rate of collisions and runts are reported. Accordingly, if you are seeing lots of collisions, check the setting of the connecting switch.

Transfer rates

Operating as 10 Base-T half-duplex, an interface transfer s data in one direction at a time at a rate of 10 megabits per second. Operating as 10 Base-T full duplex, an interface transfers data in both directions simultaneously at a rate of 10 megabits per second.

Operating as 100 Base-T half-duplex, an interface transfer s data in one direction at a time at a rate of 100 megabits per second.Operating as 100 Base-T full duplex, an interface transfers data in both directions simultaneously at a rate of 100 megabits per second.

Cables

Ethernet requires RJ-45 connectors on Category 5 UTP cables (untwisted shielded pair).

A note about half-duplex mode

When two Ethernet devices connected back-to-back are running in half-duplex mode, the Ethernet counters show transmit collisions as runts and CRCs. This can be normal occurence for a half-duplex, back-to-back Ethernet configuration.

In half-duplex mode, standard Carrier Sense Multiple Access with Collision Detection (CSMA-CD) rules are in effect. Any device that is contending for the Ethernet channel must wait for the channel to be idle before transmitting. Here is a description of what can happen with two directly-connected Ethernet devices, A and B.

Device A detects an idle channel and begins to transmit. It takes the preamble signal a certain amount of time to reach device B, up to 64 byte times are allowed for this to happen. However, in the interval before the signal is received, device B could detect the channel as idle and begin its own transmission. Assume device B does see the idle channel and starts a transmission.

At this point, A is transmitting to B and B is transmitting to A. When A's transmission starts to arrive at B, device B detects a collision and stops its in-flight transmission. Device A is now bound to receive a runt.

Likewise, when A receives the first part of B's transmission, A also detects a collision and stops in the middle of its transmission. Now device B will also see a runt reported.

Since A is back-to-back with B and if both are sending identical packet sizes to each other, there is a high degree of synchronization between each side. Each side detects collisions and runts along with the other, and the counts increment together.

Additionally, the CRC counter also increases at the same rate. This counter starts immediately after it receives the 8-byte preamble. As long as the counter receives a total of at least 12 bytes (8 for preamble, 4 for CRC), it will use the last four to determine if they form a valid CRC. If the last four bits do not, the CRC counter signals a CRC error, it does not wait for a minimum 64-byte frame. If the CRC counter starts to exceed the collsion/runt value, it means something other than normal CSMA/CD collisions are occuring.

This example holds true for two devices. The more devices configured, the more the collision and CRC counters tend to diverge from each other.

To maximize available bandwidth when running back-to-back, it is recommended that you set both sides to use full-duplex mode whenever possible.

Ethernet on the GRF

This section describes features of the Ethernet card.

Physical interfaces

The dual-speed Ethernet media card provides four or eight physical interfaces. An interface can run in either full duplex or half-duplex mode.

Additionally, an interface can operate at 10 or 100 megabits per second, as needed. This enables the Ethernet media card to interoperate with 10Base-T and 100Base-T devices. An interface can be configured to perform in a specific mode and transfer rate, or to autosense the mode and rate capacity of the connected host or network.

Logical interfaces

A logical interface is configured by its entry in the `/etc/grifconfig.conf` file where it is assigned an IP address and netmask. A logical interface is uniquely identified by its Ethernet interface name.

Large route table support

The Ethernet media card supports a route table with 150K entries. The card has the 4MB of memory required for large route tables and also has the /Q level of hardware support for efficient route table look up.

LLC/SNAP support

The Ethernet media card supports LLC/SNAP (IEEE 802.3) frames for IP and ARP.

ARP support

For Ethernet, the `/etc/grarp.conf` file maps an IP address to a six-byte physical address.

Proxy ARP

Proxy ARP is supported on GRF broadcast media, FDDI and Ethernet cards.

Proxy ARP enables a router to answer an ARP request on one of its networks that is actually destined for a host on another of the router's networks. This leads the sender of the ARP request into thinking that the router is the destination host, when in fact the destination host is "on the other side" of the router. The router acts as a proxy agent for the destination host, relaying packets to it from the other hosts.

MTU

The default Ethernet MTU size is 1500 bytes.

IS-IS protocol support

IS-IS is a link state interior gateway protocol (IGP) originally developed for routing ISO/CLNP (International Organization for Standardization/Connectionless Network Protocol) packets. In ISO terminology, a router is referred to as an “intermediate system” (IS). IS-IS intra-domain routing is organized hierarchically so that a large domain may be administratively divided into smaller areas using level 1 intermediate systems within areas and level 2 intermediate systems between areas.

IS-IS is supported on Ethernet media cards.

This example shows Ethernet interface `ge030` configured for IS-IS in the GateD IS-IS statement:

```
isis yes {
      area "49000080";
      systemid "326032603260";
      interface "ge030" metric 10 priority 60;
} ;
```

An ISO address must also be assigned to the Ethernet logical interface in `/etc/grifconfig.conf`. This is in addition to the entry for the IP address also assigned in that file. Refer to the *Introduction to IS-IS* chapter for more information.

Here is an example of Ethernet entries in `/etc/grifconfig.conf`:

```
#name   address         netwmask       broad_dest arguments
ge030  xxx.xxx.xxx.xxx 255.255.255.0        -     mtu 4352
#interface_name <iso_address> <iso_area> - iso
ge030 49.0000.80.3260.3260.3260.00 49.0000.80  -   iso
```

Transparent bridging

The GRF implements IEEE 802.1d transparent bridging on GRF Ethernet and FDDI interfaces. A FDDI interface may simultaneously bridge layer-2 frames and route layer-3 packets--that is, forward frames destined to a system attached to another LAN at the MAC layer, but still receive IP packets destined for a remote system attached to a non-broadcast GRF interface and route those packets at the IP layer.

On the FDDI card, frame forwarding is compatible with any station sending and receiving FDDI LLC frames. IPv4 frames are fragmented as necessary, as when bridging an FDDI frame of more than 1500 bytes to an Ethernet interface. The GRF bridge will attempt to break such a frame into fragments that will fit the sending interface. This is possible if the frame contains an IP datagram; then the GRF may use the fragmentation rules of IP to split the frame. Otherwise, the GRF must drop the frame.

Refer to the *Transparent Bridging* chapter in this manual for more information.

Selective packet discard

Selective packet discard can be enabled on the Ethernet card to ensure that dynamic routing packets are transmitted on the media in the presence of a sustained high volume of data packets. During high traffic volumes, data packets are discarded in a rate that favors dynamic routing packets.

Packet discard is regulated by reserving buffers for dynamic routing packets. This gives the operator complete control over the point at which congestion management begins to discard data packets. A user-configured threshold defines the percentage of buffers to reserve for dynamic routing packets.

When the threshold is set to zero, no buffers are reserved for dynamic routing packets and dynamic routing packet discard is disabled. In this case, dynamic routing packets and data packets are treated identically.

When the threshold is set to 100, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the threshold of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the number of free transmit buffers is less than the user-defined threshold of buffers required to be reserved for dynamic routing packets. When the number of free buffers used for switch receive/media transmit falls below the congestion threshold, non-dynamic routing packets are discarded until the congestion condition clears. Because the congestion condition is updated thousands of times per second and busy buffers are rapidly transmitted and returned as free buffers, a congested state ends rapidly after its onset. This prevents prolonged discard of non-dynamic routing packets and ensures the transmission of dynamic routing packets even during periods of heavy network load.

The discard mechanism applies only to the transmit side of the media card, and has no impact on packets received from the media. There is no analogous treatment of packets received from the media. The discard threshold is set to zero by default, and is therefore disabled by default.

The threshold value is unique per media card in the chassis, and is set at the Card profile in the CLI. Ascend recommends the threshold value be set low, to a small value that maximizes the benefit for dynamic routing packets and minimizes the impact on data packets. As the number reserved for dynamic routing packets increases, the number of buffers available for data traffic decreases and dynamic routing packets are a small percentage of all packets when the card is congested, Practice has shown it unnecessary to set the threshold above single digits as it is unlikely that dynamic routing packets account for more than a few percent of all packets.

Checking results

Examine GateD log files to determine the number of dynamic routing packets transmitted and their timestamps. A little arithmetic using the timestamps in the log files for packets transmitted to a neighbor (remember this is a transmit-only feature) should indicate the number of dynamic routing updates per unit time. Compare this number to the cumulative packet counters for switch receive over the same unit of time and you should arrive at the percentage of all transmit packets that are dynamic routing packets. Compare the average number over a few minutes to the number in a worst-case condition during bursts of dynamic routing packets based on periodic updates, and then select a percentage that balances the two.

Precedence handling

Precedence handling prioritizes delivery of dynamic routing update packets, even when the transmitting media card is congested. To ensure that dynamic routing update packets and other high priority packets are not dropped, the GRF uses precedence features to avoid this instability:

The GRF dynamic routing agent sets a precedence value in the internal packet header of the dynamic routing update packets it generates, which communicates to the media card a high-priority status for the packet.

The media card maintains a user-configurable threshold of transmit buffers that always remain available for high-priority traffic, ensuring that dynamic routing update packets are forwarded during congested conditions.

Precedence field

With selective packet discard enabled, the available buffer pool is managed as two pools, one for those with the "precedence field" set (high priority) and one for low priority data. Therefore, as the packets are taken off the switch, the buffer pools can be set up so that high priority packets will always find a buffer available, and the low priority packets will be dropped.

The precedence field is set in the IP packet header in one of two ways:

- by GateD on dynamic routing packets
- by filters configured to set this field on incoming data that matches any filter definition

Most dynamic routing packets sourced by the GRF have the precedence field set. This results in priority handling on the outbound (transmit) side of the media card in that a buffer is always made available for these packets as the data is read off the switch or communications bus. The media card starts discarding "low priority" packets before it completely runs out of buffers.

Controlled-load (class filtering)

Controlled-Load is supported on the Ethernet media card. The GRF delivers Controlled-Load service to a specific flow by marking its packets precedence field to prevent Selective Packet Discard (SPD). The marking mechanism uses filters to identify the packets belonging to the class of applications for which resources are reserved. Class filters are manually configured by adding them to `/etc/filterd.conf`.

Controlled-Load protects packets that match the filter from being lost. Packets that match the filter are marked so they will not be dropped by SPD. SPD drops packets that are not marked when the number of free buffers gets too low. Dynamic routing packet precedence fields are marked by GateD. The class filter is another way of setting the same precedence bit in the IP packet header. Refer to the *Integrated Services: Controlled-Load* chapter in this manual for information about class filters.

ICMP throttling

The Internet Control Message Protocol (ICMP) is a message control and error-reporting protocol between a host and a gateway to the Internet. ICMP uses IP datagrams, and the messages are processed by the TCP/IP software. ICMP throttling is a way of limiting the number of messages generated per GRF card.

You can specify how many of several types of ICMP messages can be generated by the Ethernet media card per one-tenth second. These are the message types:

– number of replies to echo requests
– number of "cannot deliver packet" replies (unreachable)
– redirect messages, number is not limited
– number of time-to-live replies
– number of parameter problem (packet discard) messages
– number of time of day time stamp replies to send

Specify ICMP throttling parameters in the Card profile.

Looking at the Ethernet card

Each Ethernet port has a set of four LEDs that describe the presence of a link and its type, the type of duplex or collision interface implemented, and port transfer activity. An 8-port Ethernet faceplate and LEDs are shown in Figure 9-1.

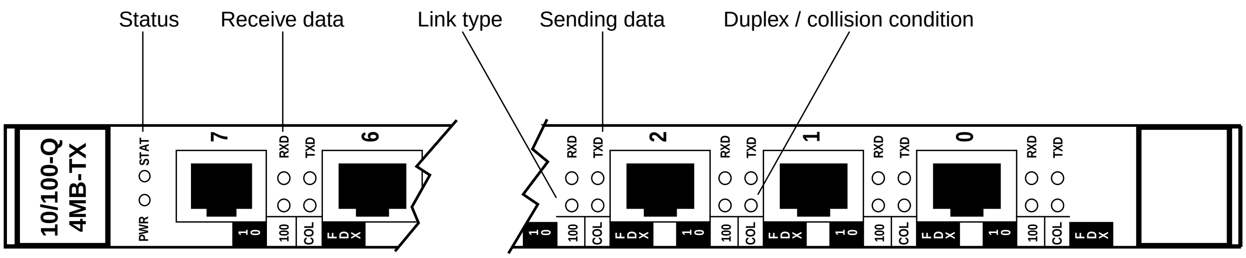


Figure 9-1. Ethernet media card faceplate and LEDs

LEDs for either type of Ethernet card are described in Table 9-1.

Table 9-1. Ethernet media card LEDs

LED	Description
PWR	This green LED is on when GRF power is on.
STAT	During normal operations, this LED is green. If an error condition is detected, this LED turns amber and remains on.
100	This LED is green for a 10 megabit link. This LED is amber for a 100 megabit link. This LED remains off (dark) when there is no viable link.
COL	This LED reads black (dark) for a half-duplex interface. This LED is amber for a half-duplex interface when encountering a transmission collision condition. This LED is green for a full-duplex interface.
RXD	This green LED indicates this port is receiving data.
TXD	This green LED indicates this port is transmitting data.

List of Ethernet configuration steps

These are the steps to configure Ethernet interfaces:

1 Assign IP address to each logical interface

Edit `/etc/grifconfig.conf` to assign an IP address for each Ethernet interface.

2 Specify Ethernet card parameters in the Card profile:

- specify verbose option for messages from Ethernet card
- configure interface mode: autonegotiate, 10 or 100 Base-T, full or half duplex
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard threshold
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

These next steps are optional, they describe tasks that are performed infrequently:

3 Change Load profile (optional).

Global executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every Ethernet card.

If you want to change the run-time code in one Ethernet card (per interface), make the change in the Card profile, in the `load` field.

4 Change Dump profile (optional).

Global dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.

If you want to change dump settings for one Ethernet card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1 To save files in the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

2 In the command-line interface, use **set** and **write** commands to save a profile.

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configure Ethernet interfaces

This section describes how to configure an Ethernet interface in the `/etc/grifconfig.conf` file. Use a UNIX editor to make entries in `/etc/grifconfig.conf`.

Each logical Ethernet interface is identified as to its:

- interface name, `ge0yz` (names are always lower case)
- Internet address
- netmask
- broadcast/destination address (optional)
- arguments field (optional)

Assign each interface an IP address, a GRF interface name, and if required, a netmask and destination/broadcast address. Here is the format for `grifconfig.conf` file entries:

```
#name   address    netmask       broad_dest    arguments
ge030    192.0.2.1    255.255.255.0    192.0.2.255
ge031    192.0.3.1    255.255.255.128  192.0.3.127
ge032    192.0.3.129  255.255.255.128  192.0.3.255
ge033    192.0.4.1    255.255.255.0    192.0.4.255
ge034    192.0.5.12   255.255.255.0    192.0.5.255
ge035    192.0.6.1    255.255.255.192  192.0.6.63
ge036    192.0.6.65   255.255.255.192  192.0.6.127
ge037    192.0.6.129  255.255.255.192  192.0.6.191
```

Interface name *go0yz*
Each logical GRF interface is given an interface name `ge0yz` where:

- the "ge" prefix indicates a Ethernet interface
- the chassis number is always "0"
- "y" is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
- "z" is the logical interface number in hex, 0–7 for Ethernet interfaces

Address
Enter the IP or ISO address to be assigned to this interface.

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured.

Broadcast address
Use the broadcast address when you wish to specify other than all 1s as the broadcast address.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO addess is being added to an interface's IP address. Specify the MTU value as `mtu` *xyz*. Leave the `arguments` field blank if you are not using it.

Example

The entry assigns an IP address for logical interface 0 on the Ethernet card in slot 6. If needed, a dash is used as a placeholder for the broadcast address:

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
ge060 192.0.2.1  255.255.255.0 192.0.2.255
```

If an interface is nonbroadcast (NBMA), do not include a destination address in its `/etc/grifconfig.conf` entry.

Save the /etc file

Save the file with the editor. Then, use **grwrite** to write the file to the `/etc` configuration directory:

```
# grwrite -v
```

Check contents of grifconfig.conf file

After you save the `/etc` directory and reset the media card, use **netstat -in** to display the contents of the `/etc/grifconfig.conf` file and verify that the logical interface is configured with the correct IP address.

Here is the output from a **netstat** command looking at the Ethernet interfaces:

```
# netstat -in | grep ge
ge0c1  1500  <link6>     00:c0:80:89:40:f4    0    0    0    0    0
ge0c1  1500  204.101.12  204.101.12.156       0    0    0    0    0
```

Please refer to the **netstat** man page for information about other **netstat** options.

Check system-level IP configuration

The UNIX **ifconfig** ***interface*** command returns system level information for the specified interface name, here is the interface for logical interface 0 (`ga020`):

```
# ifconfig ge0c1
ge0c1: gritether flags=140b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
              MULTICAST> mtu 1500
    inet 204.101.12.156 netmask 0xffffff00 broadcast 204.101.12.255
```

Setting parameters in the Card profile

This section describes how to verify and/or change Ethernet parameters in the Card profile. The parameters are presented in this order:

– specify verbose option for messages from Ethernet card
– configure interface mode: autonegotiate, 10 or 100 Base-T, full or half duplex
– OPTIONAL: specify ICMP throttling settings
– OPTIONAL: specify selective packet discard %
– OPTIONAL: change run-time binaries
– OPTIONAL: change dump variables

1. Specify Ethernet verbose setting

The `ether-verbose` field controls the level of messaging on the card. A level of 0 is normal, level 1 provides a higher number of messages, you can specify a level up to 9:

```
super> read card 7
CARD/7 read

super> list
card-num* = 7
media-type = ethernet-v1
debug-level = 3
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = Cisco-HDLC
ether-verbose = 1
ports = < { 0 {off on 10 3} {single off}{"" "" 1 sonet internal-oscilla+
load = { 0 < > 1 3 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10000 10 2147483647 10 10 10 }

super> set ether = 8
super write
CARD7/ written
```

2. Set the negotiation or transfer rate for each interface.

Set the negotiation or transfer rate in the `ports` / `ether` field. By default, the setting for each interface is autonegotiate. While you are at the top level of the Card profile, this is the path to the interface 0 (`if-config`) field:

```
super> list ports 0 ether
if-config = autonegotiate
super>
```

Use the **set** command to look at the interface options (autonegotiate is the default):

```
super> set if-config ?
if-config:
 Ethernet interface configuration.
```

```
Enumerated field, values:
 autonegotiate:  autonegotiate
 10-half:  10 BaseT Half Duplex
 10-full:  10 BaseT Full Duplex
 100-half:  100 BaseT Half Duplex
 100-full:  100 BaseT Full Duplex
```

The **set** command to specify `100 BaseT Half Duplex` for interface 0 is:

```
super> set if-config = 100-half
super> write
CARD/7 written
```

The shortest way to move on to the next port is to re-read the Card profile:

```
super> read card 7
CARD/7 read
super> list ports 1 eth
if-config = autonegotiate
super>set if-config = 10-full
```

3. Specify ICMP throttling

ICMP throttling settings are in the `icmp-throttling` section at the top level of the Card profile. Refer to Chapter 1 for an explanation of each field or do a `set <field-name>?` for a brief description.

Here is how to find out about the `param-problem` field:

```
super> set param-problem ?
param-problem:
 The number of ICMP packets indicating something wrong in params that
 are generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

You can use this shortcut to get to the ICMP fields when you read the Card profile:

```
super> read card 7
CARD/7 read
super> list ic
echo-reply = 10000
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
super>
```

Change two ICMP settings with these commands:

```
super> set echo-reply = 8
super> set TTL-timeout = 12
super> write
CARD/7 written
```

4. Specify selective packet discard threshold

Specify a SPD threshold for this Ethernet card in the `spd-tx-thresh` field. This field is contained in the `config` field of the top-level Card profile.

```
super> read card 7
CARD/7 read

super> list
card-num* = 7
media-type = ethernet-v1
debug-level = 3
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = Cisco-HDLC
ether-verbose = 1
ports = < { 0 {off on 10 3} {single off}{"" "" 1 sonet internal-oscilla+
load = { 0 < > 1 3 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10000 10 2147483647 10 10 10 }

super> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0

super> set spd-tx-thresh = 6
super> write
CARD/7 written
```

On reboot, the congestion threshold message should indicate the new setting, as shown below:

```
[2] [TX] Current congestion thresholds, out of 256 available buffers:
[2] [TX] Congestion:  17 (6%) [2] [TX]        Overshoot:  8
```

A discussion of how to decide an appropriate SPD percentage is provided in the “Selective packet discard” section on page 9-6.

5. Specify a different executable binary

Card-specific executable binaries can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this Ethernet card only.

```
super> read card 7
card/7 read
super> list load

super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you want to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_execut_for_ethernet
super> write
CARD/7 written
```

6. Change default dump settings

Card-specific dump file names can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 7
card/7 read
super> list dump

config = 0
hw-table = < >
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex. Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD/7 written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

To save the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

```
# grreset <slot_number>
```

Optional: change Ethernet binaries– Load profile

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** Ethernet cards.

Here is the path, defaults are shown:

```
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the Ethernet card settings:

```
super> list ether
type = ethernet-v1
rx-config = 0
rx-path = /usr/libexec/portcards/ether_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/ether_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

To execute different run-time code on the receive side of the Ethernet card, replace `/usr/libexec/portcards/ether_rx.run` with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/newether_rx.run
super> write
LOAD written
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. You have the option to configure the card's boot sequence so that after booting, the card loads and runs diagnostics before it loads and runs the executable binaries. Set the `enable-boot-seq` field to on and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the "Management Commands and Tools" chapter in this manual for information.

Optional: change Ethernet dumps – Dump profile

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Defaults are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 /var/portcards/grdump 0} {rmb 20 /var/portc+
dump-vector-table = <{ 3 rmb "RMB default dump vectors" < { 1 SRAM 26214+
config-spontaneous = off
keep-count = 2
super> set keep-count = 3
```

The `hw-table` field has settings for when dumps are taken and where dumps are stored. Here is the path to examine the Ethernet settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list ether
media = ethernet-v1
config = 20
path = /var/portcards/grdump
vector-index = 8
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP written
```

Dump vectors

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for all Ethernet cards. These fields are read-only and cannot be changed.

Here is the path, first you **cd** .. up to the main level:

```
super> cd ..
super> list . d
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" <{1 "atm inst memory" 167+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" <{1 "fddi/Q CPU0 core memor+
7 = {7 hssi "HSSI default dump vectors" <{1 "hssi rx SRAM memory" 209715+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" <{1 "Ethernet rx SRAM+
9 = {9 dev1 "DEV1 default dump vectors" <{1 "dev1 rx SRAM memory" 209715+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" <{1 "ATM-12 SDRAM +
11 = {11 sonet-v1 "SONET default dump vectors" <{1 "SONET rx SRAM memory+
```

This sequence shows a portion of the areas in the Ethernet card that are dumped:

```
super> list 8
index = 8
hw-type = ethernet-v1
description = "ETHERNET default dump vectors"
segment-table = <{1 "Ethernet rx SRAM memory" 2097152 4194304}{2 "Ether+

super> list s
1 = { 1 "Ethernet rx SRAM memory" 2097152 4194304 }
2 = { 2 "Ethernet shared SRAM memory" 131072 32768 }
3 = { 3 "Ethernet tx SRAM memory" 69206016 2097152 }

super> list 1
index = 1
description = "Ethernet rx SRAM memory"
start = 2097152
length = 4194304

super> cd ..
super> list s 2
index = 2
description = "Ethernet shared SRAM memory"
start = 131072
length = 32768
```

Monitoring Ethernet media cards

Use the **maint** commands to look at packet statistics on the Ethernet media card.

The **maint** commands operate on the control board and require the GR> prompt. Execute the **grrmb** command to switch prompts.

If you are not sure of the card's slot number, use the **grcard** command to view the location of installed cards.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

grrmb

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the Ethernet media card you are working with. For example, if you are working with a card in slot 2, enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 2
GR 2>
```

To leave the **maint** prompt, enter **`quit`**.

Receive / transmit side maint commands

Use **maint 1** to see the list of **maint** commands for the receive side, use **maint 101** to see the list for the transmit side.

Receive side list

```
GR 7> maint 1
[RX]  1:     Display this screen of Options
[RX]  2:     Display Version Numbers
[RX]  3:     Display Configuration and Status
[RX]  4:     Display Media Statistics
[RX]  5:     Display SWITCH Statistics
[RX]  6:     Display Combus Statistics
[RX]  7:     Clear statistics counters (may mess up SNMP)
[RX]  8:     Display ARP Table
[RX]  9:     History trace on/off [ 0 | 1]
[RX]  10:    Display History Trace
[RX]  11:    Display IPC Stats
[RX]  12:    Display HW Registers
[RX]  16:    Display Multicast Routing Table
[RX]  22:    Display RX Packet-Per-Second Rates [# sec avg]
[RX]  30:    Switch Test: Clear Stats (but not setup)
[RX]  32:    Switch Test: Setup [ patt len slots... ]
[RX]  33:    Switch Test: Start [ slots...]
```

```
[RX]  34:    Switch Test: Stop [ slots...]
[RX]  35:    Switch Test: Status [ slots...]
[RX]  38:    Switch Test: Send One [ slots...]
[RX]  45:    List next hop data: [family]
[RX]  50:    Filtering filter list: [detail_level [ID]]
[RX]  51:    Filtering filter list: [detail_level [IF]]
[RX]  52:    Filtering action list: [detail_level [ID]]
[RX]  53:    Filtering action list: [detail_level [IF]]
[RX]  54:    Filtering binding list: [detail_level [ID]]
[RX]  55:    Filtering binding list: [detail_level [IF]]
[RX]  56:    Display filtering statistics: [IF#]
[RX]  57:    Reset filtering statistics: [IF#]
[RX]  58:    Show filter protocol statistics
[RX]         note, IF/ID may be '-1' to indicate all of the given item
[RX]               while detail level is 0|1|2.
[RX]  70:    Display ATMP Home Network table
```

Transmit side list

Use **maint 101** to view the list of transmit-side Ethernet **maint** commands:

```
GR 7> maint 101
[TX]  101:     Display this screen of Options
[TX]  106:     Display Combus Statistics
[TX]  108:     Display Arp Table
[TX]  112:     Display HW Registers
[TX]  145:    List next hop data: [family]
[TX]  150:    Filtering filter list: [detail_level [ID]]
[TX]  151:    Filtering filter list: [detail_level [IF]]
[TX]  152:    Filtering action list: [detail_level [ID]]
[TX]  153:    Filtering action list: [detail_level [IF]]
[TX]  154:    Filtering binding list: [detail_level [ID]]
[TX]  155:    Filtering binding list: [detail_level [IF]]
[TX]  156:    Display filtering statistics: [IF#]
[TX]  157:    Reset filtering statistics: [IF#]
[TX]  158:    Show filter protocol statistics
[TX]         note, IF/ID may be '-1' to indicate all of the given item
[TX]               while detail level is 0|1|2.
```

Display operating status

Use **maint 3** to display configuration and status of all ports.

```
GR 7> maint 3
```

```
Ethernet Configuration and Status.
Up Time: 8 days, 10:2.30
Free Memory: 3433376
Port : [MAC.Address...] Link    Method...    Configuration....   -> Partner
----------------------------------------------------------------------
0 : [00:c0:80:09:3d:88]  Down   Fixed-Neg    100/HDX/Broadcast   ->   */*
1 : [00:c0:80:09:3d:89]  Up     Fixed         10/FDX/Broadcast   ->   -/-
2 : [00:c0:80:09:3d:8a]  Down   Negotiate
3 : [00:c0:80:09:3d:8b]  Down   Negotiate
4 : [00:c0:80:09:3d:8c]  Up     Fixed-Neg    100/FDX/Broadcast   ->  100/FDX
5 : [00:c0:80:09:3d:8d]  Up     Fixed-Neg     10/FDX/Broadcast   ->   10/FDX
6 : [00:c0:80:09:3d:8e]  Up     Negotiate     10/FDX/Broadcast   ->   ?/?
7 : [00:c0:80:09:3d:8f]  Up     Negotiate     10/HDX/Broadcast   ->   ?/?
```

Interface number

Interface MAC address

Link status

Connection method

Interface configuration

Negotiate
- negotiated with link partner

Fixed-Neg
- negotiated to a fixed value

Fixed
- link partner doesn't negotiate so interface is a fixed value

Link partner status

-/- = don’t care, partner doesn’t negotiate and I/F config fixed.
?/? = don’t know, partner doesn’t negotiate.
/ = not done negotiating.

Media statistics

Use **maint 4** to display media statistics for both the input side and the output side for one or all eight interfaces. If you do not specify one interface, you see the output for all eight.

The input (receive) port side is reported on first:

```
GR 7> maint 4
[RX]
[RX]                    Media Statistics
[RX] input:
[RX] Port      Bytes            Packets        Errors    Discards
[RX] ------------------------------------------------------------
[RX] 0 00000000000000201828 0000000000000003058 000000000 000000000
[RX] 1 00000000000010184370 0000000000000137667 000000000 000000000
[RX] 2 00000000000000000000 0000000000000000000 000000000 000000000
[RX] 3 00000000000000000000 0000000000000000000 000000000 000000000
[RX] 4 00000000000232721627 0000000000000162891 000000000 000000000
[RX] 5 00000000000000000000 0000000000000000000 000000000 000000000
[RX] 6 00000000000000000000 0000000000000000000 000000000 000000000
[RX] 7 00000000000000000000 0000000000000000000 000000000 000000000
[RX]
[RX] Port 0:
[RX]    Unsupported type: 0
[RX]    Runt errors:      32896
[RX]    Out of buffers:   0
[RX]
                 •
                 •
                 •
[RX]
[RX] Port 3:
[RX]    Unsupported type: 0
[RX]    Runt errors:      1
[RX]    Out of buffers:   0
[RX]
[RX] Port 4:
[RX]    Unsupported type: 0
[RX]    Runt errors:      36140
[RX]    Out of buffers:   0
[RX]
[RX] Port 5:
[RX]    Unsupported type: 0
[RX]    Runt errors:      2
[RX]    Out of buffers:   0
[RX]
[RX] Port 6:
[RX]    Unsupported type: 0
[RX]    Runt errors:      154331584
[RX]    Out of buffers:   0
[RX]
[RX] Port 7:
[RX]    Unsupported type: 0
[RX]    Runt errors:      0
[RX]    Out of buffers:   0
```

Statistics for the output port side are reported next:

```
[RX]
[RX] output:
[RX] Port      Bytes            Packets        Discards
[RX] --------------------------------------------------
[RX]  0 000000000000000000 000000000000000000 000000000
[RX]  1 000000000000000000 000000000000000000 000000000
[RX]  2 000000000000000000 000000000000000000 000000000
[RX]  3 000000000000000000 000000000000000000 000000000
[RX]  4 000000000000000000 000000000000000000 000000000
[RX]  5 000000000000000000 000000000000000000 000000000
[RX]  6 000000000000000000 000000000000000000 000000000
[RX]  7 000000000000000000 000000000000000000 000000000
[RX]
[RX] Port 0:
[RX]    Odd Length TX Packets: 0
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Collision Errors:   0
[RX]    TX Dropped No ARP:     0
                 •
                 •
                 •
[RX]
[RX] Port 4:
[RX]    Odd Length TX Packets: 0
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Collision Errors:   0
[RX]    TX Dropped No ARP:     0
[RX]
[RX] Port 5:
[RX]    Odd Length TX Packets: 0
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Collision Errors:   0
[RX]    TX Dropped No ARP:     0
[RX]
[RX] Port 6:
[RX]    Odd Length TX Packets: 0
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Collision Errors:   0
[RX]    TX Dropped No ARP:     0
[RX]
[RX] Port 7:
[RX]    Odd Length TX Packets: 0
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Collision Errors:   0
[RX]    TX Dropped No ARP:     0
```

When you specify the interface number (0-7), only the statistics for that interface are returned:

```
GR 7> maint 4 0
[RX]
[RX]                    Media Statistics
[RX] input:
[RX] Port      Bytes               Packets          Errors    Discards
[RX] -------------------------------------------------------------------
[RX]  0 00000000000000000000 00000000000000000000 0000000000 0000000000
[RX]
[RX] Port 0:
[RX]    Unsupported type: 0
[RX]    Runt errors:      49600
[RX]    Out of buffers:   0
[RX]
[RX] output:
[RX] Port      Bytes               Packets         Discards
[RX] --------------------------------------------------------
[RX]  0 00000000000000000000 00000000000000000000 0000000000
[RX]
[RX] Port 0:
[RX]    Odd Length TX Packets: 0
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        0
[RX]    TX Collision Errors:   0
[RX]    TX Dropped No ARP:     0
```

Display switch statistics

The **maint 5** command returns GRF switch statistics:

```
GR 7> maint 5
[RX]                    Switch Statistics
[RX] input:
[RX]        Bytes             Packets         Errors
[RX] ----------------------------------------------------------
[RX] 00000001816321556872 00000000025214736515 000000000
[RX]
[RX] output:
[RX]        Bytes             Packets         Errors    Overruns
[RX] -------------------------------------------------------------
[RX] 00000000001222582269 00000000000002691018 000000000 000000000
[RX]
[RX] Switch Transmit Data Errors:            0
[RX] Switch Transmit Fifo Parity Errors:     0
[RX] Switch Transmit Internal Parity Errors: 0
[RX] Switch Transmit Connection Rejects:     0
[RX] Switch Receive Encoding Errors:         0
[RX] Switch Receive Running Disparity Errors: 0
[RX] Switch Receive Receiver Errors:         0
[RX] Switch Receive Running Checksum Errors: 0
```

Display Combus statistics

The communications bus provides the 80-Mbs inter-card and IP switch control board communications path. Use **maint 6** to view Combus statistics:

```
GR 7> maint 6
[RX] Combus Status:
[RX]    Last interrupt status:          0x50703055
[RX] Combus Statistics:
[RX]    Message ready interrupts:          1625441
[RX]    Truncated input messages:             2415
[RX]    Grit messages for TX-CPU:              433
[RX]    Ip messages Rcvd (non-bypass):           0
[RX]    Raw messages:                            0
[RX]    ISO messages:                            0
[RX]    Grid messages:                     1625008
[RX]    Grid echo requests:                  46202
[RX]    Port available messages:                 0
[RX]    Segmented Packets:                  665416
[RX]    Segments Sent:                     1995260
[RX] Combus Errors:
[RX]    Bus in timeouts:      2865     Bus out timeouts:         0
[RX]    Out of buffer cond.:     0     Bad packet type:          0
[RX]    Dropped IP packets:      0     Bad packet dest:          0
[RX]    Receive Msg Errors:      0     Receive Format Errors:    0
[RX]    Receive Past End:     2415     Received Long Message: 10295
```

Clear status info

Use **maint 7** to clear the current collected statistics:

```
GR 7> maint 7
[RX]
[RX] All Media Statistics Cleared.
[RX] All Switch Statistics Cleared.
[RX] All Combus Statistics Cleared.
```

Display ARP tables

maint 8 displays the ARP table for one interface or, if no interface is specified, for all interfaces:

```
GR 7> maint 8
[TX]
[TX]    Arp Table for Interface 0:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]    204.101.10.158  00:c0:80:89:08:35      03      161
[TX]    Arp Table for Interface 1:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]    204.100.2.158   00:c0:80:89:08:36      03      392
[TX]    Arp Table for Interface 2:
```

```
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]
[TX]    Arp Table for Interface 3:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]
[TX]    Arp Table for Interface 4:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]    204.100.1.136   00:00:77:88:8d:8e      03      414
[TX]    Arp Table for Interface 5:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]
[TX]    Arp Table for Interface 6:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
[TX]
[TX]    Arp Table for Interface 7:
[TX]    IP Address      Mac Address            Status  TTL
[TX]    ==============  ==================     ======  ===
```

List of filters

Note: The *IP Packet Filtering* chapter has information about using the **maint** filtering command set.

maint 50 returns the list of filters by filter ID:

```
GR 7> maint 50
GR 7> filterID     type   status  access
       00000911   ctable (loaded)    0002
       00000912   ctable (loaded)    0004
       00000913   ctable (loaded)    0002
       00000918   ctable (loaded)    0002
```

Display filtering statistics

maint 56 returns a set of filtering statistics:

```
GR 7> maint 58
[RX]
[RX] -------libfilter->filterd protocol statistics------
[RX] Bad end points on ACK packets:        0
[RX] Bad end points on request packets:    0
[RX] Out of sync ack with none queued:     0
[RX] Out of sync ack with queue:           0
[RX] Out of sync request:                  0
[RX] Retranmitted packets:                 0
[RX] Recieved packets:                     8
[RX] Transmitted packets:                  8
```

Display IPC statistics

```
GR 7> maint 11
[RX] IPC Stats:
[RX] ==========
[RX] RX IPC Message Received:  1158978
[RX] RX IPC Message Sent:      51300
[RX] RX Grid Packets Received: 0
[RX] RX Overruns:              0
[RX] RX Local Messages:        12
[RX] TX IPC Message Received:  51300
[RX] TX IPC Message Sent:      1158977
[RX] TX Grid Packets Received: 7490
[RX] TX Overruns:              0
[RX] TX Local Messages:        12
```

Collect data via grdinfo

With a single command, **grdinfo** collects the output from nearly all of the Ethernet **maint** commands and compresses it in a log file. Refer to the “Management Commands and Tools” chapter in this manual for more information.

SONET OC-3c Configuration

10

Chapter 10 provides information needed to configure the SONET OC-3c media card.

Three framing protocols are supported over SONET OC-3c: Frame Relay, Point-to-Point Protocol (PPP), and Cisco HDLC.

The SONET card provides one redundant connection using two physical interfaces capable of connecting at speeds of 155 megabits per second.

Chapter 10 contains:

SONET implementation

The GRF SONET OC-3c supports the APS 1+1 Architecture of automatic protection switching, unidirectional, and non-revertive.

The GRF SONET implementation complies with:

- Bellcore Technical Reference
  Synchronous Optical Network (SONET) Transport Systems:
  Common Generic Criteria
  TR-NWT-000253
  Issue 2, December 1991
- ANSI T1.105-1988
  American National Standard for Telecommunications-
  "Digital hierarchy: optical interface rates and formats specifications"

This *Guide* assumes the user is familiar with the technical descriptions of APS functionality and requirements found in these specifications.

APS overview

APS allows the SONET media card to detect a failure on the working channel and to switch to a protection (standby) channel to handle the traffic until the fault is cleared.

In this configuration, the upper physical interface "A" is the working channel and the lower physical interface "B" is the protection channel.

APS 1+1 architecture

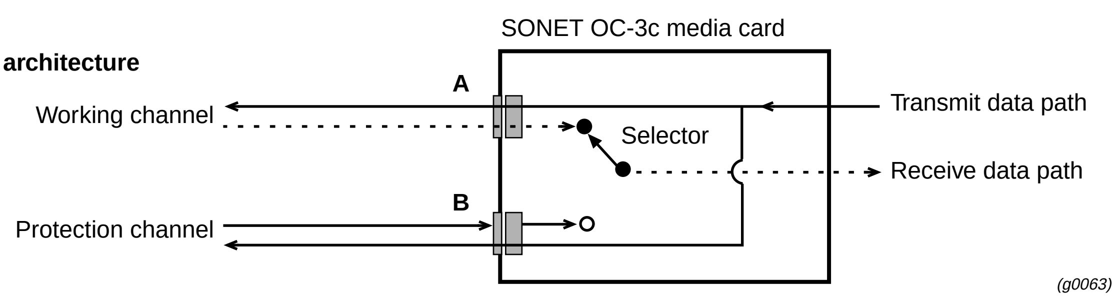


Figure 10-1. APS 1+1 architecture on the SONET OC-3c card

In a 1+1 architecture, outbound transmit signals are permanently bridged and are sent over both the working and protection channels. The transmit lines on both physical interfaces transmit the same data simultaneously. However, only the receive line on the working channel is connected to the media card's receive data path. The media card's selector component selects (switches) either the working or protection channel depending on the channel CSR (command, status, or request) setting. In unidirectional switching, the transmit and receive lines operate independently of each other. Non-revertive means that once the protection channel receive signal is selected, it stays selected until manually switched back to the working channel.

SONET OC-3c on the GRF

This section describes features of the SONET OC-3c card.

Physical interfaces

The SONET OC-3c media card provides a single redundant full-duplex interface. The automatic protection switching (APS) feature sets one interface to be the working channel and the other interface to be the protection channel. When hardware on the SONET media card detects degradation or interruptions in received signals, it automatically switches from the working channel to the protection channel.

Logical interfaces

A logical interface is configured by its entry in the `/etc/grifconfig.conf` file where it is uniquely identified by a SONET interface name (go0yx) and is assigned an IP address and netmask.

The number of logical interfaces configurable on the SONET media card depends upon which protocol is running.

A SONET card that is to run the PPP or HDLC protocol requires one entry into the `/etc/grifconfig.conf` file because these protocols only support one logical interface.

A SONET card that is to run Frame Relay will have as many as 128 entries since Frame Relay supports 128 logical interfaces per physical interface.

Protocols supported

Frame Relay

Frame Relay services provide a subset of the Data Link Layer and Physical Layer services, supporting the IETF encapsulation protocol and encapsulation of ARP frames. Frame Relay on a SONET interface supports User-to-Network-Interface (UNI) interfaces (DTE and DCE functionality), and Network-to-Network Interface (NNI). Up to 128 logical interfaces can be configured on the SONET media card.

The default Frame Relay MTU is set at 4352 bytes.

For interoperability, the following vendor documents are primary guides for defining the Frame Relay protocol:

- Frame Relay Physical Layer and Link Layer
  (including the subset of ANSI T1.602 LAPD protocol), documented in the US Sprint *Frame Relay Service Interface Specification* (Document #5136.03)
- the ANSI local management protocol developed and approved by ANSI,
  part of *T1.617, Annex-D*
- the CCITT local in-channel signaling protocol,
  part of *Q.933 ANNEX-A*

High-level Data Link Control protocol (HDLC)

The SONET OC-3c card supports the Cisco default High-level Data Link Control protocol (HDLC). CHDLC is the name given to Cisco's default version. Proper operation of this protocol is verified through interoperability testing using a GRF connected to a Cisco 7000 router.

The default HDLC MTU is 4352 bytes. A different value can be configured in the `/etc/grifconfig.conf` file.

Point-to-Point Protocol (PPP)

Point-to-Point Protocol (PPP) implementation conforms to IETF RFCs 1661 and RFC 1662. The current implementation supports link quality monitoring, but does not yet support a link quality policy to take action when the link quality is inadequate.

This release supports the following standard PPP options:

– maximum receive unit (LCP option 1)

– quality protocol (LCP option 4)

– magic number (LCP option 5)

– IP address (IPCP option 3)

The default PPP MTU is 1496. A different value can be configured in the `/etc/grifconfig.conf` file.

Note that the **ifconfig** command reports the MTU as 1500, this is actually the MRU (Maximum Receive Unit) and is a reporting error.

IS-IS protocol support

IS-IS is a link state interior gateway protocol (IGP) originally developed for routing ISO/CLNP (International Organization for Standardization/Connectionless Network Protocol) packets. In ISO terminology, a router is referred to as an "intermediate system" (IS). IS-IS intra-domain routing is organized hierarchically so that a large domain may be administratively divided into smaller areas using level 1 intermediate systems within areas and level 2 intermediate systems between areas.

The GRF SONET card supports IS-IS over Frame Relay.

This example shows SONET interface `go030` configured for IS-IS in the GateD IS-IS statement:

```
isis yes {
      area "49000080";
      systemid "326032603260";
      interface "go030" metric 10 priority 60;
} ;
```

An ISO address must also be assigned to the SONET logical interface in `/etc/grifconfig.conf`. This is in addition to the entry for the IP address also assigned in that file. Refer to the *GRF GateD Manual* for IS-IS configuration information.

Here is an example of SONET IP and IS-IS entries in `/etc/grifconfig.conf`:

```
#name   address         netwmask       broad_dest arguments
#
go030  196.132.11.24  255.255.255.0
#
#
#interface_name <iso_address> <iso_area> - iso
#
go030 49.0000.80.3260.3260.3260.00 49.0000.80  -   iso
```

Large route table support

The SONET OC-3c media card supports a route table with 150K entries. The card has the 4MB of memory required for large route tables and also has the /Q level of hardware support for expanded route table look up.

ICMP throttling

The Internet Control Message Protocol (ICMP) is a message control and error-reporting protocol between a host and a gateway to the Internet. ICMP uses IP datagrams, and the messages are processed by the TCP/IP software. ICMP throttling is a way of limiting the number of messages generated per GRF card.

You can specify how many of several types of ICMP messages can be generated by the SONET media card per one-tenth second. These are the message types:

- number of replies to echo requests
- number of “cannot deliver packet” replies (unreachable)
- redirect messages, number is not limited
- number of time-to-live replies
- number of parameter problem (packet discard) messages
- number of time of day time stamp replies to send

Specify ICMP throttling parameters in the Card profile.

On-the-fly configuration

Frame Relay supports on-the-fly configuration of links and PVCs without requiring the media card to be reset. The **grfr** command has options to add and delete, enable and disable, and modify links and PVCs.

Please refer to the “Configuring Frame Relay” chapter in this manual for more information.

Selective packet discard

Selective packet discard can be enabled on the HSSI card to ensure that dynamic routing packets are transmitted on the media in the presence of a sustained high volume of data packets. During high traffic volumes, data packets are discarded in a rate that favors dynamic routing packets.

Packet discard is regulated by reserving buffers for dynamic routing packets. This gives the operator complete control over the point at which congestion management begins to discard data packets. A user-configured threshold defines the percentage of buffers to reserve for dynamic routing packets.

When the threshold is set to zero, no buffers are reserved for dynamic routing packets and dynamic routing packet discard is disabled. In this case, dynamic routing packets and data packets are treated identically.

When the threshold is set to 100, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the threshold of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the number of free transmit buffers is less than the user-defined threshold of buffers required to be reserved for dynamic routing packets. When the number of free buffers used for switch receive/media transmit falls below the congestion threshold, non-dynamic routing packets are discarded until the congestion condition clears. Because the congestion condition is updated thousands of times per second and busy buffers are rapidly transmitted and returned as free buffers, a congested state ends rapidly after its onset. This prevents prolonged discard of non-dynamic routing packets and ensures the transmission of dynamic routing packets even during periods of heavy network load.

The discard mechanism applies only to the transmit side of the media card, and has no impact on packets received from the media. There is no analogous treatment of packets received from the media. The discard threshold is set to zero by default, and is therefore disabled by default.

The threshold value is unique per media card in the chassis, and is set at the Card profile in the CLI. Ascend recommends the threshold value be set low, to a small value that maximizes the benefit for dynamic routing packets and minimizes the impact on data packets. As the number reserved for dynamic routing packets increases, the number of buffers available for data traffic decreases and dynamic routing packets are a small percentage of all packets when the card is congested, Practice has shown it unnecessary to set the threshold above single digits as it is unlikely that dynamic routing packets account for more than a few percent of all packets.

Refer to the “Specify selective packet discard threshold” section later in this chapter.

Checking results

Examine GateD log files to determine the number of dynamic routing packets transmitted and their timestamps. A little arithmetic using the timestamps in the log files for packets transmitted to a neighbor (remember this is a transmit-only feature) should indicate the number of dynamic routing updates per unit time. Compare this number to the cumulative packet counters for switch receive over the same unit of time and you should arrive at the percentage of all transmit packets that are dynamic routing packets. Compare the average number over a few minutes to the number in a worst-case condition during bursts of dynamic routing packets based on periodic updates, and then select a percentage that balances the two.

Precedence handling

Precedence handling prioritizes delivery of dynamic routing update packets, even when the transmitting media card is congested. To ensure that dynamic routing update packets and other high priority packets are not dropped, the GRF uses precedence features to avoid this instability:

The GRF dynamic routing agent sets a precedence value in the internal packet header of the dynamic routing update packets it generates, which communicates to the media card a high-priority status for the packet.

The media card maintains a user-configurable threshold of transmit buffers that always remain available for high-priority traffic, ensuring that dynamic routing update packets are forwarded during congested conditions.

Precedence field

With selective packet discard enabled, the available buffer pool is managed as two pools, one for those with the "precedence field" set (high priority) and one for low priority data. Therefore, as the packets are taken off the switch, the buffer pools can be set up so that high priority packets will always find a buffer available, and the low priority packets will be dropped.

The precedence field is set in the IP packet header in one of two ways:

- by GateD on dynamic routing packets
- by filters configured to set this field on incoming data that matches any filter definition

Most dynamic routing packets sourced by the GRF have the precedence field set. This results in priority handling on the outbound (transmit) side of the media card in that a buffer is always made available for these packets as the data is read off the switch or communications bus. The media card starts discarding "low priority" packets before it completely runs out of buffers.

Controlled-load (class filtering)

Controlled-Load is supported on the SONET media card.

The GRF delivers Controlled-Load service to a specific flow by marking its packets precedence field to prevent Selective Packet Discard (SPD). The marking mechanism uses filters to identify the packets belonging to the class of applications for which resources are reserved. Class filters are manually configured by adding them to `/etc/filterd.conf`.

Controlled-Load protects packets that match the filter from being lost. Packets that match the filter are marked so they will not be dropped by SPD. SPD drops packets that are not marked when the number of free buffers gets too low. Dynamic routing packet precedence fields are marked by GateD. The class filter is another way of setting the same precedence bit in the IP packet header.

Refer to the Integrated Services: Controlled-Load chapter in this manual for information about constructing class filters.

Using the graps command

The **graps** command provides a way to manage the working and protection channel selection. Use the command to change the APS settings. (These settings can also be changed in the SONET Card profile.) **graps** provides standard APS options, for example, those defined by Bellcore R5-89.

The syntax is **graps -p *port*** where number is the card's slot number. The command returns three pieces of information:

– which channel is active, more specifically, on which channel is the receive data path fully connected. WORKING indicates the receive data path is active on the working channel, upper interface A. PROTECTION indicates the receive data path is active on the protection channel, lower interface B.

– the current CSR setting

– the last APS command (1 through 6 below) entered:

Then **graps** prompts you to enter another command or to quit. Here is an example:

```
# graps -p 6

APS channel selection: WORKING
      APS channel CSR: Do not revert
     Last APS command: Clear all other switch commands

Please enter command (? for help): ?
Commands:
   1) Clear all other switch commands
   2) Lockout protection channel
   3) Forced switch to protection channel
   4) Forced switch to working channel
   5) Manual switch to protection channel
   6) Manual switch to working channel
   q) quit
```

Looking at the SONET card

The SONET OC-3c media card provides a single redundant full-duplex interface. The SONET card is available in single and multimode versions. The single mode cards contain a Class 1 laser product. Figure 10-2 shows a SONET faceplate and LEDs.

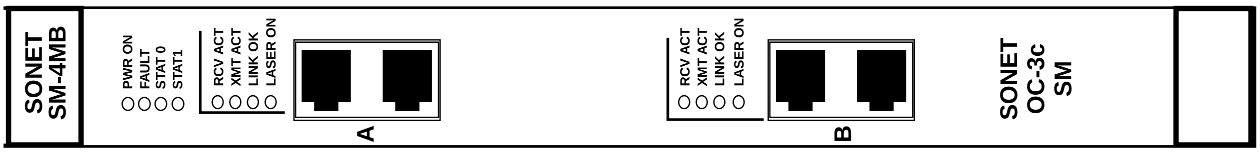


Figure 10-2. Faceplate of the SONET OC-3c media card

The SONET card provides redundant link connections across two physical interfaces. Only one logical interface is supported. By default, the upper link (A) is active. Use this interface if you are not setting up redundant links. If the active link is terminated, the redundant interface automatically becomes active.

LEDs on the faceplate

Table 10-1 describes the SONET card LEDs.

Table 10-1.SONET OC-3c LEDs

LED	Description
Power	This green LED is on when GRF power is on.
Fault	This amber LED turns on and remains on if an error condition is detected.
STAT 0 STAT 1	These green LEDs blink during self-test. When self-test completes, STAT 0 blinks ten times a second and STAT 1 blinks once a second. STAT 0 and STAT 1 indicate the activity of normal system interrupts. If the media card hangs, they either turn off and remain off, or they turn on and remain on.
RCV ACT	This amber LED blinks as data is received at the interface.
XMIT ACT	This amber LED blinks as data is transmitted out of the interface.
LINK OK	This green LED is on steadily to indicate which of the interfaces is active. The LED for the non-active interface blinks on and off.
LASER ON	This green LED provides a safety warning on single mode SONET cards. One should not look into a laser-active interface component when the cable is not plugged in.

List of SONET configuration steps

These are the steps to configure SONET cards:

1 Assign IP address to each logical interface

Edit `/etc/grifconfig.conf` to assign an IP address for each logical SONET interface.

2 Specify SONET card parameters in the Card profile:

- specify a framing protocol
- set APS command according to number of cables attached
- set mode to SDH or SONET
- specify internal oscillator or clock (null modem cable)
- specify SONET payload identifier
- if running HDLC, specify Cisco HDLC parameters
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard threshold
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

3 Configure the framing protocol:

Cisco HDLC - Steps 1 and 2 complete the configuration, reset the card.

Frame Relay - After steps 1 and 2, set Frame Relay and PVC parameters in the `/etc/grfr.conf` configuration file and reset the card.

Point-to-Point Protocol - After steps 1 and 2, set PPP parameters in the `/etc/grppp.conf` configuration file and reset the card.

These next steps are optional, they describe tasks that are performed infrequently:

4 Change Load profile (optional).

Global executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every SONET card.

If you want to change the run-time code in one SONET card, make the change in the Card profile, in the `load` section.

5 Change Dump profile (optional).

Global dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accommodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.

If you want to change dump settings for one SONET card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1. To save files in the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

2. In the command-line interface, use **set** and **write** commands to save a profile.

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configuring a SONET interface

This section describes how to configure a SONET interface in the `/etc/grifconfig.conf` file. Use a UNIX editor to make entries in `/etc/grifconfig.conf`.

Each logical SONET interface is identified in `/etc/grifconfig.conf` as to its:

- interface name, `go0yz` (names are always lower case)
- Internet address
- netmask
- broadcast/destination address (optional)
- arguments field (optional)

The format for an entry in the `grifconfig.conf` file is:

```
#name   address    netmask       broad_dest    arguments
```

Interface name *go0yz*
Each logical GRF interface is given an interface name `go0yz` where:

- the “go” prefix indicates a SONET interface
- the chassis number is always “0”
- “y” is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
- “z” is the logical interface number in hex, on the SONET card it is usually 0

IP address
Enter the IP address to be assigned to this interface.

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured. If no broadcast or destination is supplied, a netmask is required.

Broadcast or destination address
When you configure a logical interface on a point-to-point media, enter the destination IP address in the `broad_dest` address field.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO address is being added to an interface’s IP address. Specify the MTU value as `mtu` *xyz*. Leave the `arguments` field blank if you are not using it.

Example

The entry assigns an IP address for logical interface 0 on the SONET card in slot 6. If needed, a dash is used as a placeholder for the broadcast address:

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
```

```
go060 192.0.2.1 255.255.255.0 192.0.2.255
```

Note: By default, the WORKING interface is the upper one, interface 0. The interface name that correlates to interface 0 is `go060`. This is the interface always configured for the SONET card.

Save the /etc file

Save the file with the editor. Then, use **grwrite** to write the file to the `/etc` configuration directory:

```
# grwrite -v
```

Check system-level IP configuration

The UNIX **ifconfig** ***interface*** command returns system level information for the specified interface name, here is the interface for logical interface 0 (`go020`):

```
# ifconfig go060
go060: linktype 72 flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 1500
        inet 208.1.12.156 netmask 0xffffff00 broadcast 208.1.12.255
```

Check contents of grifconfig.conf file

After you save the `/etc` directory and reset the SONET card, use **netstat -in** to display the contents of the `/etc/grifconfig.conf` file and verify that the logical interface is configured with the correct IP address.

Here is the output from a **netstat** command looking at the SONET interfaces:

```
# netstat -in | grep go
go060  1500  <link46>                    20    0    20    0    0
go060  1500  208.1.11   208.1.11.156     20    0    20    0    0
```

Please refer to the **netstat** man page for information about other **netstat** options.

Setting parameters in the Card profile

This section describes how to verify and/or change SONET parameters in the Card profile. The parameters are presented in this order:

- framing protocol: `cisco-hdlc`, `ppp`, `frame-relay` (default is PPP)
- SONET hardware settings:
  APS: specify APS command, 1–6
  mode: specify card mode, SONET or SDH
  clock: specify internal oscillator or recovered clock (null modem cable)
- OPTIONAL: specify Cisco HDLC settings
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard threshold
- OPTIONAL: set card-specific load variables
- OPTIONAL: set card-specific dump variables

1. Set framing protocol

When you read and list the Card profile for this SONET media card, you will see that media card type, `sonet-v1`, is automatically in the read-only `media-type` field. Other values shown are defaults.

By default, the `sonet-frame-protocol` field is set to `PPP`. If the card is to run another protocol, you must change it to `Frame-Relay` or `Cisco-HDLC`.

```
super> read card 6
CARD/6 read
super> list card 6
card-num* = 6
media-type = sonet-v1
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < { 0 {off on 10 3} {single off}{"" "" 1 sonet internal-oscillat+
load = { 0 < > 1 3 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

In this example, the user changes the framing protocol to Frame Relay and saves the change:

```
super> set sonet-frame-protocol = Frame-Relay
super> write
CARD/6 written
```

You do not have to do a **write** until you have finished all changes in the Card profile. If you try to exit a profile without writing the changes, you get a warning message.

2. Set SONET parameters

You can change default settings for APS, mode, clock, and other SONET parameters.

The SONET parameters are located in the `ports 0` section of the top-level Card profile.

```
super> read card 6
CARD/6 read
super> list ports 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 1 sonet internal-oscillator 0 207 }
hssi = { 1 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disab+

super> list sonet
path_trace_message = ""
section_trace_message = ""
aps-command = 1
mode = sonet
clock = internal-oscillator
aps-status = 0
payload-identifier = 207
```

- The `path_trace_message` and `section_trace_message` fields are not currently in use.
- The APS commands have values 1 through 6:
  1 - clears out all other switch commands so you can enter a new one, **set** and **write** the 1 setting first, then **set** and **write** the new value you are entering.
  2 - prevents the protection channel, typically used when a card has only one cable
  3 - forces a switch of the working channel, overrides a hardware switch
  4 - forces a switch of the protection channel, overrides a hardware switch
  5 - manual switch of working channel
  6 - manual switch of protection channel
- Mode is `sonet` for SONET or `sdh` for SDH, the default is SONET.
- Clock is set to either `internal-oscillator` or to `recovered-clock`, default is internal.
- APS status is not available.
- Payload identifier can be set between 0 and 255, the default is 207.

Warning: Do not change this setting without first consulting with your local BCC/supplier. Payload identifier represents a path signal label that must be specified by the supplier.

Use **set** commands to change the SONET parameters:

```
super> set aps-command = 2
super> set mode = sdh
super> set clock = recovered-clock
super> set payload-identifier = 200
super> write
CARD/6 written
```

Check the changes you have made and saved. Since you are at the `ports` level, use **cd ..** to go "up" a level so you can access the SONET section again:

```
super> cd ..

super> list sonet
path_trace_message = ""
section_trace_message = ""
aps-command = 2
mode = sdh
clock = recovered-clock
aps-status = 0
payload-identifier = 200
```

3. Specify Cisco HDLC settings if running HDLC

If the card is to run HDLC, verify the HDLC settings are correct. The Cisco HDLC parameters are located in the `ports 0` or `ports 1` section of the Card profile. If you are at the `ports` level, use **cd ..** to go "up" a level so you can access the HDLC fields:

```
super> cd ..

super> list ports 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { single off }
sonet = { "" "" 2 sdh recovered-clock 0 200 }
hssi = { 1 16-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 incremental 5 300 10 10 03:00:0f:c0 disab+

super> list cisco
debug = off
keepalive-enabled = on
keepalive-interval = 10
keepalive-error-thresh = 3
```

The Cisco HDLC settings are:

- Debug turns on diagnostic messages about the Cisco-HDLC keepalive activity, messages are written to the `gr.console` log. The default is off, no diagnostic messages are collected.
- Keepalive activity can be turned off, the default is on.
- The default keepalive interval setting specifies how often the SONET interface sends keepalive messages, the default is every 10 seconds. Remember to specify the `keepalive-interval` setting in milliseconds.
- The keepalive error threshold specifies how many keepalive messages can go unanswered before the SONET interface marks the connection as down, three is the default.

If necessary, change the default keepalive settings and save the changes:

```
super> set keepalive-interval = 25
super> set keepalive-error-thresh = 6
super> write
CARD/6 written
```

4. Optional: Specify ICMP throttling

ICMP throttling settings are in the `icmp-throttling` section of the Card profile.

ICMP messages are described on page 10-5. Or, do a `set <field-name>?` for a brief description.

Default values are shown:

```
super> read card 6
CARD/6 read
super> list card 6
card-num* = 6
media-type = sonet-v1
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < { 0 {off on 10 3} {single off}{"" "" 1 sonet internal-oscillat+
load = { 0 < > 1 3 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }

super> list ic
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

Here is how to access the help information for the `echo-reply` field:

```
super> set echo ?
echo-reply:
 The number of ICMP ping responses generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

Change and save the default echo reply and TTL settings with these commands:

```
super> set echo-reply = 8
super> set TTL-timeout = 12
super> write
CARD/6 written
```

5. Specify selective packet discard threshold

Specify a SPD threshold for this SONET card in the `spd-tx-thresh` field. This field is contained in the `config` section of the Card profile.

```
super> read card 6
CARD/6 read
super> list
card-num* = 6
media-type = sonet-v1
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = Frame-Relay
ether-verbose = 0
ports = < {0{ off on 10 3} {single off}{"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 8 10 2147483647 12 10 10 }

super> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0

super> set spd-tx-thresh = 5
super> write
CARD/6 written
```

On reboot, the congestion threshold message should indicate the new setting, as shown below:

```
[2] [TX] Current congestion thresholds, out of 256 available buffers:
[2] [TX] Congestion:  17 (5%) [2] [TX]        Overshoot:  8
```

A discussion of how to determine an SPD threshold is provided in the "Selective packet discard" section earlier in this manual. The SONET **maint 4** command reports discard counts.

6. Specify a different executable binary

Card-specific executable binaries can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this SONET card only.

```
super> read card 6
card/6 read
super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you need to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_executable_for_sonet
super> write
CARD/6 written
```

To return the card to the previous code, change the file name in the `hw-table` field.

7. Change default dump settings

Card-specific dump file names and settings can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 6
card/6 read
super> list dump
config = 0
hw-table = < >
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD/6 written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

To save the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

```
# grreset <slot_number>
```

Optional: change SONET binaries – Load profile

Global executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** SONET cards.

Here is the path, defaults are shown:

```
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the SONET card settings:

```
super> list sonet
type = sonet-v1
rx-config = 0
rx-path = /usr/libexec/portcards/sonet_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/sonet_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

To execute different run-time code on the receive side of the SONET card, replace `/usr/libexec/portcards/sonet_rx.run` with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/newsonet_rx.run
super> write
LOAD written
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. In the Load profile, you have the option to change the boot sequence for all the cards of one type of media so that, after booting, those cards load and run diagnostics before they load and run the executable binaries. Set the `enable-boot-seq` field to on and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the “Management Commands and Tools” chapter in this manual for information.

Optional: change SONET dumps – Dump profile

Global dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Defaults are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accommodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 /var/portcards/grdump 0} {rmb 20 /var/portc+
dump-vector-table = <{ 3 rmb "RMB default dump vectors" < { 1 SRAM 26214+
config-spontaneous = off
keep-count = 2
```

The `hw-table` section has settings for when dumps are taken and where dumps are stored. Here is the path to examine the SONET settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list hw sonet
media = sonet-v1
config = 20
path = /var/portcards/grdump
vector-index = 11
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP written
```

Dump vectors

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for all SONET cards.

Here is the path, first you **cd** .. up to the main level:

```
super> cd ..
super> list . d
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" <{1 "atm inst memory" 167+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" <{1 "fddi/Q CPU0 core memor+
7 = {7 hssi "HSSI default dump vectors" <{1 "hssi rx SRAM memory" 209715+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" <{1 "Ethernet rx SRAM+
9 = {9 dev1 "DEV1 default dump vectors" <{1 "dev1 rx SRAM memory" 209715+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" <{1 "ATM-12 SDRAM +
11 = {11 sonet-v1 "SONET default dump vectors" <{1 "SONET rx SRAM memory+
```

This sequence shows a portion of the areas in the SONET card that are dumped:

```
super> list 11
index = 11
hw-type = sonet-v1
description = "SONET default dump vectors"
segment-table = <{ 1 "SONET rx SRAM memory" 2097152 2097152} 2 "SONET sh+

super> list segment
1 = { 1 "SONET rx SRAM memory" 2097152 2097152 }
2 = { 2 "SONET shared SRAM memory" 131072 32768 }
3 = { 3 "SONET tx SRAM memory" 69206016 2097152 }

super> list 1
index = 1
description = "SONET rx SRAM memory"
start = 2097152
length = = 2097152

super> cd ..
super> list seg 2
index = 2
description = "SOMET shared SRAM memory"
start = 131072
length = 32768
```

Configuring HDLC on SONET

Setting up the SONET card to run HDLC requires four configuration tasks:

- Specify logical interface in `/etc/grifconfig.conf`.
- Set framing protocol in Card profile to Cisco-HDLC.
- Check SONET settings in Card profile and change them as needed.
- Check Cisco-HDLC settings in Card profile and change them as needed.

These tasks are described in the preceding configuration sections, “Configuring a SONET interface” on page 10-12 and “Setting parameters in the Card profile” on page 10-14. Please use those sections to set up HDLC on the SONET interface.

Note: By default, the working SONET interface is the upper one, interface 0. For a card in slot 6, the interface name that correlates to interface 0 is `go060`. For HDLC, configure `go060` in `/etc/grifconfig.conf`.

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
#
go060 192.0.2.1 255.255.255.0 192.0.2.255
```

Configuring Frame Relay on SONET

Setting up the SONET card to run Frame Relay requires three configuration tasks:

– Specify logical interface in `/etc/grifconfig.conf.`

– Set framing protocol in Card profile to Frame-Relay.

– Check SONET settings in Card profile and change them as needed.

These tasks are described in the preceding configuration sections, "Configuring a SONET interface" on page 10-12 and "Setting parameters in the Card profile" on page 10-14. Please use those sections to set up HDLC on the SONET interface.

Note: By default, the active SONET interface is the upper one, interface 0.
For a card in slot 6, the interface name that correlates to interface 0 is `go060.` For Frame Relay, configure interfaces between 0 and 127, that is, `go060` through `go067f`.

What to do next...

Please use the "Configuring Frame Relay" chapter in this manual to configure PVCs on the SONET interfaces you have created in `/etc/grifconfig.conf`. Frame Relay is configured in the `/etc/grfr.conf` file. A copy of the file is in Chapter 2 of the *GRF Reference Guide*.

Configuring PPP on SONET

Setting up the SONET card to run the Point to Point (PPP) requires four configuration tasks:

- Specify logical interface in `/etc/grifconfig.conf.`
- Make sure the framing protocol field in the Card profile is set to PPP.
- Check SONET settings in Card profile and change them as needed.
- Create PPP interface in `/etc/grppp.conf`.

The first three tasks are described in the preceding configuration sections, “Configuring a SONET interface” on page 10-12 and “Setting parameters in the Card profile” on page 10-14. Please use those sections to set up Frame Relay on the SONET interface.

Note: By default, the working SONET interface is the upper one, interface 0. For a card in slot 6, the interface name that correlates to interface 0 is `go060`. For PPP, configure `go060` in `/etc/grifconfig.conf`.

```
# /etc/grifconfig.conf
#name  address   netmask     broad_dest    arguments
go060 192.0.2.1 255.255.255.0 192.0.2.255
```

Configuring the PPP interface in grppp.conf

The fourth task is to create the PPP interface in `/etc/grppp.conf` and assign the required PPP parameters. This configuration step binds PPP to the SONET interface.

Here are the steps:

1 Open the UNIX shell:

```
super> sh
# cd /etc
# vi grppp.conf
```

Use a UNIX editor to edit `/etc/grppp.conf`. A copy of the file is provided on page 10-29.

Note: To make immediate, temporary changes to the PPP configuration, use the **grppp** command, refer to the **grppp** man page for more information. Temporary settings done with **grppp** are lost when the SONET card is reset or the GRF is rebooted. Make permanent changes in the configuration file.

2 Set up a PPP interface, this setting binds PPP to the interface.

Identify the interface using the `go0yz` name:

```
# configure SONET i/f in slot 6
interface go060
   enable negotiation trace     #writes traces into /var/log/gr.console
   enable ipcp                  #allow IP traffic over PPP
   enable osinlcp               #allow osi traffic over PPP
```

The three “enable” parameters that follow the interface entry are frequently used. These are actually **grppp** commands.

Other **grppp** commands can be entered in the configuration file. Most of these commands override default values and should be used with caution. These are described in the next steps. The function of each command is provided here. Refer to the **grppp** man page for more information about each.

3 Optional: Specify optional automata parameters.

`set maximum configuration request count = INTEGER`

- Sets number of unanswered configuration requests allowed (default is 10).

`set maximum failure count = INTEGER`

- Sets number of connection non-acknowledgments taken. (default is 5)

`set maximum terminate count = INTEGER`

- Limits number of termination requests sent. (default is 2)

`set restart timer interval = INTEGER`

- Times sending of configuration and termination requests. (default is 3000 milliseconds)

4 Optional: Specify Link Control Protocol (LCP) parameters.

`enable lcp magic number`

- Enabled only to detect looped-back networks.

`set lcp keepalive interval = INTEGER`

- Time allowed between packets, the default of 0 milliseconds disables keepalive feature.

`set lcp keepalive packet threshold = INTEGER`

- Limits number of echo packets unanswered before link is closed. (default is 5 packets)

`set lcp mru = INTEGER`

- Defines maximum packet size. (default is 1500 octets)

5 Optional: Set Link Quality Reporting (LQR) parameters .

`enable lqr`

- Turns on collection of link quality reporting statistics. (default is disabled)

`set lqr timer interval = INTEGER`

- Sets time period between LQR messages sent by one endpoint to peer, begins the exchange of statistics between endpoints, specified in 1/100 seconds. (default is 0)

6 After you have entered the appropriate parameters, save the file with the UNIX editor. Then use the **grwrite** command to write the file to the `/etc` directory:

```
# grwrite -v
```

Verifying interface configuration with netstat

Here is the output from a **netstat** command looking at the SONET interfaces:

```
# netstat -in | grep go
go060  1500  <link46>                    20     0     20     0     0
go060  1500  208.1.11   208.1.11.156     20     0     20     0     0
```

Using grppp commands

Use the **grppp** status commands to display PPP objects and configuration values.

These are the **grppp** status commands:

```
show configuration
show negotiation trace status
show maximum configuration request count
show maximum failure count
show maximum terminate count
show restart timer interval
show lcp keepalive interval
show lcp keepalive packet threshold
show lcp mru
show lcp status
show lqr timer interval
show lqr status
show ipcp status
```

At the UNIX prompt you enter the **grppp** command and the prompt changes:

```
# grppp
>
```

At the > prompt enter interface and the interface name, the prompt changes again:

```
>interface go060
go060>
```

Commands are entered in lower case, short forms of words can be used. Use **quit** to exit the **grppp** prompt.

Looking at a PPP configuration

Here is the output from a **grppp show config** command:

```
# grppp
>interface go060
go060> show config
  General Configuration:
    Maximum configure request count: 10
    Maximum request failure count: 5
    Maximum terminate request count: 2
    Negotiation tracing is enabled
    Restart timer interval: 3000 milliseconds
  LCP Configuration:
    Magic number is disabled
```

```
    Initial MRU: 1500
    Keepalive interval: 0 milleseconds, disabled
    Keepalive packet threshold: 5
  LQR Configuration:
    LQR is disabled
    Timer interval: 0 milleseconds
  IPCP Configuration:
    enable IPCP
  OSINLCP Configuration:
    disable OSINLCP
go060>
```

Here is the output from a **show lcp status** command:

```
go060> show lcp st
  LCP Status:
    Bad addresses: 0
    Bad controls: 0
    Packets too long: 0
    Bad FCSs: 0
    Local MRU: 1500
    Remote MRU: 1500
  LCP Configuration:
    Magic number is disabled
    Initial MRU: 1500
    Keepalive interval: 0 milleseconds, disabled
    Keepalive packet threshold: 5
go060>
```

Contents of grppp.conf file

Figure 10-3 shows the contents of the `/etc/grppp.conf` file.

```
# Netstar $Id: grppp.conf,v 1.4 1997/03/25 16:54:45 suseela Exp $
#
# Template grppp.conf file.
#
#
# This file is used to set the initial configuration of PPP interfaces.
#
# When a media card configured for PPP is reset, grinchd executes grppp
# to process this file.   The following subset of grppp commands may be
# used in the grppp.conf file.  Most of these commands are used to
# overide default values, and should be used with caution.  Refer to
# the grppp man page for a full explanation of these commands.
#
#       interface INTERFACE_NAME
#       enable negotiation trace
#       set maximum configuration request count = INTEGER
#       set maximum failure count = INTEGER
#       set maximum terminate count = INTEGER
#       set restart timer interval = INTEGER
#       enable lcp magic number
#       set lcp keepalive interval = INTEGER
#       set lcp keepalive packet threshold = INTEGER
#       set lcp mru = INTEGER
#       enable lqr
#       set lqr timer interval = INTEGER
#       enable ipcp
#       enable osinlcp
#
# The example below shows the most commonly used grppp commands used in
# a grppp.conf file.

#
# Example Gigarouter PPP initial configuration
#
# interface gs0b0                       # Card 11, port 0
#    enable negotiation trace          # copy negotiaton traces to /var/log/gr.console
#    enable ipcp                        # allow IP traffic over PPP
#
# interface gs0b1                       # Card 11, port 1
#    enable ipcp                        # allow IP traffic over PPP
#    enable osinlcp                     # allow osi traffic over PPP
#
```

Figure 10-3. Template for grppp.conf file

Monitoring SONET OC-3c media cards

Use the **maint** commands to look at packet statistics on the SONET media card.

The **maint** commands operate on the control board and require the GR> prompt. Execute the **grrmb** command to switch prompts.

If you are not sure of the card's slot number, use the **grcard** command to view the location of installed cards.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

```
# grrmb
```

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the SONET media card you are working with. For example, if you are working with a card in slot 2, enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 2
GR 2>
```

To leave the **maint** prompt, enter **`quit`**.

Receive / transmit side maint commands

Use **maint 1** to see the list of **maint** commands for the receive side, use **maint 101** to see the list for the transmit side.

Receive side list

```
GR 06> maint 1
[RX]  1:     Display this screen of Options
[RX]  2:     Display Version Numbers
[RX]  3:     Display Configuration and Status
[RX]  4:     Display Media Statistics
[RX]  5:     Display SWITCH Statistics
[RX]  6:     Display Combus Statistics
[RX]  7:     Clear statistics counters (may mess up SNMP)
[RX]  8:     Display ARP Table
[RX]  9:     History trace on/off [ 0 | 1]
[RX]  10:    Display History Trace
[RX]  11:    Display IPC Stats
[RX]  12:    Display HW Registers
[RX]  16:    Display Multicast Routing Table
[RX]  22:    Display RX Packet-Per-Second Rates [# sec avg]
[RX]  30:    Switch Test: Clear Stats (but not setup)
[RX]  32:    Switch Test: Setup [ patt len slots... ]
```

```
[RX]  33:    Switch Test: Start [ slots...]
[RX]  34:    Switch Test: Stop [ slots...]
[RX]  35:    Switch Test: Status [ slots...]
[RX]  38:    Switch Test: Send One [ slots...]
[RX]  45:    List next hop data: [family]
[RX]  50:    Filtering filter list: [detail_level [ID]]
[RX]  51:    Filtering filter list: [detail_level [IF]]
[RX]  52:    Filtering action list: [detail_level [ID]]
[RX]  53:    Filtering action list: [detail_level [IF]]
[RX]  54:    Filtering binding list: [detail_level [ID]]
[RX]  55:    Filtering binding list: [detail_level [IF]]
[RX]  56:    Display filtering statistics: [IF#]
[RX]  57:    Reset filtering statistics: [IF#]
[RX]  58:    Show filter protocol statistics
[RX]         note, IF/ID may be '-1' to indicate all of the given item
[RX]               while detail level is 0|1|2.
[RX]  80:     S/UNI-PLUS loopback [0:none, 1:line, 2: serial, 3:parallel]
[RX]  81:     Test media write: 14 bytes, 0xff038021012800020306c0a81c92
[RX]  83:     Display signal detect
[RX]  87:     Frame Relay Arp Debug.
[RX]  88:     Display Frame Relay PVC table
```

Transmit side list

Use **maint 101** to view the list of transmit side SONET OC-3c **maint** commands:

```
GR 06> maint 101
[TX]  101:     Display this screen of Options
[TX]  106:     Display Combus Statistics
[TX]  112:     Display HW Registers
[TX]  145:    List next hop data: [family]
[TX]  150:    Filtering filter list: [detail_level [ID]]
[TX]  151:    Filtering filter list: [detail_level [IF]]
[TX]  152:    Filtering action list: [detail_level [ID]]
[TX]  153:    Filtering action list: [detail_level [IF]]
[TX]  154:    Filtering binding list: [detail_level [ID]]
[TX]  155:    Filtering binding list: [detail_level [IF]]
[TX]  156:    Display filtering statistics: [IF#]
[TX]  157:    Reset filtering statistics: [IF#]
[TX]  158:    Show filter protocol statistics
[TX]          note, IF/ID may be '-1' to indicate all of the given item
[TX]          while detail level is 0|1|2.
```

Display software and hardware versions

Use **maint 2** to display component revision levels.

```
GR 06> maint 2
[RX]             SONET Port Card Hardware and Software Revisions:
[RX]             ===============================================
[RX]
[RX] HW:
[RX]    Power-On Self-Test (POST) result code: 0x0.
[RX]    SONET Media Board HW Rev: 0x2, with 2M Sram.
[RX]    SONET Xilinx Version: 0x0.
[RX]    SDC Board HW Rev: 0x9  (SDC2).
[RX]            SDC2 Combus Xilinx version: 0x6.
[RX]            SDC2 Switch Transmit Xilinx version: 0x5.
[RX]            SDC2 Switch Receive Xilinx version: 0x0.
[RX]
[RX] SW:
[RX]SONET Code Version: A1_4_12R_2 Compiled Thu Sep 10 12:49:54 CDT 1998,
[RX]            in directory: /nit/A1_4_12R_2/sonet/rx.
[RX]IF Library Version: 1.1.0.0, Compiled Thu Sep 10 12:44:15 CDT 1998.
```

Display PPP and channel status

```
GR 06> maint 3
[RX]
[RX] SONET Configuration and Status.
[RX]    Framing Protocol: PPP.
[RX]    Free Memory: 948780
[RX]    SONET Line Mode:
[RX] Active Channel: WORKING
[RX] Channel Condition, State, or Request: Do Not Revert
```

Display media and SPD statistics

Use the **maint 4** command to look at the number of packets the transmit side drops:

```
GR 6> maint 4
[RX]                    Media Statistics
[RX] input:
[RX] Port       Bytes               Packets          Errors    Discards
[RX] ------------------------------------------------------------------
[RX]  0 00000000000000000060 00000000000000000006 0000000000 0000000000
[RX]
[RX] output:
[RX] Port       Bytes               Packets         Discards
[RX] -------------------------------------------------------
[RX]  0 00000001576741652844 00000000043798379250 0000000000
[RX]
[RX] Port 0:
[RX]    Odd Length TX Packets: 2
[RX]    TX Dropped Fifo Full:  0
[RX]    TX Dropped Line Down:  0
[RX]    TX Dropped SPD:        21
```

Display switch statistics

Use **maint 5** to display switch statistics for the SONET interface:

```
GR 06> maint 5
[RX]                    Switch Statistics
[RX] input:
[RX]        Bytes            Packets         Errors
[RX] ---------------------------------------------------------
[RX] 00000003364887118740 00000000046734543313 000000000
[RX]
[RX] output:
[RX]        Bytes            Packets         Errors    Overruns
[RX] ------------------------------------------------------------
[RX] 00000000000000001360 00000000000000000017 000000000 000000000
[RX]
[RX] Switch Transmit Data Errors:            0
[RX] Switch Transmit Fifo Parity Errors:     0
[RX] Switch Transmit Internal Parity Errors:  0
[RX] Switch Transmit Connection Rejects:     0
[RX] Switch Receive Encoding Errors:         0
[RX] Switch Receive Running Disparity Errors: 0
[RX] Switch Receive Receiver Errors:         0
[RX] Switch Receive Running Checksum Errors:  0
```

Display RX combus statistics

The communications bus provides the 80-Mbs inter-card and GRF control board communications path. Use **maint 6** to view Combus statistics:

```
GR 06> maint 6
GR 06> [RX]
[RX] Combus Status:
[RX]    Last interrupt status:          0x50503055
[RX] Combus Statistics:
[RX]    Message ready interrupts:          1163832
[RX]    Truncated input messages:                0
[RX]    Grit messages for TX-CPU:              253
[RX]    Ip messages Rcvd (non-bypass):           0
[RX]    Raw messages:                            0
[RX]    ISO messages:                            0
[RX]    Grid messages:                     1163579
[RX]    Grid echo requests:                 124497
[RX]    Port available messages:                 0
[RX]    Segmented Packets:                       0
[RX]    Segments Sent:                           0
[RX] Combus Errors:
[RX]    Bus in timeouts:         2      Bus out timeouts:         0
[RX]    Out of buffer cond.:     0      Bad packet type:          0
[RX]    Dropped IP packets:      0      Bad packet dest:          0
[RX]    Receive Msg Errors:      0      Receive Format Errors:    0
[RX]    Receive Past End:        0      Received Long Message:    3
```

Clear status info

Use **maint 7** to clear the current collected statistics:

```
GR 06> maint 7
GR 06> [RX]
[RX] All Media Statistics Cleared.
[RX] All Switch Statistics Cleared.
[RX] All Combus Statistics Cleared.
```

Display Frame Relay state

Use **maint 8** to display Frame Relay PVC state and information:

```
GR 06> maint 8
GR 06> [RX]
[RX] Frame-Relay PVC Status:
[RX] ('*' = address obtained via inverse arp)
[RX] ('+' = Enabled for ISIS)
[RX] name                 port  dlci  state  protocol address
[RX] ==================== ====  ====  ====== ================
```

Display IPC statistics

Use **maint 11** to display IPC statistics for receive and transmit sides:

```
GR 06> maint 11
[RX]
[RX] IPC Stats:
[RX] ==========
[RX] RX IPC Message Received:  403
[RX] RX IPC Message Sent:      412
[RX] RX Grid Packets Received: 0
[RX] RX Overruns:              0
[RX] RX Local Messages:        0
[RX] TX IPC Message Received:  447
[RX] TX IPC Message Sent:      403
[RX] TX Grid Packets Received: 400
[RX] TX Overruns:              0
[RX] TX Local Messages:        0
```

List of filters

maint 50 returns the list of filters by filter ID:

```
GR 06> maint 50
GR 06> filterID     type   status  access
       00000911   ctable (loaded)    0002
       00000912   ctable (loaded)    0004
       00000913   ctable (loaded)    0002
       00000918   ctable (loaded)    0002
```

Display filtering statistics

maint 56 returns a set of filtering statistics:

```
GR 06> maint 58
[RX]
[RX] Inum   loc packets  [filtered   sniffed    logged   classed]
[RX]    0  IPin       0          0         0         0         0
[RX]    0  IPme       0          0         0         0         0
[RX]
[RX] : tcpdump packets discarded because of throttle: 0
```

Display PVC configuration and statistics

The **maint 88** and **maint 89** commands return information for Frame Relay PVCs.

```
GR 6> maint 88
[RX]
[RX]
[RX]
[RX] C O N F I G U R E D   P V C s   :
[RX] =================================
[RX]
[RX] EP: End-Point, ISIS: ISIS flag
[RX] Port  DLCI   Type    CIR   Bc    Be    EPs/ISIS
[RX] ----  ----   ----    ---   ---   ---   --------

GR 6> maint 89
[RX]
[RX] P V C   S T A T I S T I C S s   :
[RX] =================================
[RX]
[RX] Port  DLCI  Pkts-rx   Pkts-tx   Pkts-dropped  Octets-rx  Octets-tx
[RX] ----  ----  -------   -------   ------------  ---------  --------
[RX]
```

List next hop data

Use **maint 45** to display the next hop table

```
GR 6> maint 45
[RX]
[RX] Location is: 0
[RX] Add:     7836 Delete:     7761 noNH:        0
[RX]    0: 206.146.160.156 (1   )   6:00  0:fc   RMS
GR 6> [RX]    1: 0.0.0.0         (1   )   6:00  0:fc   MCAST
[RX]    2: 127.0.0.1       (1   )   6:00  0:00   RMS
[RX]    3: 0.0.0.0         (1   )   6:00  0:fc   UNREACH
[RX]    4: 222.222.1.1     (1   )   6:00  0:00   RMS
[RX]    5: 0.0.0.0         (1   )   6:00  0:00   UNREACH
[RX]    6: 206.146.160.1   (1   )   6:00  0:fc   RMS
[RX]    7: 0.0.0.0         (1   )   6:00  0:fc   DROP
[RX]    8: 0.0.0.0         (1   )   6:00  0:fc   BCAST
[RX]    9: 0.0.0.0         (1   )   6:00  0:fc   RMS
[RX]   10: 0.0.0.0         (37  )   6:00  2:00   BCAST
[RX]   11: 0.0.0.0         (37  )   6:00  2:00   LOCAL
[RX]   12: 0.0.0.0         (30  )   6:00  7:00   FWD
[RX]   13: 0.0.0.0         (38  )   6:00  3:80   BCAST
[RX]   14: 0.0.0.0         (38  )   6:00  3:80   LOCAL
[RX]   15: 0.0.0.0         (38  )   6:00  3:80   FWD
[RX]   16: 0.0.0.0         (10  )   6:00  5:00 * BCAST
[RX]   17: 0.0.0.0         (10  )   6:00  5:00 * LOCAL
[RX]   18: 0.0.0.0         (13  )   6:00  4:00   BCAST
[RX]   19: 0.0.0.0         (12  )   6:00  f:00   BCAST
      .                .                .
      .                .                .
      .                .                .
[RX]  247: 170.63.86.187   (29  )   7:00  5:02   FWD
[RX]  248: 170.64.86.187   (30  )   7:00  5:03   FWD
[RX]  249: 170.65.86.187   (31  )   7:00  5:04   FWD
[RX]  250: 170.72.86.187   (38  )   7:00  5:0b   FWD
[RX]  251: 170.61.86.187   (27  )   7:00  5:00   FWD
[RX]  252: 170.62.86.187   (28  )   7:00  5:01   FWD
[RX]  253: 170.74.86.187   (7   )   7:00  5:7e   FWD
[RX] Location is: 1
[RX] Add:        0 Delete:        0 noNH:        0
```

ATM OC-12c Configuration Guide

11

Chapter 11 describes the implementation of ATM OC-12c on the GRF and other features supported on the ATM OC-12c media card. It includes the information needed to configure ATM OC-12c interfaces and parameters in CLI profiles as well as in the `/etc/grifconfig.conf` and `/etc/gratm.conf` files.

The OC-12c media card provides two independent physical ATM interfaces, each of which supports 220 logical interfaces.

Chapter 11 covers these topics:

ATM components

This section briefly describes components used in ATM configuration.

Virtual circuits and VCIs

A virtual circuit exists between two ATM devices. It is the point-to-point connection between two ATM devices. It is of no significance to other ATM devices.

Each virtual circuit is identified by a pair of numbers, representing a virtual path identifier (VPI) and a virtual circuit identifier (VCI). This pair is represented using a slash (/) to separate them (for example, 0/996). The VPI/VCI must be unique on a link. Because it is acceptable to use the same VPI/VCI on different links, a GRF can have the same VPI/VCI active on each physical interface.

The ATM OC-12c media card supports up to 1408 virtual circuits (VCs) as defined by the ATM Forum. Each virtual circuit has an associated IP address.

Virtual circuit identifiers (VCIs) name virtual circuits. VCIs are assigned in the `/etc/gratm.conf` file. Virtual circuits 0-31 on each VPI are reserved for use by the ATM Forum for specific functions:

Virtual paths and VCIs

A virtual path consists of one or more virtual circuits. The ATM OC-12c media card supports four virtual path identifiers. Virtual path identifiers VPI 0–3 are available.

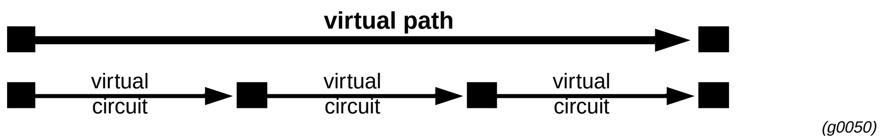


Figure 11-1. Components that form a virtual path

VPI/VCI

VPI/VCI specifies the Virtual Path Identifier and Virtual Circuit Identifier of the virtual circuit, separated by a slash (/), for example, 0/126. VPI/VCI is assigned in the `/etc/gratm.conf` file.

VPI 0: VCI 1 through VCI 1023 can be used.

On VPIs 1-3: VCI 1 through VCI 127 can be used

You cannot assign the same VPI/VCI to more than one circuit on the same physical interface.

Permanent virtual circuits

A permanent virtual circuit (PVC) is a logical connection across a physical path. Multiple PVCs share the same physical path. PVCs are configured statically and can be assigned a quality of service in terms of an amount of bandwidth. PVCs are configured in `/etc/gratm.conf`.

Switched virtual circuits

Switched virtual circuits (SVCs) are not supported on the ATM OC-12c card.

Traffic shaping

Traffic shaping is a specification of transmission parameters designed to ensure a specific quality of service (qos) between endpoints in ATM virtual circuits.

Traffic shaping parameters can be specified for PVCs and SVCs (via logical interface settings), but only for output; the GRF does not control (police) incoming cell packets. Outbound traffic flow is determined by the rates set on the transmitting interface. Traffic shaping only affects cells *leaving* the ATM card.

The GRF receives and sends IP packets. When a received packet has an ATM destination, the packet is sent over the switch to the forwarding ATM media card. The ATM card segments the packet into cells and sends them out over the appropriate virtual circuit.

Parameters

Traffic shaping on the ATM card uses three parameters that effectively manage the timing of the transmission of ATM cells over SONET OC-3c media.

The parameters are set in `/etc/gratm.conf` and include:

- peak cell rate, in kilobits/second (PCR)
- sustained cell rate, in kilobits/second (SCR)
- maximum burst size, in cells (MBS)

Quality of Service (qos) is also set in `/etc/gratm.conf`:

- qos priority is either high or low

Note: Remember to specify PCR and SCR in kilobits/second, not in bits/second.

For example, use 622080, not 622. If you specify 622, the ATM card moves data at 75 bytes per second, and appears to be non-functional.

Peak cell rate

Peak cell rate is the fastest rate at which cells will be output, or the maximum rate allowed a short burst of cells. The maximum peak rate for ATM OC-12c is 622080 kilobits/second. Cells can be sent at rates lower than the specified peak, but never faster.

Peak rate is the most basic level of traffic shaping. The peak is set to match the highest rate at which the receiving endpoint is able to accept incoming cells.

The GRF has a large buffer memory in which to buffer cells when they are arriving faster than the selected peak rate allows. If the mismatch in speeds is large, packets on the faster incoming network eventually will be lost, and retransmission will be required.

Sustained cell rate

Sustained cell rate (SCR) is the average send rate a VC is not allowed to exceed over time. On ATM OC-12c media cards, the sum of the sustained rates specified in the (up to) 16 traffic shapes cannot exceed 622080 kilobits/second.

In the `gratm.conf` file, the sum of all the `sustain=` values of all the shapes used on one card cannot exceed 622080.

The sustained rate sets an upper bound on the average cell rate (number of cells transmitted / duration of connection). If not specified, it defaults to the specified peak rate. Software adjusts each specified sustained cell rate so that it is a simple fraction (1/2, 1/3, 1/4, ... , 1/63) of the associated peak cell rate (PCR), rounding up.

Maximum burst size

Maximum burst size (MBS) is the number of cells a VC is allowed to send at the peak rate before it has to return to the sustained rate.

A maximum burst size can be specified to allow the peak rate to be larger than the sustained rate for some short period of time. If not specified, it defaults to the peak rate.

Maximum burst size is expressed in `/etc/gratm.conf` only as multiples of 53-byte cells:

- The smallest burst size allowed is 32 53-byte cells (13,568 bits).
- The largest burst allowed is 255 53-byte cells (108,120 bits).

Software for `/etc/gratm.conf` rounds the requested burst size downward to the next such multiple, or to 32, whichever is greater.

Maximum burst size has no meaning unless the specified sustained rate (SCR) is less than the peak rate (PCR). As long as the VC has data to send, it sends its cells at the sustained rate. If the VC runs out of data, it can accumulate a certain number of "credits" for cells not sent. Then when a packet is queued for output, cells can be sent at the peak rate until the credits (one per cell) are used up. After that, transmission goes back to the sustained rate. Within a certain latitude determined by the MBS, this allows a VC to transmit at the sustained rate on the average, even though it cannot supply data at that steady rate. The MBS value is the maximum credit in cells that a VC can accrue.

When setting the MBS, consider the ability of the connecting ATM switch to buffer cells. The larger the buffer, the bigger you can set MBS. A 1500-byte IP datagram takes 32 cells. A 9180-byte datagram uses 192. If the switch can handle it, it is likely that setting MBS to at least one of these values means that an entire packet can be sent at the peak rate even while the VC maintains a lower average rate.

Burst rate credits

Burst rate credits come from unused sustained rate transmit credits. This means that the virtual circuit (VC) has to have been transmitting below the sustained rate in order for any burst rate credits to accumulate. For bursting to occur, the VC must average less than the sustained rate. Unused sustained rate transmit credits can accumulate due to recent idle and under-subscribed periods.

- In a recent idle period, the circuit usually transmits at the sustained rate but has been idle for the last N cell times.
- In an under-subscribed period, the circuit usually transmits below the sustained rate.

Burst credits are accumulated at the sustained rate but are transmitted at peak rate.

In summary, if a circuit is not able to send a cell when it is its “turn”, the circuit accumulates a credit. When there is an accumulation of such credit, the circuit can issue cells at the peak rate until the credit is used.

Assigning traffic shaping profiles

Peak cell rate is the fastest rate at which a VC outputs cells, or the maximum rate allowed a short burst of cells. Sustained cell rate is the average send rate a VC is not allowed to exceed over time. Maximum burst size (MBS) is the number of cells a VC is allowed to send at the peak rate before it has to back off to the sustained rate.

The sustained rate allows the VC to send a certain quota of cells over time. If the VC transmits below the sustained rate, it accumulates credits that allow it to temporarily use the peak rate to catch up to the quota.

Five restrictions apply to the way in which traffic shaping profiles are assigned to VCs:

- Up to 16 different traffic shaping profiles are possible. A profile becomes “active” when a VC is created with that profile.
- There are 16 different priority levels. No two traffic shaping profiles can have the same priority.

  Cell-times claimed by one active profile are not available to lower-priority profiles. This is true even if the higher-priority profile has no cells to send.
- A profile’s basic bandwidth is not shared by the VCs assigned to it.

  N number of VCs assigned the same profile can consume up to (*N**SCR) worth of bandwidth.
- At each priority, the maximum bandwidth available to a single VC is equal to the total link bandwidth minus the sum of the SCRs of all higher priority profiles.

  Bandwidth not used by higher priority VC s is not available to lower priority ones.
- The previous statement implies that no single VC can have an SCR equal to the total link of bandwidth unless all VCs are configured to the same full rate profile. As a result, link bandwidth is wasted when VCs assigned to a particular profile are not using it, and there are active VCs at lower priority. The statement also means that VCs with different profiles cannot be serviced round-robin, only VCs with the same profile can be serviced round-robin.

ATM OC-12c traffic shaping parameters

The maximum ATM OC-12c peak rate is 622080 kilobits/second, the maximum aggregate sustained rate is 622080 kilobits/second, and the largest burst size that can be specified is 255 cells.

- The interfaces and PVCs configured for any one ATM OC-12c card can only use a maximum of 16 traffic shapes.

- The sum of the `sustain=` values of all the shapes (the peak value if sustain is not given) used on one card cannot exceed 622080.

If either rule is violated, the card will ignore the traffic shape of the offending interface or PVC, and will assign the VC one of the legal traffic shapes that most closely matches the desired one.

Queueing

Queueing among the VCs assigned to the same traffic shape is round-robin. Between those with different shapes, there is a priority queueing scheme that gives highest priority to those with the lower sustain or peak rates.

Priority

Priority is a characteristic of a traffic queue.

If high-priority and low-priority messages are both queued for output and are equally eligible to be sent as determined by traffic shaping, all high-priority queues will be serviced before any low-priority queues.

The rate queues are divided into two groups. Eight are high-priority, and eight are low-priority. PVCs and logical interfaces assigned to rate high rate queues have absolute priority for transmission over those assigned to low-priority queues.

In practice, all high-priority queues have the same high level of access, all low-priority queues have the same low level of access.

Priority becomes an attribute of the logical interface. A high-priority (for access) queue means a QoS = high. A low-priority (for access) queue equates to a QoS = low. QoS is specified in `/etc/gratm.conf` as part of the traffic shaping name.

Setting output rates

Sending at a controlled rate

To ensure that the transmission of cells does not exceed a specific rate, you can create a traffic shape specifying that peak rate.

When the optional sustained rate and maximum burst size are not specified, the ATM card automatically sets sustained rate to equal the specified peak rate. The GRF card attempts to steadily issue cells at the peak rate, but no faster.

Should cells come in faster than the specified peak rate allows them to go out, the GRF's memory will buffer them as necessary. Buffering serves to smooth the speed mismatch that can occur if, for example, data from a HIPPI source is being sent to an ATM end point.

However, if the speed mismatch is large enough, packets on the faster network will eventually be lost and retransmission will be required.

Allowing an average or fluctuating rate

To ensure that a defined average rate of cell transmission is maintained over the duration of a connection, specify a sustained cell rate (SCR), a maximum burst size (MBS), and a peak cell rate (PCR) for the VC.

A *sustained* rate is the upper bound of an average or *sustained* rate.

If SCR and MBS are specified, cells issue at the sustained rate. The sustained rate can be thought of as equivalent to assigning cell “slots” to the VCC at a certain time interval. If the VCC is not able to use its slot because no cell is ready to send, it accumulates a “credit”. Whenever there is accumulated credit, cells can issue at the peak rate until the credit is exhausted, and then cells will again issue at the sustained rate. Due to the time-slotted nature of ATM, the sustained cell rate must be no more than one-half of the peak rate to be effective.

Protocols supported

The ATM OC-12c card supports the protocols listed here. Each protocol has an associated `proto=` field in the `/etc/gratm.conf` file. Use this field to assign a protocol to a PVC.

This protocol is switched:

– raw adaptation layer (AAL-5) packets (`proto=raw`)

These protocols are routed:

– IP with LLC and SNAP encapsulation (ARP and IP) (`proto=ip`)

– LLC encapsulated ISIS, IS-IS only (`proto=isis`)

– LLC/SNAP encapsulated IP (IP, ARP, ISIS) and LLC encapsulated ISIS EXCEPT for RFC 1483 bridging (`proto=llc`)

– VC multiplexed IP, an interface using `bridge_method=vc_multiplexed`. (`proto=vc`)

The LLC and LLC/SNAP methods can encapsulate many datagram types, not just IP. The GRF can determine if an encapsulated packet is a type it can process based on fields that indicate payload type in the LLC or LLC/SNAP header.

The `proto=llc` type supports routed PDUs. When you specify `proto=llc`, the ATM card handles all the LLC or LLC/SNAP types it can —ISIS, IP, and ARP). Hence, you can specify IS-IS and IS-IS with IP as `proto=llc`. (This is referred to a wide-open LLC, anything which can be routed is routed.)

On an LLC/SNAP encapsulated circuit, the GRF can determine payload type on a per packet basis from the encapsulation header. It can be useful to restrict which encapsulated protocols the GRF actually processes. In the `/etc/gratm.conf` PVC statement, `proto=ip` refers to LLC/SNAP encapsulated IP and ARP in which all non-IP and non-ARP packets are discarded.

Using `proto=raw`, you can switch two ATM PVCs from one interface to another. The ATM circuit acts as an ATM AAL 5 switch, not an ATM cell switch, as it reassembles everything before "switching" packets. Mapping is port-VPI-VCI -> port-VPI-VCI, operating as a switch to extract the port from the destination interface field. This is non-routed, transparent transport of successfully reassembled AAL 5 PDUs from input to output, not a switch of ATM cells.

ATM OC-12c on the GRF

This section describes the implementation of ATM OC-12c card features on the GRF router.

The GRF ATM OC-12c card supports classical IP over ATM. The IP packet is carried directly over ATM.

Physical and logical interfaces

Figure 11-2 shows the organization of physical and logical interfaces on an ATM OC-12c media card:

Media card:

	Logical interfaces	VPI /	VCI	Total # of active VCs
Physical interface 0 (center)	0 – ff *(range)*	0	0 – 1023	1404
		1 – 3	0 – 127	

Figure 11-2. ATM OC-12c physical and logical interfaces

The ATM OC-12c media card supports a single physical interface that supports the assignment of 220 logical interfaces out of a range of 256.

Logical interfaces provide a simple way of mapping many IP addresses onto a single ATM port. The logical interface serves as the connection between ATM and IP, and is assigned a unique IP address in the `/etc/grifconfig.conf` file. Logical interfaces are numbered between 0 and 255 (0-ff).

Modes of operation

SDH and SONET

The ATM physical layer can be set to either SONET or SDH mode, SONET is the default. Mode is configured per physical interface (`connector=`), and not on a logical interface.

You specify mode in the Signalling section of the `/etc/gratm.conf` file. The example shows how mode is specified for the card in slot 5 as SDH and the card in slot 6 as SONET:

```
# Signaling parameters
Signaling card=5 connector=top protocol=UNI3.0  mode=SDH
Signaling card=6 connector=top protocol=UNI3.0  mode=SONET
```

Clock source

The ATM OC-12c SUNI component has a receive and a transmit clock. The receive clock is always at the SUNI's internal setting.

The transmit side clock setting can be toggled between the recovered receive clock (default, the SUNI's own internal clock) and the external oscillator (the clock of the transmitting node). The clock setting is specified in `/etc/gratm.conf` in a *Signalling* line entry.

Transmit clock can be toggled temporarily using the ATM card's **maint 22** command. The setting reverts back to the recovered receive clock (internal) at ATM card reboot and system reset.

Loop timing configures the transmit port to the recovered receive clock, receive and transmit are synchronized.

AAL 5

The ATM OC-12c media card supports only AAL-5. The system ignores any other AAL settings.

The ATM Adaptation Layer (AAL) supports the different types of traffic that can cross over ATM. The AAL consists of the Convergence Sublayer and a Segmentation and Reassembly (SAR) layer. The Convergence Sublayer consists of two smaller parts, the Common Part CS (CPCS) and the Service Specific CS (SSCS). The SSCS is used to specify which type of encapsulation is inside an ATM cell.

MTU

The maximum transmit unit for an ATM OC-12c packet is 9180 bytes, it cannot be set to a higher value. MTU settings are done per interface in the `/etc/grifconfig.conf` file.

LLC/SNAP encapsulation

The ATM OC-12c card supports LLC/SNAP encapsulation of IP datagrams. This is the encapsulation specified by RFC 1483.

NULL encapsulation

The ATM OC-12c card does not support NULL encapsulation (VC multiplex mode), and cannot communicate with an ATM subnet using NULL encapsulation.

Raw ATM mode limitations

The ATM OC-12c card supports raw ATM mode to other ATM OC-12c cards in the same GRF chassis. Raw connections between OC-12 and OC-3c cards are not supported.

Because it does not support raw mode from ATM OC-12c cards across the GRF backplane to ATM OC-3c cards in the same chassis, you cannot configure the GRF to operate as an OC12-OC3 ATM switch by configuring a raw PVC "through" the box, the PVC cannot come in on one type of ATM card and exit on the other type of ATM card.

ATM OC-12c does support raw mode between an ATM OC-12c card and an ATM OC-3c card interconnected through an ATM switch.

UNI signaling

UNI signalling is not supported. Set proto=NONE on the signalling entry in `/etc/gratm.conf`.

Large route table support

ATM OC-12c card software maintains route tables containing up to 150K entries, and provides hardware support for full table lookups. Use the first **netstat** command shown below to find out the number of table entries, use the second to display the system route table:

```
# netstat -rn | wc -l
# netstat -rn
```

On-the-flyconfiguration of PVCs

On ATM OC-12c cards you can reconfigure PVCs in the `/etc/gratm.conf` file without rebooting the card. The process uses the **gratm** command and is described in the "ATM on-the-fly PVCs" section of this chapter.

Packet buffering

The ATM OC-12c card has 1024 2KB buffers in each direction. A full packet of 9180 bytes uses five buffers. A 64-byte packet uses one buffer.

Buffering is provided for 204 full packets on the receive side and 204 full packets on the transmit side. A full packet contains 9180 bytes, the size of the ATM MTU.

A full packet contains 192 cells (192 is obtained by dividing 9180 by 48 bytes, the length of a cell's data payload). If packets are full, the transmit and receive sides can each output 39168 cells (204 packets x 192 cells).

The **maint 10** command displays memory and buffer usage, and reports free, fragmented, and available units. Refer to the "Getting ATM information" section at the end of this chapter.

Hardware forwarding

The ATM OC-12 card incorporates hardware forwarding on the media transmit side of the card. With hardware forwarding, or fast path, the CPU does not see or touch the packets at all. This hardware interprets the contents of the route table entry, but it does not search the ARP table. Packets for which the next hop is not the final destination can use the hardware fast path, packets that require an ARP table lookup are handled by the CPU. Hardware forwarding supports traffic in intermediate hops.

ICMP throttling

The Internet Control Message Protocol (ICMP) is a message control and error-reporting protocol between a host and a gateway to the Internet. ICMP uses IP datagrams, and the messages are processed by the TCP/IP software. ICMP throttling is a way of limiting the number of messages generated per GRF card.

You can specify how many of several types of ICMP messages can be generated by the ATM OC-12c media card per one-tenth second. These are the message types:

- number of replies to echo requests
- number of "cannot deliver packet" replies (unreachable)
- redirect messages, number is not limited
- number of time-to-live replies
- number of parameter problem (packet discard) messages
- number of time of day time stamp replies to send

Specify ICMP throttling parameters in the Card profile.

Inverse ARP

The ATM 0C-12c media card supports ATM inverse ARP.

The GRF supports Inverse ATM ARP for determining the IP address of the other end of the VPI/VCI. If the connecting device does not support Inverse ATM ARP, an ARP entry for the IP and VPI/VCI of the other device must be made in `/etc/grarp.conf`.

The GRF takes the ARP entry learned via ATM inverse ARP as opposed to the one in the `/etc/grarp.conf` file. If no ARP entry exists for a given PVC when **grarp** is run, the ARP entry given in the `/etc/grarp.conf` file is accepted.

When a GRF ATM interface receives an ARP entry via ATM inverse ARP for a PVC and the **gratm** process also tries to add an ARP entry for the same PVC, then **gratm** may exit with a message similar to this:

```
Jun 17 15:32:49 GigaRouter grinchd[120]:
/usr/sbin/grarp -i ga0yz -f /etc/grarp.conf exited status 1
```

ATM statistics and configuration data

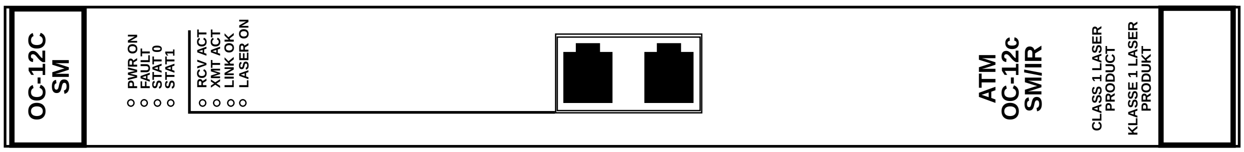
The ATM OC-12c card has **maint** commands that display card configuration and traffic data. Tese commands are described at the end of this chapter. Other tools useful for looking at the ATM OC-12c media cards include:

- **netstat -in**
- **ifconfig -a**
- **grstat** (layer 3 statistics only)

Examples of these tools are in the "Management Commands and Tools" chapter of this manual.

Looking at the ATM OC-12c card

The ATM OC-12c media card provides one full-duplex interface. ATM OC-12c cards are available in single and multimode versions. Figure 11-3 shows a single mode ATM OC-12c faceplate. Single and multimode faceplates are the same except that each single mode interface has a “LASER ON” LED.



(g0138)

Figure 11-3. Faceplate of the ATM OC-12c single mode media card

LEDs on the faceplate

The top four LEDs indicate card status. The duplex interface has a set of LEDs. Table 11-1 describes the ATM card LEDs.

Table 11-1.ATM OC-12c LEDs

LED	Description
Power	This green LED is on when GRF power is on.
Fault	This amber LED turns on and remains on if an error condition is detected.
STAT 0 STAT 1	These green LEDs blink during self-test. When self-test completes, STAT 0 blinks ten times a second and STAT 1 blinks once a second. STAT 0 and STAT 1 indicate the activity of normal system interrupts. If the media card hangs, they either turn off and remain off, or they turn on and remain on.
RCV ACT	This amber LED blinks as ATM cells are received at the interface.
XMT ACT	This amber LED blinks as ATM cells are transmitted out of the interface.
LINK OK	This green LED goes on when an optic cable is plugged into an interface and remains on while connection is good at both cable ends.
LASER ON	This green LED provides a safety warning on single mode ATM cards. One should not look into a laser-active interface component if a cable is not plugged in.

Ping times

You may notice some local pings to an ATM card can take a long time while other pings to that card are much faster. The following short discussion attempts to explain the differences in ping times. Ping times are affected by:

- amount of traffic going through the router generally
- low or high priority of the assigned rate queue
- traffic on VCs assigned to low priority rate queues in relation to the traffic on VCs assigned to high rate queues

Answering local pings from the RMS is a low priority task for any media card. The more packets there are passing through the router, the longer a local ping may take since packet processing has priority over local ping processing.

Another factor is the priority of the assigned rate queue. Any packet on a high priority rate queue superceedes ALL traffic on low priority rate queues. All QoS = high packets are transmitted before any QoS = low packets are transmitted. Therefore, pinging a low priority rate queue in the presence of high priority traffic should have high delay. The ping packets are the least likely to be processed.

Also, if many more VCs are assigned to the low priority queues than are assigned to the high priority queues, and you ping a VC on rate queue 07, that one low priority packet has to wait for all high priority traffic to be processed.

List of ATM configuration steps

These are the steps to configure ATM cards:

1 Assign IP address to each logical interface

Edit `/etc/grifconfig.conf` to assign an IP address to each logical ATM interface.

2 Configure PVCs and SVCs in the `/etc/gratm.conf` file.

Step 3 includes options a site may wish to configure, none of them are required:

3 Specify ATM card parameters in the Card profile

– OPTIONAL: specify ICMP throttling settings

– OPTIONAL: change run-time binaries

– OPTIONAL: change dump variables

These next steps are also optional, they describe tasks that are performed infrequently:

4 Change Load profile (optional)

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in every ATM card.

If you want to change the run-time code in one ATM card , make the change in the Card profile, in the `load` field.

5 Change Dump profile (optional)

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the recommended default.

If you want to change dump settings for one ATM card, make the change in the Card profile, in the `dump` field.

Save / install configurations and changes

1. In the command-line interface, use **set** and **write** commands to save a profile. The profiles are stored in the `/etc` directory.

2. To save files in the `/etc` configuration directory, use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card to have the change take effect. Enter:

```
# grreset <slot_number>
```

Configuring an ATM OC-12c interface

This section describes how to configure an ATM interface in the `/etc/grifconfig.conf` file. Defining the logical interface is the first step to configure an ATM virtual circuit. Use a UNIX editor to make entries in `/etc/grifconfig.conf`.

Each logical ATM interface is identified in `/etc/grifconfig.conf` as to its:

- interface name, `ga0yz` (always lower case)
- Internet address
- netmask
- broadcast/destination address (optional)
- arguments field (optional)

The format for an entry in the `grifconfig.conf` file is:

```
name   address    netmask       broad_dest    arguments
```

Interface name *ga0yz*
Each logical GRF interface is given an interface name `ga0yz` where:

- the “ga” prefix indicates an ATM interface
- the chassis number is always “0”
- “y” is a hex digit (0 through f) for the slot number (GRF 400, 0–3; GRF 1600, 0–15)
- “z” is the logical interface number in hex

Logical interfaces range from `0` to `ff`.

Address
Enter the IP or ISO address to be assigned to this interface.

Netmask
Specify the netmask as a 32-bit address for the network on which the interface is configured.

Broadcast address
Use the broadcast address when you wish to specify other than all 1s as the broadcast address.

Arguments
The `arguments` field is optional, and is currently used to specify an MTU value that is different from the standard or default value. Also, the `arguments` field is used to specify ISO when an ISO addess is being added to an interface. Specify the MTU value as `mtu` *`xyz`*. Leave the `arguments` field blank if you are not using it.

Examples

The first entry assigns an IP address for the ATM OC-12c card in slot 2, and specifies an MTU value lower than default. A dash is used as a placeholder for the broadcast address:

```
#/etc/grifconfig.conf
#name   address    netmask       broad_dest    arguments
#
ga030 10.20.2.234 255.255.255.0 - mtu 9100
ga040 10.20.2.238 255.255.255.0 10.20.2.239
```

The second entry sets an IP address for the ATM OC-12c card in slot 13, and specifies a destination address.

Save the /etc file

After you use the editor to save and close an `/etc` configuration file, write the file to the `/etc` configuration directory. Use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

Check system-level IP configuration

The UNIX **ifconfig *interface*** command returns system level information for the specified interface name, here is the interface for logical interfaces `ga020` and `ga027f`:

```
# ifconfig ga020
ga020: gritatm flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 9180
        inet 10.20.2.234 netmask 0xfffffffc broadcast 10.20.2.235

# ifconfig ga027f
ga027f: gritatm flags=b043<UP,BROADCAST,RUNNING,LINK0,LINK1,
               MULTICAST> mtu 9180
        inet 10.20.2.238 netmask 0xfffffffc broadcast 10.20.2.239
```

Check contents of grifconfig.conf file

The **netstat -in** command returns the contents of the `/etc/grifconfig.conf` file. Please refer to the **netstat** man page for information about other **netstat** options and explanantions of the type of information presented.

Here is the output from a **netstat** command looking at the ATM interfaces:

```
# netstat -in | grep ga
```

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs	Coll
ga000	9180	<link14>	00:c0:80:fb:0f:00	437	0	14	0	0
ga000	9180	205.1.10	205.1.10.156	437	0	14	0	0
ga010	9180	<link15>	00:c0:80:f8:33:00	0	0	0	0	0
ga010	9180	208.1.11	208.1.11.156	0	0	0	0	0
ga0180	9180	<link16>	00:c0:80:f8:34:80	13	0	13	0	0

```
ga0180  9180  205.1.11    205.1.11.156       13    0    13    0   0
ga020   9180  <link37>    00:c0:80:f7:b2:00  12    0    12    0   0
ga020   9180  205.1.12    205.1.12.156       12    0    12    0   0
ga0380  9180  <link38>    00:c0:80:f7:72:8   14    0    14    0   0
ga0380  9180  205.1.13    205.1.13.156       14    0    14    0   0
ga040   9180  <link13>    00:00:00:00:00:00  16    0   189    0   0
ga040   9180  208.1.10    208.1.10.156       16    0   189    0   0
ga090   9180  <link17>    00:c0:80:fa:54:00   4    0     4    0   0
ga090   9180  204.101.11  204.101.11.156      4    0     4    0   0
```

Using the gratm.conf file

This section describes the `/etc/gratm.conf` configuration file. All ATM circuits and circuit parameters are configured in `gratm.conf`.

The file has five sections: Service, Traffic Shaping, Signalling, Interfaces, and PVC.

Service section

In the Service section you define the available ARP services. Give a different name to each type of ARP service you define. These names are assigned to the interfaces defined in the Interface section and specify the ARP service a logical interface will use.

Traffic shaping section

In the Traffic Shaping section you define the available traffic shapes. Give a different name to each type of traffic shape you define. These names are assigned to the interfaces defined in the Interface and PVC sections, and specify the traffic resources allotted to a logical interface.

Signalling section

In the Signalling section you assign a signalling protocol to each physical interface, top and bottom. The ATM OC-3c interface is always top, `connector=top`.

Interfaces section

In the Interfaces section you identify the logical interfaces configured on this ATM card. Optionally, you can assign an ARP service name, a Traffic shaping name, and a bridging method to each logical interface.

PVC section

In the PVC section you assign three required parameters to each Permanent Virtual Circuit: the interface name (`ga0yz`), a VPI/VCI, and a protocol. Optional parameters include: an input AAL, a traffic shape name, or a destination interface

When editing `/etc/gratm.conf`, remember:

- Statements in `gratm.conf` may be longer than a single line. A statement can span multiple lines by ending each incomplete line of the statement with a back slash (\ ) character.
- Comments follow the Bourne Shell style. All characters following a # on a line are ignored.
- Names for ARP services and traffic shapes must be defined before they are assigned in the Interface and PVC sections.

A copy of the template for `/etc/gratm.conf` is in the *GRF Reference Guide*.

Configuring a PVC

This example configures a PVC with the following attributes:

- connects to a destination that does not support inverse ARP
- requires high priority quality of service
- is on card in slot 4
- runs in SDH mode
- must be set to destination clock
- is on logical interface 153 (hex=99)
- has a VPI/VCI of 0/32
- runs IP protocol, AAL-5 (default, no matter what is set)
- IP address is 192.0.130.1
- the remote IP address is 192.0.130.111

In configuring a PVC, the IP address of the local ATM interface should be on the same subnet as the remote IP address.

Entries in /etc/gratm.conf

- Service section

  Define the destination ARP server, you can define up to three.

  ```
  Service name=arp0 type=arp addr=47000580ffe1000000f21c20e80020481c20e800
  ```

- Traffic Shaping section

  Set traffic shaping name and quality of service parameters.

  ```
  Traffic_Shape name=oc12 \
          peak=622000 sustain=500000 burst=255 qos=high
  ```

- Signaling section

  Set protocol=NONE, PVCs do not require signalling.

  The ATM OC-12c connector is always `top`.

  ```
  Signalling card=4 connector=top protocol=none
  ```

- Interface section

  Specify interface name and traffic shape name for the logical interface.

  ```
  Interface ga040  traffic_shape=oc12
  ```

- PVC section

  Specify these characteristics:

  - assigned logical interface name
  - VPI/VCI
  - protocol supported

  ```
  PVC ga040  0/32 proto=ip
  ```

Entry in /etc/grifconfig.conf

Assign the IP address to the interface name, a netmask is required.

```
#name  address     netmask       broad_dest   arguments
ga040 192.0.130.1 255.255.255.0
```

Entries in /etc/grarp.conf

An entry is needed that maps IP and NSAP addresses if the destination device does not support inverse ARP.

Saving the files

After you use the editor to save and close an `/etc` configuration file, write the file to the `/etc` configuration directory. Use **grwrite -v**, verbose mode displays the file name as each is saved:

```
# grwrite -v
```

Verifying the PVC configuration

This section describes commands to review and verify PVC configuration parameters.

Check gratm.conf file entries

This **gratm** command parses the `/etc/gratm.conf` file on the specified media card without performing any configuration actions. It reports errors and file omissions.

```
# gratm -n ga02
gratm: Accepted traffic shape hshq qos=high for top connector card 2.
gratm: Begin on-the-fly PVC configuration for card 0x2
/usr/nbin/grinch -p 2 2.12.2.3.17.3.1=1
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.1=-1
/usr/nbin/grinch -p 2 2.12.2.3.4.2.5.1=-1
/usr/nbin/grinch -p 2 -A 2.12.2.3.10=1
/usr/nbin/grinch -p 2 -A 2.12.2.3.4.1.5.3=1
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.3.1.1=0
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.3.1.2=155000
/usr/nbin/grinch -p 2 2.12.2.3.4.1.5.3.1.3=1
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.32=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.33=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.34=2048
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.35=0
/usr/nbin/grinch -p 2 -A 2.12.2.3.10=128
/usr/nbin/grinch -p 2 -A 2.12.2.3.11.3.12=1
/usr/nbin/grinch -p 2 2.12.2.3.11.3.12.1.1=1
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.31=0
/usr/nbin/grinch -p 2 -A 2.12.2.3.11.3.12.1.2=1
/usr/nbin/grinch -p 2 2.12.2.3.11.3.12.1.2.1.1=10.20.2.237
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.32=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.33=155000
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.34=2048
/usr/nbin/grinch -p 2 2.12.2.3.10.128.5.35=0
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.1=00000130 00000000 00000000
    00007fff 00000000 00000000 00000000 00000000 00000002 00000000
    00000000 00000000 00000000 00000001 00025d78 00025d78 00000800
    00000000 00000000
/usr/nbin/grinch -p 2 2.12.2.3.10.1.5.1=00000128 0000007f 0000000f
    000001ff 00000000 00000000 00000000 00000000 00000002 00000000
    00000000 00000000 00000000 00000001 00025d78 00025d78 00000800
    00000000 00000000
/usr/nbin/grinch -p 2 2.12.2.3.17.3.1=0
gratm: Sent 0 grinches for card 0x2
```

Here is an error message from **gratm**:

```
# gratm -n ga0a
gratm: Parse error in "/etc/gratm.conf" file near line 232.
gratm: Input error on 'm' in 'Signalling' section.
# Oct  5 21:25:51 herman gratm: Parse error in "/etc/gratm.conf" file
near line 232.
Oct  5 21:25:51 herman gratm: Input error on 'm' in 'Signalling' section.
```

Verify VPI/VCIs per port

This **maint 13** command reports the VPI/VCIs that are configured.

```
GR 4> maint 13

GR 4> IF VPVCI TYP RQ  FPCR   FSCR   FMBS           DEST
-------------------------------------------------------------
00 0/150 pvc 00 622079 500052 000255 LLC
```

Check ARP entries

Use **maint 8** to check the card's current ARP entries in `/etc/grarp.conf`.

```
GR 4> maint 8
GR 4>
       IP       IF  ST               NSAPA                     VPI/VCI
-----------------------------------------------------------------------
208.001.010.158 00 PERM  ???????????????????????????????????????  0/150
```

Check physical link

Use the **maint 20** command to verify the port link is up, and verify physical parameters such as mode and timing.

```
GR 4> maint 20
GR 4> SUNI/SONET   -Internal timing -Internal loop-back disabled
        -Line loop-back disabled -Signal present -Clock present -Link on
TACP-  TSOCI: 00000000  FOVRI: 00000000
RACP-  OOCDI: 00000000  CHCSI: 00000000  UHCSI: 00000000  FOVRI: 00000000
       FUDRI: 00000000
RPOP-  FEBEI: 00000000  BIPEI: 00000000  PYELI: 00000000  PAISI: 00000000
       LOPI:  00000000  PSLI:  00000000
RLOP-  FERFI: 00000000  LAISI: 00000000  BIPEI: 00000000  FEBEI: 00000000
RSOP-  BIPEI: 00000000  LOSI:  00000000  LOFI:  00000000  OOFI:  00000000

Section BIP-8: 00000000  Line BIP-24: 00000000  Line FEBE: 00000000
Path FEBE: 00000000 Path BIP-8: 00000000
Correctable HCS: 00000000  Uncorrectable HCS: 569bfb12
```

Add/delete PVCs on-the-fly

On ATM OC-12c cards you can add/delete PVCs in the `/etc/gratm.conf` file without rebooting the media card.

There are four steps to add interface `ga03c8` as a PVC on the ATM card in slot 3:

1 Edit `/etc/grifconfig.conf` to reflect the added/deleted PVC:

```
# name   address      netmask         broad_dest      arguments
ga03c8  192.0.130.1   255.255.255.0
```

2 Edit `/etc/gratm.conf` to reflect the added/deleted PVC:

```
# Traffic shaping parameters
Traffic_Shape name=sshq peak=15000 qos=high #slow_speed_high_quality
# Interfaces
Interface ga03c8  traffic_shape=sshq
# PVC's
PVC ga03c8 0/32 proto=ip traffic_shape=sshq
```

3 Use the **gratm -n ga0<*slot*>** command to first check for any errors in `/etc/gratm.conf`:

```
# gratm -n ga03
```

As this command executes, you see numerous messages similar to these:

```
gratm: Accepted traffic shape sshq qos=high for bottom connector card 0.
gratm: Accepted traffic shape sshq qos=high for bottom connector card 1.
gratm: Begin on-the-fly PVC configuration for card 0x3
   /usr/nbin/grinch -p 1 2.12.2.4.17.3.1=1
```

Errors encountered by **gratm** are indicated by a line number where the error is detected. Fix the problem before re-running the **gratm -n** command.

4 Use **gratm ga0<*slot*>** to reconfigure the ATM OC-12c card:

```
# gratm ga03
```

As this command executes, you see numerous messages similar to these:

```
# gratm ga01
gratm: Begin on-the-fly PVC configuration for card 0x3
Oct  2 18:22:57 box1 kernel: ga03c8:  GRF ATM, GRIT address 0:1:0xf0
gratm: Sent 12 grinches for card 0x3
# Oct 2 18:22:57 box1 kernel: ga03c8: GRF ATM, GRIT address 0:1:0xf0
```

Now use the **ifconfig -a** command to see that a new interface is added.

After the ATM OC-12c card is reconfigured, a summary appears in the `grconsole.log` indicating which PVCs were added, which were deleted, and which were updated.

Note: On-the-fly configuration applies only to PVCs.

Rate queue (traffic shape)
Values in a rate queue cannot be changed on the fly. Changes must be made in the `/etc/gratm.conf` file and the ATM media card rebooted.

Other ATM configuration options

Supply address for ARP service

You need to supply IP-to-physical address mapping information for ARP service ONLY if the remote destination does NOT support inverse ATM ARP. The GRF supports Inverse ATM ARP for determining the IP address of the other end of the VPI/VCI. If the other device does not support Inverse ATM ARP, an ARP entry for the IP and VPI/VCI of the other device must be made in `grarp.conf`.

```
#/etc/grarp.conf
#[ifname] hostname  phys_addr   [temp]   [pub]   [trail]
#
ga0399  192.0.130.111 0/32
```

Changing the transmit clock source

The ATM OC-12c SUNI component has a receive and a transmit clock. The receive clock is always at the SUNI’s internal setting. The transmit side clock setting can be toggled between the recovered receive clock (default, the SUNI’s own internal clock) and the external oscillator (the clock of the transmitting node).

You can specify the transmit clock permanently in the Signalling section of the `/etc/gratm.conf` file. The example shows how clock is specified for the top interface as internal (the default) and for the bottom interface as external:

```
# Signaling parameters
Signaling card=5 connector=top protocol=NONE  clock=Int
Signaling card=10 connector=top protocol=NONE  clock=Ext
```

Transmit clock can be toggled temporarily using the ATM card’s **maint 22** command. The setting reverts back to the recovered receive clock (internal) at ATM card reboot and system reset.

Using the **maint 22** ***port value*** command, you can set the top interface’s transmit clock to external oscillator. Specify ***value*** as 1:

```
GR 06> maint 22 0 1
```

To set the top interface’s transmit clock back to the default (internal, recovered receive clock), specify ***value*** as 0:

```
GR 06> maint 22 0 0
```

These **maint** settings are *temporary*, and revert back to recovered receive clock (0) at ATM card reboot and system reset.

Create and assign broadcast groups

The ATM OC-12c media card uses standard broadcast IP group addressing.

Broadcast addresses are entered in the Service section of the `/etc/gratm.conf` file. The media card’s transmit interface routes broadcast datagrams to each of the members of the broadcast group defined in Service `type=bcast`. Here is an example that also shows broadcast group assignment in the Interfaces section:

```
# Broadcast Service info
Service name=bc0 type=bcast addr=198.174.20.1 addr=198.174.22.1 \
       addr=198.174.21.1
#Interfaces
Interface ga090 service=bco traffic_shape=high_speed_high_quality
```

Verify the broadcast groups with **maint 3 3 *port***:

```
maint 3 3 1
[RX] Port 0: LINK UP
[RX] Port 1: LINK UP
[TX] IF  GROUP  BROADCAST GROUP MEMBERS
-----------------------------------------------------------------------
ga092 bc0 198.174.20.1 198.174.22.1 198.174.21.1
```

Configuring traffic shapes

Peak cell rate, sustained cell rate, and maximum burst size are specified to create a `Traffic_Shape name` in the Traffic Shaping section of the `/etc/gratm.conf` file.

A name can be any string, for example, this shape specifies the best possible service and access to bandwidth resources:

```
Traffic_Shape name=high_speed_high_quality  \
        peak=155000 sustain=155000 burst=2048 qos=high
```

Use a backslash (\) to divide a single long line of characters.

This shape specifies a minimum level of service:

```
Traffic_Shape name=lowest_speed_lowest_quality  \
             peak=10000 burst=64 qos=low
```

Note: Sustained rate defaults to peak cell rate when it is not specified.

You can create as many `Traffic_Shape` names as you need, but you can specify only eight different peak rate queues. At most, there can be four peak rate queues for high QoS, four for low QoS. You cannot borrow from one to increase the other.

- The maximum peak rate is 155520 kilobits/second.
- The maximum sustained rate is 155000 kilobits/second.
- The largest burst size you can specify is 2048 cells.

Peak rate is the only required parameter in a `Traffic_Shape`. If you do not specify a sustained rate, it defaults to the peak rate. If you do not specify a burst size, it also defaults to peak rate. Another optional parameter is Quality of Service (`QoS`). Quality of Service defaults to high priority.

PVCs and logical interfaces are individually assigned a specific `Traffic_Shape name`. An SVC inherits the `Traffic_Shape name` of the logical interface to which it is assigned.

Optional: set parameters in the Card profile

Set optional ATM card configuration parameters at the Card profile. Available options are:

– OPTIONAL: specify ICMP throttling settings

– OPTIONAL: change run-time binaries

– OPTIONAL: change dump variables

Media card type, `atm-oc12-v1,` is automatically read into the read-only `media-type` field. Other values shown are defaults. At the top level, you see config and ICMP throttling fields:

```
super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
media-type = atm-oc12-v1
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = <{ 0{off on 10 3} {single off} {"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off}
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

1. Specify ICMP throttling

You can change default ICMP throttling settings in the `icmp-throttling =` field. ICMP throttling messages are described earlier in the chapter or do a `set <field-name>?` for a brief description.

Default values are shown here:

```
super> list icmp
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

Here is how to access the help message for the `echo-reply` field:

```
super> set echo ?
echo-reply:
 The number of ICMP ping responses generated in 1/10 second.
Numeric field, range [0 - 2147483647]
```

Change default echo reply and TTL settings with these commands:

```
super> set echo-reply = 4
super> set TTL-timeout = 12
super> write
CARD/8 written
```

You do not have to do a **write** until you have finished all changes in the Card profile. However, you get a warning message if you try to exit a profile without saving your changes.

2. Specify a different executable binary

Card-specific executables can be set at the Card profile in the `load` / `hw-table` field. The `hw-table` field is empty until you specify the path name of a new run-time binary. This specified run-time binary will execute in this ATM OC-12c card only.

```
super> read card 8
card/8 read

super> list load
config = 0
hw-table = < >
boot-seq-index = 1
boot-seq-state = 0
boot-seq-diagcode = 0
```

If you want to try a test binary, specify the new path in the `hw-table` field:

```
super> set hw-table = /usr/libexec/portcard/test_executable_for_ATM12
super> write
CARD/8 written
```

3. Change default dump settings

Card-specific dump file names can be set at the Card profile in the `dump` / `hw-table` field. The `hw-table` field is empty until you specify a new path name.

```
super> read card 8
card/8 read
super> list dump
config = 0
hw-table = < >
config-spontaneous = off
dump-on-boot = off
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex. Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
CARD/8 written
```

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

To save the `/etc` configuration directory, use **grwrite**:

```
# grwrite -v
```

Additionally, when you enter configuration information or make changes, you must also reset the media card for the change to take place. Enter:

Optional: change ATM binaries – Load profile

Global values for executable binaries are set at the Load profile in the `hw-table` field. These only change when you want to execute new run-time code in **all** ATM cards.

Here is the path, default settings are shown:

```
super> list
super> read load
LOAD read
super> list
hippi = {"" N/A on 0 1 <{1 /usr/libexec/portcards/xlxload.run N/A} {2 /u+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/portcards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/portcards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/portcards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/portcards/+
```

Look at the ATM OC-12c card settings:

```
super> list atm-oc12-v1
type = atm-oc12-v1
rx-config = 0
rx-path = /usr/libexec/portcards/atmq_rx.run
tx-config = 0
tx-path = /usr/libexec/portcards/atmq_tx.run
enable-boot-seq = off
mode = 0
iterations = 1
boot-seq-table = < >
```

To execute different run-time code on the receive side of the ATM OC-12c card, replace `/usr/libexec/portcards/atmq_rx.run` with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/newatmq_rx.run
super> write
LOAD written
```

You can also enable a diagnostic boot sequence using the `enable-boot-seq` field. In the default boot sequence, a media card boots, its executable run-time binaries are loaded, and the card begins to execute that code. You have the option to configure the card's boot sequence so that after booting, the card loads and runs diagnostics before it loads and runs the executable binaries. Set the `enable-boot-seq` field to on and use **write** to save the change:

```
super> set enable-boot-seq = on
super> write
LOAD written
```

You can also use the **grdiag** command to run a set of hardware diagnostics on the media card. Refer to the "Management Commands and Tools" chapter in this manual for information.

Optional: change ATM dumps – Dump profile

Global values for dump settings are at the Dump profile. These settings are usually changed only for debug purposes. Default settings are shown in this example.

The `keep-count` field specifies how many dumps are compressed and stored at one time for each media card. The file system accomodates the default setting of 2 which actually stores two dumps per day in addition to the current dump and the first dump of the day. Use caution if you change the default.

Here is the path, default settings are shown:

```
super> read dump
DUMP read

super> list
hw-table = < { hippi 20 var 0 } { rmb 20 var 3} { hssi 20 var 7 }+
dump-vector-table = <{{ 3 rmb "RMB default dump vectors" < { 1 SRAM 2621+
config-spontaneous = off
keep-count = 2
```

The `hw-table` field has settings to specify when dumps are taken and where dumps are stored. Here is the path to examine the ATM OC-12c settings:

```
super> list hw-table
hippi = { hippi 20 /var/portcards/grdump 0 }
rmb = { rmb 20 /var/portcards/grdump 3 }
hssi = { hssi 20 /var/portcards/grdump 7 }
dev1 = { dev1 20 /var/portcards/grdump 9 }
atm-oc3-v2 = { atm-oc3-v2 20 /var/portcards/grdump 5 }
fddi-v2 = { fddi-v2 20 /var/portcards/grdump 6 }
atm-oc12-v1 = { atm-oc12-v1 20 /var/portcards/grdump 10 }
ethernet-v1 = { ethernet-v1 20 /var/portcards/grdump 8 }
sonet-v1 = { sonet-v1 20 /var/portcards/grdump 11 }

super> list atm-oc12-v1
media = atm-oc12-v1
config = 20
path = /var/portcards/grdump
vector-index = 10
```

In the `config` field you can specify when dumps will be taken. The setting is the sum of one or more values, expressed in hex.

Here are the values used:

0x0001 - dump always (override other bits)
0x0002 - dump just the next time it reboots
0x0004 - dump on panic
0x0008 - dump whenever reset
0x0010 - dump whenever hung
0x0020 - dump on power up

The setting `config = 14` is the sum of 0x0004 (dump on panic) and 0x0010 (dump whenever hung) expressed in hex.

The setting `config = 20` is the sum of 0004, 0008, and 0020: dump during panic, reset, and power up (you sum to obtain 0x20).

```
super> set config = 14
super> write
DUMP/ written
```

Dump vectors

The `segment-table` fields in the `dump-vector-table` describe the areas in core memory that will be dumped for all ATM OC-12c cards. These parameters are read-only, they cannot be changed.

Here is the path, **cd** .. back up to the main level if necessary:

```
super> cd ..
super> list dump-vector-table
3 = {3 rmb "RMB default dump vectors" < { 1 SRAM 262144 524288 } > }
5 = {5 atm-oc3-v2 "ATM/Q default dump vectors" < {1 "atm inst memory" 16+
6 = {6 fddi-v2 "FDDI/Q default dump vectors" < { 1 "fddi/Q CPU0 core mem+
7 = {7 hssi "HSSI default dump vectors" < { 1 "hssi rx SRAM memory" 2097+
8 = {8 ethernet-v1 "ETHERNET default dump vectors" < {1 "Ethernet rx SRA+
9 = {9 dev1 "DEV1 default dump vectors" < { 1 "dev1 rx SRAM memory" 2097+
10 = {10 atm-oc12-v1 "ATM OC-12 default dump vectors" < {1 "ATM-12 SDRAM+
11 = {11 sonet-v1 "SONET default dump vectors" < {1 "SONET rx SRAM memor+
```

This sequence shows a portion of the areas in the ATM OC-12c card that are dumped:

```
super> list 10
index = 10
hw-type = atm-oc12-v1
description = "ATM OC-12 default dump vectors"
segment-table = <{1 "ATM-12 SDRAM memory" 16777216 4194304}{2 "ATM-12 S+
super>

super> list seg
1 = { 1 "ATM-12 SDRAM memory" 16777216 4194304 }
2 = { 2 "ATM-12 SSRAM memory" 25165824 131072 }
3 = { 3 "SUNI Registers" 6291456 1024 }
4 = { 4 "Tx CTRL Regs" 33554432 16 }
5 = { 5 "Tx SAR Regs" 34603008 128 }
6 = { 6 "Tx DESC Memory" 35651584 131072 }
7 = { 7 "Rx CTRL Regs" 50331648 16 }
8 = { 8 "Rx SAR Regs" 34603008 128 }
9 = { 9 "Rx DESC Memory" 52428800 131072 }

super> list 1
index = 1
description = "ATM-12 SDRAM memory"
start = 16777216
length = 4194304

super> list seg 2
index = 2
description = "ATM-12 SSRAM memory"
start = 25165824
length = 131072
```

Getting media card statistics

This section describes the use of **maint**, **grarp -a**, **gratm**, and **grstat ip** commands to obtain ATM OC-12c card information.

maint commands for ATM OC-12c media cards

maint commands display a range of information about a specific type of media card. Each media card has its own set of **maint** commands. The same **maint** command may work on more than one media card.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

```
# grrmb
```

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the ATM media card you are working with. For example, if you are working with a card in slot 4, enter:

```
GR 66> port 4
```

This message is returned along with the changed prompt:

```
Current port card is 4
GR 4>
```

To leave the **maint** prompt, enter **`quit`**.

List of maint commands

Use **maint 1** to see the list of **maint** commands.

```
# grrmb
GR 4> maint 1
GR 4> new   (old)
 1:   (999) Display this screen of options
 2:  (1203) Display version numbers
 3:  (1105) Display Interface configuration
 4:  (1104) Display VC statistics [vpi vci]
 5:         Display SWITCH statistics
 6:  (1201) Display COMBUS statistics
 7: (13,23) Clear counters (may mess up SNMP)
 8:  (1106) Display ARP Table
 9:  (1107) Display ARP Server info
 10: (1202) Display MEMORY usage
 11:        Display packet counters
 12:   (22) Display error counters
 13: (1103) Display VC configuration
 14:   (14) Trace control [on=1/off=0]
 15:   (15) Display history trace (number of entries)
```

```
16:        Display interrupt counters
18: (1108) Display Broadcast Groups
19:        Display IS-IS multicast list
20: (1100) Display SUNI
21:    (7) Select SDH/SONET (SONET=0, SDH=1)
22:    (8) Select SUNI timing source (0= internal 1= external)
23:    (9) Select SUNI local loopback (0= off 1= on)
24:   (10) Select SUNI line loopback(0= off 1= on)
```

Display s/w and h/w version data

The **maint 2** command displays the software and hardware version information:

```
GR 4> maint 2
GR 4>   Code Version: A1_4_12R_3
        Compiled in: /nit/A1_4_12R_3/atm-12,
                 on: Wed Sep 23 02:46:07 CDT 1998.
GRIPV4 Library Version: 1.4.12R.3, Compiled on Wed Sep 23 02:30:28 CDT
1998.
Board rev: 3 FPGA rev: 1
DIP switch settings: 1=ON 2=ON 3=ON 4=ON
```

Display the interface configuration

The **maint 3** command displays configuration information for each interface on the card. `Index` refers to the interface index. `ARP-S` is the number given the interface's associated ARP server. `BCST-G` is the number for a broadcast group the interface may be assigned to.

```
GR 4> maint 3
GR 4>
IF ARP-S BCST-G     IP        Index                NSAPA
-----------------------------------------------------------------------
07  00    00   227. 1.14.153     53
06  00    00   227. 1.13.153     52
05  00    00   223.3.14.153      51
04  00    00   224.101.12.153    50
03  1     00   227.101.10.153    57
02  00    00   224.101.13.153    56
01  00    00   223.3.13.153      55
00  00    00   223.3.12.153      54
```

Display virtual circuit statistics

The **maint 4** command displays statistics per VC. Note that `rq` is the assigned rate queue, values are 0–15.

```
GR 4> maint 4
                        VC statistics
               output                         input
 CID Vpi  Vci  rq    pkts     errs  aal pro    pkts     errs
-------------------------------------------------------------
  50 000 0050  00 0000001797 000000  5  IPL 0000000000 000000
  51 000 0051  00 0000001797 000000  5  IPL 0000000000 000000
 100 000 0100  00 0000001797 000000  5  IPL 0000000000 000000
 101 000 0101  00 0000001797 000000  5  IPL 0000000000 000000

  RX ATM Cells received: 209992530  Cells discarded: 5  Errors: 0
  TX ATM Cells transmitted = 1107727945
  RX FPP Packets dropped: 0
  TX FPP Packets dropped: 5343975 selectively, 73 due to buffer
  exhaustion.
  TX packets lost: 96961
```

Display combus statistics

Use **maint 6** to monitor the amount of traffic and error messaging between the ATM OC-12c media card and the communications bus.

```
GR 4> maint 6
GR 4>
Combus Status:
        Last interrupt status:           0x0
Combus Statistics:
        Message ready interrupts:            533266
        Truncated input messages:               383
        Grit messages for TX-CPU:                 0
        Ip messages Rcvd (non-bypass):            0
        Raw messages:                             0
        ISO messages:                             0
        Grid messages:                       533266
        Grid echo requests:                   30764
        Port available messages:                  0
        Segmented Packets:                        0
        Segments Sent:                            0
Combus Errors:
        Bus in timeouts:         0      Bus out timeouts:          0
        Out of buffer cond.:     0      Bad packet type:           0
        Dropped IP packets:      0      Bad packet dest:           0
        Receive Msg Errors:      0      Receive Format Errors:     0
        Receive Past End:      383      Received Long Message:   544
```

Display ARP information

Use **maint 8** to look at ARP addresses per VPI/VCI**.** A row of question marks indicates the field is not applicable.

```
GR 4> maint 8
    IP        IF  ST                NSAPA                       VPI/VCI
----------------------------------------------------------------------
???.???.???.??? 01 PERM  ???????????????????????????????????????  0/101
???.???.???.??? 00 PERM  ???????????????????????????????????????  0/100
223.3.14.153 01 PERM 0X47.0005.80ffe1000000f21513eb.0020481513eb.00 0/51
???.???.???.??? 00 PERM  ???????????????????????????????????????  0/50
```

The **maint 9** command returns address and status information about the ARP server:

```
GR 4> maint 9
ARP-S            NSAPA                                  STATE
---------------------------------------------------------------------
00:   0X47.0005.80ffe1000000f21513eb.0020481513eb.00  0/51
      ------                                          Not Configured
      ------                                          Not Configured
```

Note: Use **grarp -a** to obtain comprehensive ARP information. An example of **grarp -a** output is at the end of this section.

Display memory usage

The **maint 10** command displays memory information. The number of total and free buffers are provided. There are three lines:

- TB-MEM = allocatable table memory, units are 4-byte words, a low number indicates that there may not be enough resources to configure more interfaces.
- TBL-BF = COM bus receive buffers, very high number indicates the combus is processing many error messages for this card.
- VCT-0 = the number of VC table entries, if the number changes significantly, there may be a loss of table entries.

The UNIT column shows which memory unit is being described, the 4-byte table memory (for media card tables), or the 616-byte table memory buffers (for Combus usage).

TOTAL shows how much is available after fixed requirements.

FREE shows what is currently allocatable.

N-FRAGs is the number of areas of contiguous free memory.

```
GR 4> maint 10
TYPE     UNIT     TOTAL     FREE     N-FRAGS   LRG-FRAG   BUSY
-----------------------------------------------------------------
TB-MEM   00004    0417927   0236997  000002    0236997    -------
TBL-BF   00616    0000010   0000009  ------    -------    -------
VCT-0    -----    0001408   0001404  -------   -------    0000004
```

Display packet traffic counts

The **maint 11** command returns the number of received and transmitted packets.

```
GR 4> Receive packet activity:
0002749114 Total IPv4 Fast forward actions
0000254961 Received IS-IS
1827006901 Other software queued packets

Transmit packet activity:
-249882644 Automatically forwarded by hardware
0000000002 IP Queued to SAR by software
0000000010 Local ping copied to Rx and routed
```

Display VPCI configuration

Use **maint 13** to return information about VPI/VCIs on a per port basis.

Interface `IF` is always 0 on the ATM OC-12c card.

`FPCR` is forward peak cell rate.

`FSCR` is the forwarding sustained cell rate.

`FMBS` is the forwarding maximum burst size.

```
GR 4> maint 13
IF VPVCI TYP FPCR   FSCR   FMBS           DEST
------------------------------------------------------------------------
00 0/032 pvc 622080 622080 000255 IP pt-pt
00 0/033 pvc 622080 622080 000255 IP pt-pt
00 0/034 pvc 622080 622080 000255 IP pt-pt
00 0/035 pvc 622080 622080 000255 IP pt-pt
00 0/036 pvc 622080 622080 000255 IP pt-pt
00 0/037 pvc 622080 622080 000255 IP pt-pt
00 0/038 pvc 622080 622080 000255 IP pt-pt
00 0/039 pvc 622080 622080 000255 IP pt-pt
00 0/040 pvc 622080 622080 000255 IP pt-pt
00 0/041 pvc 622080 622080 000255 IP pt-pt
00 0/042 pvc 622080 622080 000255 IP pt-pt
00 0/043 pvc 622080 622080 000255 IP pt-pt
00 0/044 pvc 622080 622080 000255 IP pt-pt
```

Display broadcast groups

If broadcast groups are configured, the **maint 18** command returns the list of members in each group. `GRP`, group, is given a number, members are defined in a list of IP addresses.

```
GR 4> maint 18
GRP                                  MEMBERS
--------------------------------------------------------------------------
2             227.101.14.153 223.3.14.153 223.3.13.153 223.3.12.153
```

Setting parameters

A set of **maint** commands are available to select SDH/SONET, SUNI timing source, local and line loopback:

- select SONET = **maint 21 0**
- select SDH = **maint 21 1**
- select internal SUNI timing source = **maint 22 0**
- select external SUNI timing source = **maint 22 1**
- put SUNI local loopback on = **maint 23 1**
- set SUNI local loopback to off = **maint 23 0**
- put SUNI line loopback on = **maint 24 1**
- set SUNI line loopback to off = **maint 24 0**

Use grarp -a to display ARP addresses

The **grarp -a** command displays the contents of the GRF system ARP table.

```
# grarp -a
box1.minnet.com (206.146.160.1) at 0:c0:80:89:15:e5
box2.minnet.com (206.146.160.131) at (incomplete)
box3.minnet.com (206.146.160.132) at 0:a0:24:a3:c:36
box4.minnet.com (206.146.160.133) at 0:c0:80:86:14:e2 permanent
box5.minnet.com (206.146.160.202) at 0:e0:1e:5d:a4:7f
ga040   (13): 208.1.10.158 at VPI=0, VCI=150 permanent
ga027f  (73): 10.20.2.237 at VPI=15, VCI=511 permanent
ga020   (66): 10.20.2.233 at VPI=0, VCI=32767 permanent
ge035   (12): 10.20.1.182 at 0:60:70:ac:38:20
ge035   (12): 10.20.1.131 at 0:c0:80:8f:8a:3
#
```

Use grstat ip to look at layer 3 statistics

Layer 2 statistics are not available from **grstat** for ATM OC-12c cards.

```
# grstat ip ga04
card 4 (1 interfaces found)
  ipstat totals
       count description
          14 total packets received
          14 packets forwarded to the RMS
  ipdrop totals
       count description
```

Use grrt to look at next hop data

Here are a few lines of **grrt** output:

```
# grrt -S -p 1
default                          3    0.0.0.0         RMS      UNREACH
0.0.0.0         255.255.255.255  7    0.0.0.0         RMS      DROP
127.0.0.0       255.0.0.0        5    0.0.0.0         RMS      UNREACH
127.0.0.1       255.255.255.255  2    127.0.0.1       RMS      RMS
198.174.11.0    255.255.255.0    6    206.146.160.1   RMS      RMS
203.1.10.156    255.255.255.255  77   0.0.0.0         gf0d2    LOCAL
203.1.10.255    255.255.255.255  76   0.0.0.0         gf0d2    BCAST
203.3.10.0      255.255.255.0    68   0.0.0.0         gf081    FWD
203.3.10.156    255.255.255.255  67   0.0.0.0         gf081    LOCAL
203.3.10.255    255.255.255.255  66   0.0.0.0         gf081    BCAST
203.3.11.0      255.255.255.0    55   0.0.0.0         gh0a0    FWD
203.3.11.156    255.255.255.255  49   0.0.0.0         gh0a0    LOCAL
203.3.11.255    255.255.255.255  46   0.0.0.0         gh0a0    BCAST
203.3.12.0      255.255.255.0    37   204.101.10.158  ge070    FWD
203.3.13.0      255.255.255.0    57   208.1.12.158    go060    FWD
```

Collect data via grdinfo

With a single command, **grdinfo** collects the output from nearly all of the ATM OC-12c **maint** commands and compresses it in a log file. Refer to the “Management Commands and Tools” chapter in this manual for more information.

ATMP Configuration Guide

12

Chapter 12 introduces the user to the implementation of the Ascend Tunnel Management Protocol, ATMP, on the GRF router. Configuration information for each feature and several configuration examples are included.

Chapter 12 covers these topics:

Introduction to ATMP

ATMP (Ascend Tunnel Management Protocol) is a layer 3 UDP/IP-based protocol that provides a cross-WAN (Internet or other) tunnel mechanism using standard Generic Routing Encapsulation between two Ascend units. ATMP is described in RFC 2107 and also in an Ascend publication, *Ascend Tunnel Management Protocol*.

Generic Routing Encapsulation (GRE) hides packet contents and enables transmission of packets that would otherwise be unacceptable on the Internet, such as IP packets that use unregistered (non-routable) addresses. GRE is described in RFCs 1701 and 1702.

The ATMP tunnel protocol creates and tears down the tunnel between two Ascend units. In effect, the tunnel collapses the Internet cloud and provides what looks like direct access to a home network. This manual describes a specific implementation in which one unit is a GRF IP switched router and the other unit is a TNT. Because the GRF receives packets through the tunnel which it must then route, ATMP on the GRF applies only to IP networks.

How ATMP connections work

Figure 12-1 shows the components that comprise an ATMP tunnel:

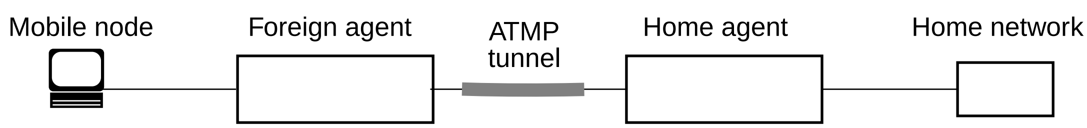


Figure 12-1. Components of a basic ATMP tunnel

These elements interact in an ATMP connection:

- Mobile node

  A mobile node is a user who accesses a “private” home network across the Internet.

  For example, a user could be a sales person on the road who wants to dial into a local ISP to log into his or her private corporate network.

- Foreign agent

  The foreign agent is an Ascend unit dialed by the mobile node. It is the starting point of the ATMP tunnel.

  Typically, the foreign agent is a TNT unit. The foreign agent first authenticates the mobile node using a RADIUS profile that includes ATMP parameters. Then the foreign agent brings up an IP connection to the home agent over which to negotiate the tunnel.

- Home agent

  The home agent is the termination point of the tunnel.

  The home agent must be able to communicate with the home network directly, across a dedicated WAN connection. A GRF or TNT unit acts as the home agent.

- Home network

  The home network is usually a “private” corporate network.

  A private network is one that cannot communicate directly on the Internet. It might be an IPX network, or an IP network with an unregistered network address.

Support for virtual private networks

Virtual private networks provide low-cost remote access to private LANs via the Internet. The tunnel to the private corporate network can be from an ISP to enable mobile nodes to dial-in to a corporate network. Tunnels can be established between two corporate networks to enable them to use a low-cost Internet connection to access each other. The Ascend GRF supports virtual private networking through the Ascend Tunnel Management Protocol (ATMP).

An ATMP session (tunnel) occurs between a GRF router and a TNT unit via UDP/IP. All packets passing through the tunnel are encapsulated in standard Generic Routing Encapsulation (GRE) as described in RFC 1701. ATMP creates and tears down a cross-Internet tunnel between the two units. In effect, the tunnel collapses the Internet cloud and provides what looks like direct access to a home network. Bridging is not supported through the tunnels. In the GRF ATMP implementation, all packets must be routed using IP.

Private address space

The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:

10.0.0.0 – 10.255.255.255	(10/8 prefix)
172.16.0.0 – 172.31.255.255	(172.16/12 prefix)
192.168.0.0 – 192.168.255.255	(192.168/16 prefix)

As described in RFC 1918, we refer to the first block as “24-bit block”, to the second as “20-bit block”, and to the third as “16-bit block.” In pre-CIDR notation, the first block is a single class A network number, the second block is a set of 16 contiguous class B network numbers, and the third block is a set of 256 contiguous class C network numbers.

These addresses are also referred to as non-routable or unregistered.

ATMP on the GRF

GRF routers support a subset of the Ascend Tunnel Management Protocol (ATMP) which enables the GRF to function as an ATMP home agent in IP gateway mode.

Feature summary

In this mode, the GRF home agent:

- authenticates a tunnel request from a foreign agent according to the ATMP specification. This authentication is between the foreign agent and the home agent, and does not authenticate the mobile node. The mobile node is first authenticated by the foreign agent.
- establishes a tunnel for a specified mobile node.
- strips encapsulation from IP traffic received across the tunnel from the mobile node and forwards it as normal IP traffic to the home network.
- encapsulates IP traffic received from the home network that is destined for the mobile node, and forwards the encapsulated IP traffic across the tunnel to the foreign agent.
- supports pre-fragmentation (fragmentation prior to encapsulation) or post-fragmentation (after encapsulation) of packets received from the home network
- can use RIPv2 to advertise mobile node routes to the home networks.
- supports Ethernet, HSSI Frame Relay, and ATM OC-3c as WAN connections to the home network and foreign agent.
- supports RFC 1483 VC-based multiplexing as well as ATM LLC SNAP encapsulation on ATM connections to the home network. RFC 1577 conformance provides Inverse ATM ARP support for LLC encapsulation.

Connection and routing media:

- Ethernet between the foreign agent and the GRF home agent (GRF)
- HSSI (Frame Relay) from foreign agent to home agent (GRF)
- HSSI (Frame Relay) from home agent (GRF) to home network
- ATM OC-3c from foreign agent to home agent (GRF)
- ATM OC-3c from home agent (GRF) to home network

These functions are not supported:

- The GRF does not perform ATMP IPX tunneling, or operate in ATMP IPX gateway mode (this item is noted because TNT ATMP does support IPX). Likewise, the GRF does not route any non-IP traffic, and drops any encapsulated non-IP packets (IPX, AppleTalk) received across a tunnel or from a home network.
- The GRF does not function as a foreign agent and does not perform session management or connection-level user authentication of a mobile node
- The GRF does not function as an ATMP home agent in IP router mode. The GRF does not install a host route to a mobile node at the time the tunnel is established or advertise itself as the next hop (gateway) for the mobile node on the public network.

GRF in gateway mode

The GRF can be configured as a home agent in gateway mode to a home network.

In ATMP gateway mode, the home agent has a tunneled circuit to the home network and passes packets received from the tunnel across the circuit to the associated home network router. Normal routed traffic does not use the tunneled circuit.

For a GRF home agent, the tunneled circuit is either a special HSSI Frame Relay permanent virtual circuit (PVCATMP) or an ATM OC-3c virtual circuit.

Scalability

The GRF home agent supports tunnel connections for up to 10,000 mobile nodes. The 10,000 tunnels can operate simultaneously. Up to 300 home networks can be configured using any combination of HSSI Frame Relay and ATM PVC interfaces.

Each home agent configured on the GRF requires a unique home agent address. A home agent address represents a single home network. The foreign agent sees this IP address and associates it with a single home network. To the foreign agent, it appears that there is one GRF system per home network. Actually, hundreds of home networks can connect to a single GRF since each home network is connected to a different internal home agent.

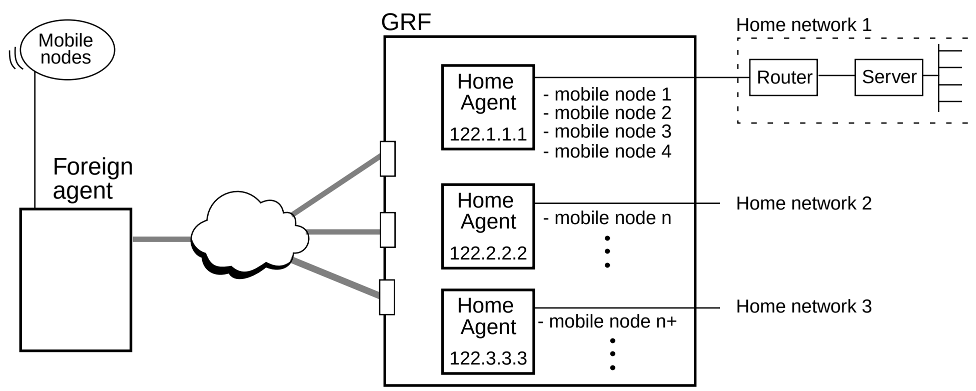


Figure 12-2. Support for multiple home agents on the GRF

Multiple home agents on a single GRF is supported by the `atmp0` software interface. The ATMP daemon, **aitmd**, manages this interface so that many IP addresses can be assigned to it. All packets coming from a tunnel “arrive” at `atmp0` where they are decapsulated and forwarded to the correct media card logical interface. The **netstat -i** command returns the list of `atmp0` IP addresses configured on a GRF system.

If a customer requires hundreds of home agents, GRF home agent configuration must be carefully organized. Connections to the 300 home networks should be configured according to the amount of bandwidth the mobile nodes may require.

Also, one logical interface (such as `gs030` or `ga028`) is consumed per home network. Each HSSI media card supports up to 128 home networks. Each ATM OC-3c media card supports up to 70 home networks.

GRF memory usage

On the media card, tunnel connection for each mobile node consumes memory comparable to space required by two routes. As additional tunnels are negotiated, less media card memory is available for route tables.

Interoperability

The GRF implementation interoperates with other equipment as defined in the official ATMP specification, RFC 2107, authored by Kory Hamzeh. Home agent modes (gateway, router) are described in other documentation from Ascend. ATMP is also supported by Ascend MAX and TNT products.

RIPv2 transmission

The ATMP daemon, **aitmd**, can be configured to multicast RIPv2 packets to the home networks attached to a GRF home agent. These packets advertise routes to each mobile node tunnel registered on the home agent. RIPv2 transmission enables a home network to learn the paths to its mobile nodes. This capability supports the configuration of a GRF as a secondary home agent.

RIPv2 transmission is enabled in the home network record of the `/etc/aitmd.conf` file. Participating home networks must also run RIPv2. RIPv2 parameters are described in the "Using the /etc/aitmd.conf parameters" section of this chapter.

LLC encapsulation

The ATM circuit from the GRF home agent to the home network can support LLC encapsulation. This enables the home agent to communicate with Frame Relay-attached devices via an ATM interface.

The circuit is configured as a PVC in `/etc/gratm.conf`. Assign the PVC a protocal value of `proto=llc_atmp`. Because this must be a dedicated connection, if an `llc_atmp` PVC is defined for a logical interface, no other PVCs can be defined on that interface.

If a VPN address is defined for the associated home network in the `/etc/aitmd.conf` file, then the interface will also support ATM Inverse ARP (as defined by RFC 1577) and will respond to Inverse ARP queries with its configured VPN IP address.

The ATM **maint 13** and **maint 113** commands display `llc_atmp` PVCs as ATMPLLC. This display is for the PVC configured in `/etc/gratm.conf` as:

```
PVC ga017f  14/16 proto=llc_atmp

GR 1> maint 13 0
GR 1> [RX]
[RX] IF  VPI/VCI  TYPE  AAL  ENCAPSULATION
------------------------------------------------------------------------
[RX] 00   0/512   pvc    5  IPLLC
[RX] 00   0/513   pvc    5  IPLLC
[RX] 7f  14/16    pvc    5  ATMPLLC
```

Null encapsulation

The ATM VC-based multiplex circuit from the GRF home agent to the home network can use null encapsulation. The circuit is configured as a PVC in `/etc/gratm.conf`. Assign the PVC a protocal value of `proto=vc_atmp`. Because this PVC is a dedicated connection, if a `vc_atmp` PVC is defined for a logical interface, no other PVCs can be defined on that interface.

The **maint 13** and **maint 113** commands display `vc_atmp` PVCs as ATMPNULL. This display is for the PVC configured in `/etc/gratm.conf` as:

```
PVC ga01ff 15/100 proto=vc_atmp

GR 1> maint 13 1
[RX]
[RX] IF  VPI/VCI  TYPE  AAL  ENCAPSULATION
------------------------------------------------------------------------
[RX] ff  15/100   pvc    5  ATMPNULL
[RX] 80   0/32766 pvc    5  IPNULL
```

Fragmentation options for encapsulated packets

The GRF home agent encapsulates packets received from the home network by adding the GRE header and a second IP header. However, the resulting packet may be too large to transmit as a single datagram on the public network interface over which the packet will be forwarded to the foreign agent. These packets must be fragmented by the GRF. Fragmentation is done either before or after a packet is encapsulated. By default, the GRF home agent will fragment the packet after encapsulation (post-fragmentation).

When the GRF performs post-fragmentation, the GRE header and the second IP header are prepended to the packet before it is fragmented. The resulting packet is then fragmented to fit on the transmitting interface. In this case, the foreign agent must reassemble the fragments. The foreign agent cannot decapsulate and deliver the individual fragments directly to the mobile node because only the first fragment contains the original IP header.

The user can specify in the `/etc/aitmd.conf` file that pre-fragmentation be used. When the GRF performs pre-fragmentation, the original packet is broken into suitable fragments, each with it own IP header, prior to encapsulation. Then, the encapsulation process prepends a GRE header and a second IP header to each fragment before its transmission to the foreign agent. Pre-fragmentation enables the foreign agent to decapsulate and forward each fragment directly to the mobile node. The mobile node then reassembles the fragments.

A GRF home agent does not pre-fragment and post-fragment the same packet. If the user enables pre-fragmentation, then the GRF home agent will always pre-fragment packets if necessary.

Pre-fragmentation on the GRF behaves similarly to the MAX and TNT home agent implementation. Configuration options use similar semantics as the MAX and TNT profile parameters.

Fragmentation parameters are described in the “Using the /etc/aitmd.conf parameters” section.

Tunnel operations

This section provides more information about tunnel components and operations.

Life-cycle of a tunnel

Here is a typical scenario that describes how an ATMP tunnel is requested, established, used, and torn down between a TNT foreign agent and a GRF home agent. Figure 12-3 illustrates the components and connections involved:

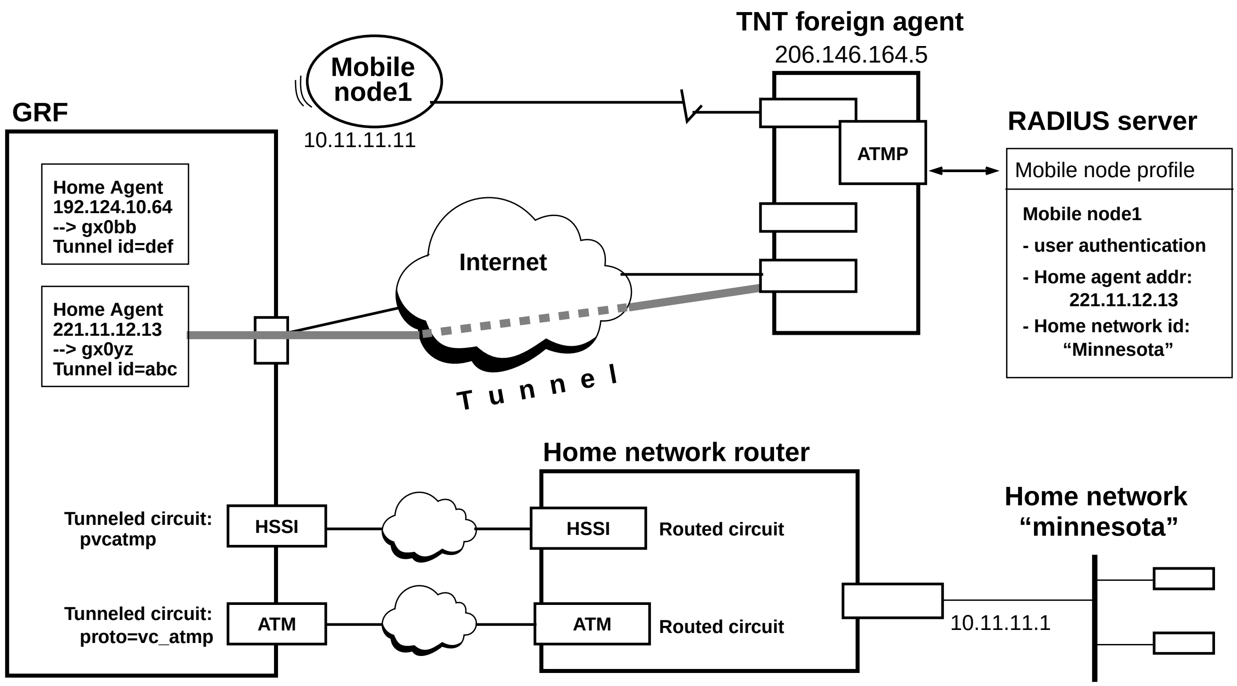


Figure 12-3. GRF home agent connections to foreign agent and home network

Initiation by mobile node:

– A mobile node dials a connection to the TNT foreign agent.

– The TNT foreign agent authenticates the mobile node using a RADIUS profile. (RADIUS authentication of the mobile node is used because the required attributes are supported only in RADIUS.)

Foreign agent tunnel negotiation:

– The foreign agent determines which home agent is the gateway to the target home network based on the Ascend-Primary-Home-Agent parameter in the mobile node's RADIUS profile. On the GRF, this is called the home agent address and is given as 221.11.12.13 in the example above. Using this address, the foreign agent sends a tunnel request message to that home agent.

– The GRF home agent requests a password for validation.

– The TNT returns an encrypted version of the Ascend-Home-Agent-Password found in the mobile node's RADIUS profile. This password must match the password that has already been configured in the GRF's `/etc/aitmd.conf` file for this particular foreign agent.

GRF home agent:

– If the password matches, the GRF ATMP daemon (**aitmd**) creates a tunnel ID and returns it to the foreign agent in a RegisterReply message. The tunnel ID is a number that uniquely identifies the tunnel. The GRF uses the tunnel ID to find the target home network.

At this point, a tunnel is created between the TNT and the GRF home agent.

– If negotiation fails, a message is logged and the TNT foreign agent disconnects the mobile node.

Termination:

– A tunnel is active as long as the mobile node is logged in to a node on the home network. The foreign agent can limit the session length if it is configured with a time limit option. When the mobile node disconnects from the TNT foreign agent, the TNT sends a DeregisterRequest to the GRF to close down the tunnel.

The foreign agent can send its request a maximum of ten times, or until it receives a DeregisterReply. If the foreign agent receives packets for a mobile node whose connection has been terminated, it silently discards the packets (no message to sender).

– If the circuit between the home agent and the home network goes down, the GRF notifies the foreign agent and terminates the tunnel.

Tunnel ID

aitmd creates and maintains a mobile node lookup table that contains the following for each mobile node:

- – mobile node IP address - usually non-routable
- – home network IP address - usually non-routable
- – tunnel ID
- – foreign agent IP address - routable
- – home agent IP address - home agent address, routable

aitmd assigns a unique ID to each tunnel and returns the ID to the foreign agent in the tunnel creation message (RegisterReply). When the tunnel is established, the home agent uses the tunnel ID to correctly encapsulate and forward packets received from the home network to the appropriate foreign agent. The mobile node's private network address is not used by the home agent because different mobile nodes on separate private networks may be using the same private address.

The **maint 70** and **maint 73** commands display home network and tunnel information. Slightly different `S:P:s0:s1` information is provided for Frame Relay HSSI and for ATM. The example below shows **maint** command information provided for Frame Relay.

First, use **maint 70** to list the home networks connected to a media card:

```
[RX] H O M E  N E T W O R K   T B L  :
[RX] =================================
[RX]
[RX] S: Slot, P: Port, Rx: packets Received, BRx: Bytes Received
[RX]                  RTx packets transmitted, BTx: Bytes transmitted
[RX]
[RX]FRT-index  S:P:s0:s1   State    Address       VPN Address  VPN Netmask
[RX]--------   ---------   -----    -------       -----------  -----------
[RX]   6     2:01:888:0  LocCir,HA  221.11.12.13  10.11.11.237 255.255.255.0
```

The columns are defined as follows:

- **FRT-index**:
  Foreign agent Route Table, an arbitrarily-assigned tunnel number, not the tunnel ID, but the number you use in the **maint 73** command to display the tunnel ID
- **S:P:s0:s1 on Frame Relay (HSSI card)**:
  S = slot, P = port, s0 = DLCI number of the tunneled circuit to the home network (`pvcatmp`), s1 is currently unused and is always 0
- **S:P:s0:s1 on ATM**:
  S = slot, P = port, s0 = ATM VPI, s1 ATM VCI (`vc_atmp`)
- **State**
  indicates configuration and functional status depending upon the type/role of card.
  - – On the home agent card (usually an Ethernet card acting as the home agent)

  The foreign agent uses an IP address on this card for tunnel negotiation. A state of `HomeAgent` indicates a home agent is properly configured to this card in `aitmd.conf`.

- On any HSSI or ATM card on a GRF home agent, `State` can have three values:

  1. `HomeAgent` indicates that the home agent is properly configured in `/etc/aitmd.conf`, but it is not the actual circuit to the home network.
  2. `LocalCirc` indicates that this circuit is not part of ATMP configuration.
  3. `LocCir,HA` = the circuit to the home network is configured on this media card.

- **Address** is the associated home agent address, same as the Ascend-Primary-Home-Agent parameter in the mobile node's RADIUS profile.
- **VPN Address** is the private network address the customer assigns to the interface that has the circuit to a home network, it only appears if entered in the `/etc/aitmd.conf` file.
- **VPN Netmask** is the netmask for the VPN address.

Please ignore the "`Rx: packets Received, BRx: Bytes Received`" and "`RTx packets transmitted, BTx: Bytes transmitted`" headers, they no longer apply.

Then use **maint 73 *FRT_index*** command to display tunnel information "toward" the foreign agent, including tunnel IDs:

```
GR 2> maint 73 6
GR 2>
[RX] Mobile node tree list for home network index 6
[RX]  Mobile Node /Mask Flags Foreign Agent Tunnel Id  S:P:s0:s1
[RX]  10.11.11.11 /16   0  => 206.146.164.5 0x00000503 02:01:888:0000
```

The columns are as follows:

- mobile node non-routable IP address
- number of bits in the address netmask
- the route flags column is not currently used
- foreign agent routable IP address
- tunnel ID, in this case, 0x503
- **S:P:s0:s1 on Frame Relay (HSSI card):**
  S = slot, P = port, s0 = DLCI number of the tunneled circuit to the home network (`pvcatmp`), s1 is currently unused and is always 0
- **S:P:s0:s1 on ATM:** S = slot, P = port, s0 = ATM VPI, s1 ATM VCI (`vc_atmp`)

IP packets and GRE

Generic routing encapsulation (GRE) supports virtual private networks by extending connectivity to the non-routable IP address class. Mobile units with non-routable addresses can access home networks which are also configured with non-routable addresses because foreign and home agents use GRE to transmit their otherwise unroutable packets over WAN-based tunnels. GRE hides packet header contents, and enables transmission of packets that the Internet would otherwise not accept. These include IP packets from roaming clients that use unregistered addresses. The GRF ATMP implementation only supports encapsulated IP packets.

As shown in Figure 12-4, the original IP packets are encapsulated with two headers. The mobile node sends an IP packet with its private network address as the source and the home network server address as the destination. The foreign agent first adds a GRE header containing the tunnel ID it received from the home agent. Then the foreign agent adds a second IP header with its address as the source and the home agent address as the destination.

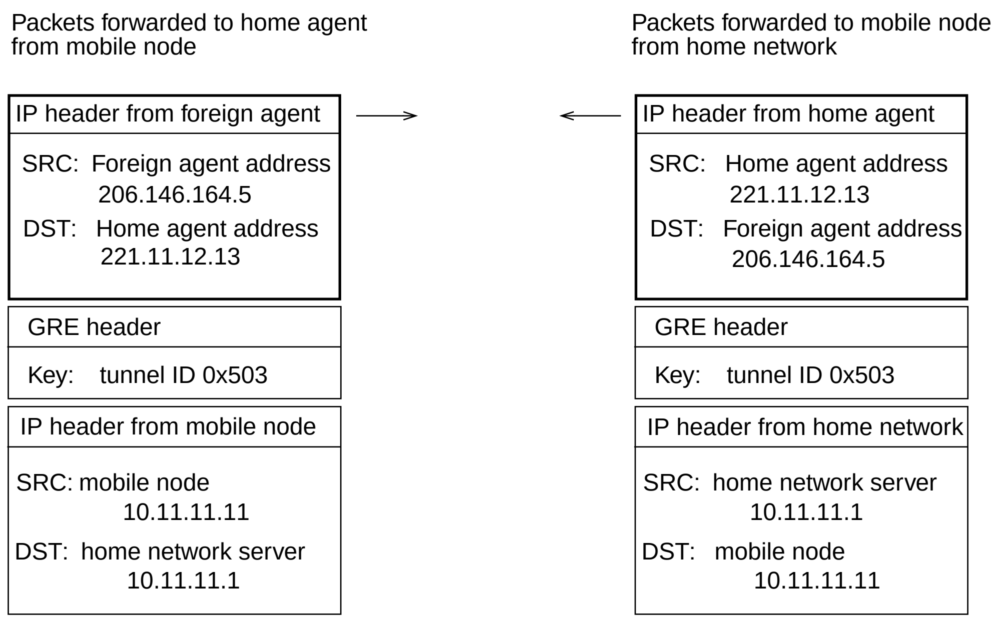


Figure 12-4. Contents of GRE packet headers

The home network server forwards IP packets with the destination as the mobile node's private network address across the tunneled circuit to the GRF home agent. The GRF encapsulates those private network IP packets with a GRE header containing the tunnel ID. Then a second IP header is added with the foreign agent address as the destination and the home agent address as the source.

The packets transit the tunnel to the foreign agent. After the foreign agent strips off the outer IP and GRE headers, the IP packets continue through the dial-up connection to the mobile node.

Tunnel addressing and connections

This section describes the addresses and connections required in GRF ATMP configuration. Figure 12-5 shows where addresses exist, this section describes how they are related and the role each plays in tunneling.

Home agent addresses

The mobile node points to its home network with a home agent address that is in the mobile node's RADIUS profile. The foreign agent uses the home agent address to negotiate a tunnel and then as the destination address in the IP headers it pre-appends to packets coming from the mobile node. The home agent address is used by the ATMP daemon, **aitmd**.

aitmd maintains a table of the home agent addresses, one address for each home network attached to the GRF. Each home agent address points to a specific GRF interface that connects to the associated home network. In other words, each home network is "represented" by a different home agent address.

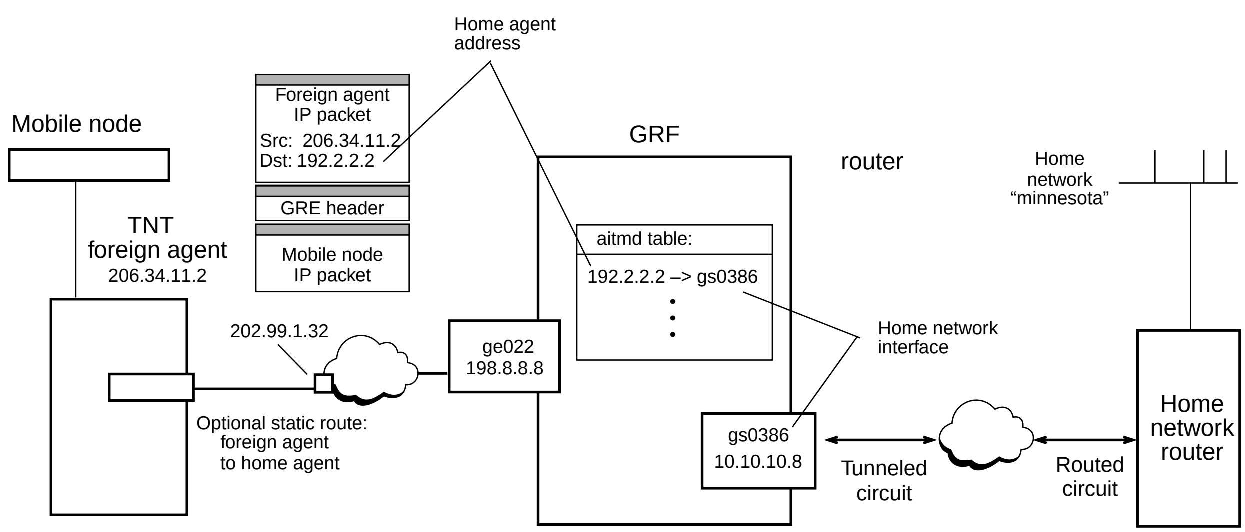


Figure 12-5. Addresses used in ATMP configuration

The foreign agent knows its interface to the home agent through a public address. The home agent knows of the foreign agent through a public address. Routes between the agents may be statically configured or obtained through a routing protocol such as OSPF.

The home network knows its interface to the home agent through a private network address. The home network router requires a routed circuit to the home agent. This circuit is described in the "Connection from home network router to home agent" section.

Foreign agent connection to home agent

Typically, the home agent address is publicly advertised by a routing protocol such as OSPF, and the foreign agent can obtain a route to the home agent via dynamic routing advertisements. This is the recommended option for the foreign agent connection to the home agent.

If a foreign agent is not running a routing protocol, or for some other site-specific reason, you can create a static route to the GRF.

Static route to home agent

On the TNT foreign agent, you can configure a static route from the foreign agent to an interface on the GRF. The gateway-address parameter is the address of the next hop on the subnet leading to the public network.

When a tunnel is "built", it uses the home agent address. The home agent address and its netmask are entered in the dest-address and netmask parameters shown in this example from a TNT `new ip-route` configuration:

```
admin> new ip-route to-ha_minnesota
IP-ROUTE/to-ha_minnesota read
admin> list
[in IP-ROUTE/to-ha_minnesota]
name* = to-ha_minnesota
dest-address = 192.2.2.2/32
netmask = 255.255.255.255
gateway-address = 202.99.1.32
metric = 1
   •
   •
   •
```

Mobile node RADIUS profile addresses

The ATMP tunnel operates independently of the private network address of the requesting mobile node. A private IP network has an unregistered IP network address, and therefore cannot communicate directly on the Internet. Home network (virtual private network) address spaces do not mix with each other or with the normal, publicly-routed IP address space.

Even when mobile nodes in different private networks have exactly the same IP address, the home agent sends their packets to the correct home network. This is because the home agent address specified in the mobile node's RADIUS profile can only direct packets to a specific home network.

RADIUS profile

The mobile node's RADIUS profile has an Ascend-Primary-Home-Agent parameter that contains the home agent address for the node's assigned home agent. The profile also contains the name of the node's home network (Ascend-Home-Network-Name). The IP address of the mobile node itself is the Framed-Address parameter.

Here is an example of a RADIUS profile for mobile node XYZ running TCP/IP:

```
nodeXYZ Password="top-secret"
   Ascend-Metric=2,
   Framed-Protocol=PPP,
   Framed-Address=10.1.1.2,
   Framed-Netmask=255.255.255.0,
   Ascend-Primary-Home-Agent=192.2.2.2,
   Ascend-Secondary-Home-Agent=192.2.8.18,
   Ascend-Home-Network-Name = minnesota,
   Ascend-Home-Agent-Password="TntCodexyz",
   Ascend-Home-Agent-UDP-Port = 5150,
   Ascend-Idle-Limit = 20
```

The home agent address of the primary (or only) home agent assigned to mobile node XYZ is specified in the Ascend-Primary-Home-Agent parameter. In the example shown in Figure 12-5, the primary home agent address is `192.2.2.2`. When the foreign agent encapsulates packets coming from the mobile node, the agent uses that IP address as the destination address in its IP header.

The home agent address of the secondary home agent assigned to mobile node XYZ is specified in the Ascend-Secondary-Home-Agent parameter. In this example the secondary address is `192.2.8.18`. When the foreign agent detects that the primary home agent is down, it references this address to negotiate a new tunnel.

The Ascend-Primary-Home-Agent parameter from the mobile node RADIUS profile must match the home network `home_agent_addr` entry in the `/etc/aitmd.conf` file.

Here is an excerpt from `/etc/aitmd.conf` showing the configuration illustrated in Figure 12-5 and reflected in the RADIUS profile shown just above:

```
home_network  {
   name minnesota;                # text string name. no more than 31 characters
   home_agent_addr 192.2.2.2;     # IP address of home agent on the GRF
                                  # Only one home_network may use this address
   interface {
      name gs0386;                # HSSI card in slot 3, port 1
      vpn_addr 10.10.10.8;        # VPN address
      vpn_netmask_size 26;        #  bits in VPN netmask
            •
            •
            •
```

Use the **netstat -i** command to display the home agent addresses associated with a specific GRF system. A sample display is on the next page.

Refer to the *MAX TNT RADIUS Configuration Guide* for more information about the mobile node RADIUS profile and ATMP parameters.

ATMP information from netstat -i

The **netstat -i** command returns a display from which you can verify several home agent and home network configuration parameters.

Interface to home network

Home agent address

The slot, port, DLCI or VPI/VCI numbers for the circuit to the home network

link

```
# netstat -i
Name    Mtu   Network       Address             Ipkts Ierrs    Opkts Oerrs  Coll
de0     1500  <link1>       00:c0:80:86:16:e2  627493     0   381326     0  7777
de0     1500  206.146.160 agent5               627493     0   381326     0  7777
rmb0    620   <link2>       00:00:00:00:00:00  587033 10649   564760     0     0
rmb0    620   <GRIT>        0:0x40:0           587033 10649   564760     0     0
lo0     1536  <link3>                            6139     0     6139     0     0
lo0     1536  <GRIT>        0:0x48:0             6139     0     6139     0     0
atmp0   1536  <link4>                              10     0       10     0     0
atmp0   1536  205.1.1.1          1,0,100,0         10     0       10     0     0
atmp0   1536  0/32          205.1.1.1              10     0       10     0     0
atmp0   1536  221.1.1.3           2,1,15,511       10     0       10     0     0
atmp0   1536  0/32          221.1.1.3              10     0       10     0     0
gl000   1496  <link5>                               0     0        0     0     0
gs000*  4352  <link33>                          57902     0    57901     0     0
gs017f  4352  <link15>                              0     0        0     0     0
gs017f  4352  <vpn3707830531> 10.20.2.221           0     0        0     0     0
ga02f1  9180  <link16>      00:c0:80:fa:cd:f1       0     0        0     0     0
ga02f1  9180  210.1.1       210.1.1.150             0     0        0     0     0
          .
          .
          .
```

Figure 12-6. ATMP entries as reported in the netstat -i display

The introductory entry in the list of ATMP (`atmp0`) entries always has an associated Link number. This is the SNMP interface index that is assigned when an interface is created:

```
atmp0   1536  <link4>                  10     0       10     0     0
```

An ATMP entry is a pair of lines. The nest four `atmp0` lines shown in the excerpt report on two home networks. The entry in the “Network” column contains the home agent address that is entered in the mobile node’s RADIUS profile. The “Address” entry on that same line contains slot, port, and DLCI or VPI/VCI information that identifies the physical interface to the home network. Although the specific interface name is not reported with the address information, you can use the `/etc/aitmd.conf`, `/etc/grfr.conf`, or `/etc/gratm.conf` files to verify the displayed configuration.

If you configure a VPN address under the interface name of the home network record in `/etc/aitmd.conf`, that private network address will be displayed. In this example, it is 10.20.2.221. A unique identifier for the private network, in this case, vpn3708730531, is also included, but it reflects internal information not useful for checking a configuration.

Note: An asterisk (*) indicates an inactive interface.

Connection from home agent to home network

The circuit (PVC) from the GRF home agent is through the home network interface to the home network and is configured as an ATMP gateway circuit.

- On a HSSI card, this circuit is defined as a PVCATMP in `/etc/grfr.conf`.
- On an ATM card, it is defined as a PVC in `/etc/gratm.conf`.

As far as the home network is concerned, this is a routed circuit. From the GRF point of view, it is a tunneled circuit. Traffic the GRF receives from the circuit is “tunneled” to the foreign agent (and hence the mobile node) associated with this home network.

The logical interface on which the tunneled circuit is configured must be identified in `/etc/grifconfig.conf`, but only by the interface name, do not assign an IP address. Here is the entry based on the example in Figure 12-7:

```
# /etc/grifconfig.conf
# name   address   netmask        broad_dest   arguments
gs0386 - - - up   # ATMP requirement=pvcatmp
```

You define the home network interface as part of the home network specification in the `/etc/aitmd.conf` file. This is where the interface name and VPN address are assigned:

```
home_network  {
   name minnesota;                # text string name. no more than 31 characters
   home_agent_addr 192.2.2.2;     # IP address of home agent on the GRF
                                  # Only one home_network may use this address
   interface {
      name gs0386;                # HSSI card in slot 3, port 1
      vpn_addr 10.10.10.8;        # VPN address
      vpn_netmask_size 26;        #  bits in VPN netmask
      ripv2 {
         enabled yes;             #  Send RIPv2 multicasts on this interface
         metric 2;                #  metric for advertised routes
         }
      }
```

Static route from home agent to foreign agent

A static route is recommended from the GRF home agent to the TNT foreign agent. Ascend also recommends it be configured in a GateD static statement. Use the TNT system address as the destination, in this example it is 206.34.11.2. The gateway is the address of the next hop on the subnet leading to the public network, in this example it is 198.8.8.13.

```
static {
        206.34.11.2 masklen 32 gateway 198.8.8.13 retain;
};
```

Optionally, the route to the foreign agent may originate from a routing protocol such as OSPF.

Connection from home network router to home agent

The home network router requires a routed circuit to the home agent. This route enables hosts and routers on the home network to reach the mobile node.

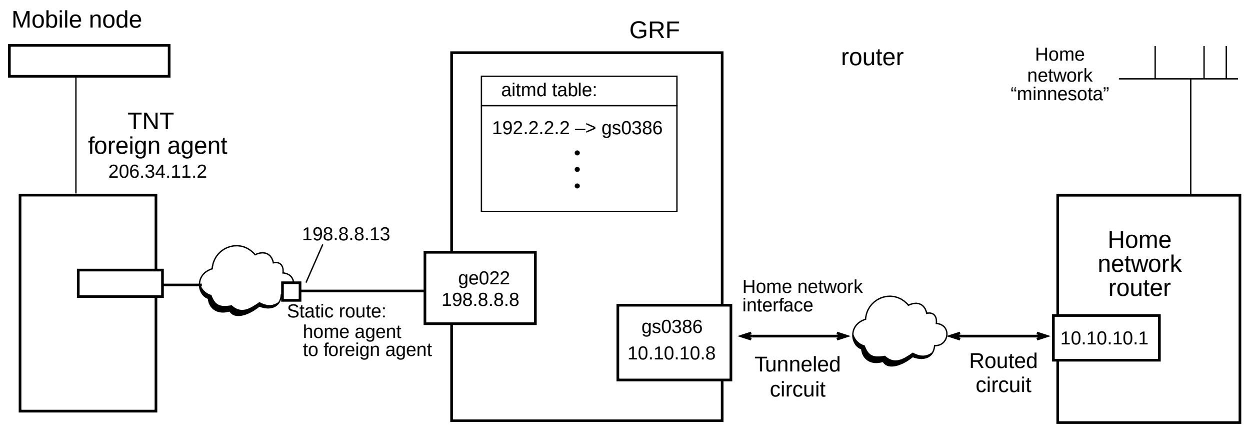


Figure 12-7. Routed circuit to the home network

The routed circuit defines the home agent as the route to the mobile node. The route's destination address specifies the Framed-Address of the mobile node and its gateway address specifies the IP address of the home agent's home network interface.

The routed circuit knows the private network address assigned to the home network interface. In the example, the HSSI interface's 10.10.10.8 private network address is statically configured on the home network router and is used only for a route in the direction of the home agent. This 10.10.10.8 address is a host address on the private home network and is assigned to the interface in the home network's interface section of the `/etc/aitmd.conf` file:

```
home_network  {
   name minnesota;                # text string name. no more than 31 characters
   home_agent_addr 192.2.2.2;     # IP address of home agent on the GRF

   interface {
      name gs0386;                # HSSI card in slot 3, port 1
      vpn_addr 10.10.10.8;        # VPN address
```

OSPF advertises home network addresses

When the GRF acts as a home agent in an Ascend Tunnel Management Protocol (ATMP) configuration, a home agent address is configured for each connecting home network. The home network addresses configured in the `/etc/aitmd.conf` file can be advertised via the OSPF protocol.

From the kernel point of view, all the home network addresses map to a single logical interface named `atmp0`. To advertise this group of addresses, configure an interface named "`atmp0`" into the OSPF section of the `gated.conf` file. The interface must be configured in such a way that it will be advertised via GateD, but GateD will not attempt to actually run the protocol over that interface. The home agent addresses will be advertised if the interface is specified in the OSPF statement as a stubhost.

To advertise an home agent address via OSPF, define it as a stubhost just as you do the loopback address. The home agent address will be exported as an OSPF route. The address does not have to be specified in the **export** section since OSPF routes are exported to OSPF by default.

Here is an example of `/etc/gated.conf` entries that assign home agent addresses to OSPF:

```
ospf yes {                                        ## START ospf
 #traceoptions "/var/tmp/gated.ospf" replace 1000k files 2 state all;
       •
       •
       •

backbone{
    stubhosts  {192.2.13.99 cost 10;};      ## routerid alias, GRF lo0
    stubhosts  {192.2.2.2  cost 10;};       ## home agt address “minnesota”
       •                                    ## home agt address “alameda”
       •                                    ## home agt address “westford”
       •                                    ## home agt address “minn.2”
    };
    ############# List of INTERFACES
    interface atmp0 cost 10 {passive; };      ## atmp home agents
    };
```

Source address notification option

The GRF home agent checks encapsulated packets received from a mobile node to verify that the packets' source address is assigned to the associated tunnel. The home agent checks the source IP address against the range of mobile node IP addresses registered for the associated tunnel. If the source IP address is out of range and does not match, the GRF home agent discards the packet.

Additionally, the home agent can be configured to notify the foreign agent of the error with the result that the foreign agent will tear down the tunnel. The configurable option in `/etc/aitmd.conf` is **bad_source_notification**, and is specified per home network.

Foreign agent notified

By default (**bad_source_notification yes**), the GRF home agent discards the packet.from a bad source address and sends an ATMP Error Notification to the foreign agent. This error indicates an Invalid Tunnel ID (code 5). The foreign agent responds by tearing down the tunnel and disconnecting the client. From a mobile node's point of view, this is a network problem because its connection keeps dropping out.

Foreign agent not notified

When **bad_source_notification no** is specified, the GRF home agent discards the packet from a bad source address but does not notify the foreign agent.

A TNT home agent handles these packets in the same way, and does not notify the foreign agent. This option supports customers using Windows 95 clients. In some cases, Windows 95 sends packets out the dial-up adapter interface with the IP source address set to use its Ethernet interface rather to use the mobile node IP address.

grstat support

The IP statistics reported by **grstat** include a related statistic, *ATMP err: wrong tunnel for mobile node.* This statistic reports the number of decapsulated packets in which the source address is not registered for the specified tunnel. The existing statistic, labeled *ATMP err: can't find mobile node entry*, continues to reflect the number of decapsulated packets with source addresses that are not registered for any tunnel. Here is an example:

```
# grstat ip
all cards (65 interfaces found)
  ipstat totals
       count description
   145973466 total packets received
   112609258 packets dropped
    33364193 packets forwarded normally
           5 packets handled by the card
          10 packets forwarded to the RMS
    23337312 packets ATMP encapsulated
  ipdrop totals
       count description
    14217922 IP option length past header length
    98391330 ATMP err: can't find mobile node entry
           6 ATMP err: wrong tunnel for mobile node
```

Using the /etc/aitmd.conf parameters

Parameters in the `/etc/aitmd.conf` file define the foreign agents and the home networks to which a GRF will connect. This section describes all available parameters.

Contents of /etc/aitmd.conf

Here is the file as updated for 1.4.10.

```
#  file:  /etc/aitmd.conf
#
#  This is a sample configuration file for the ATMP home agent server on
#  the GRF.  It is read by the daemon "aitmd".  See the man page for
#  aitmd and for aitmd.conf for additional information.  By default, the
#  aitmd daemon will expect to find its configuration file at
#  /etc/aitmd.conf
#
#  This file contains two kinds of records.  First, and simplest, are the
#  foreign agent records.  These give the IP addresses of ATMP foreign
#  agents that are permitted to initiate connections to the home agents.
#  Also, they include the password that will be used to authenticate the
#  foreign agents to us.  The foreign agent must be configured to use
#  this password.
#
#  Each home network is represented with a different home agent IP
#  address on the GRF.  For each home network you must configure a name,
#  which is what the foreign agent uses to identify the home network.
#
#  Each home network maps to a different local IP address on the GRF.
#  This is the home agent address.  This is the address to which the
#  foreign agents send to for tunnel negotiation and encapsulated
#  traffic.  Every home network must have a different IP address.
#
#  Presently, connections from the GRF back to the home network are only
#  supported on the HSSI and ATM-OC3 cards.  The home network record
#  contains an interface record which describes the logical interface
#  that is connected back the named home network.  The interface record
#  lists the logical interface name, and optionally, the IP address and
#  netmask size for this interface.  (Note that this address is on the
#  Virtual Private Network, not the public network, and must not be
#  specified in grifconfig.conf.)
#
#  The hssi/frame relay logical interfaces described in the aitmd
#  configuration file must each have a single ATMP virtual circuit
#  configured with the normal frame relay tools.  See the man pages for
#  grfr, grfr.conf, and fred for more information on frame relay circuit
#  configuration.
#
#  ATM logical interfaces described in the aitmd configuration file must
#  each have a single ATMP virtual circuit defined in gratm.conf.  See
#  the man pages for gratm and gratm.conf for more information.
#
#  Interfaces
#  ----------
#
#  Each logical interface used as an ATMP home network link must be defined
#  in grifconfig.conf.  These interfaces will be used in /etc/grfr.conf for
#  the pvcatmp statement, and in /etc/gratm.conf with the PVC where
#  proto=vc_atmp.  The interfaces in grifconfig.conf for ATMP home networks
#  should resemble the following.  Note that the first three options are
#  dashes.  Do NOT configure addresses for ATMP interfaces in
#  /etc/grifconfig.conf.
#
#  gs031   -  -  -  up     # sample interface for HSSI pvcatmp circuit
#  ga0288  -  -  -  up     # sample interface for ATM vc_atmp circuit#
```

```
#
#
#
#  Syntax
#  ------
#  Comments begin with the pound sign (#) and continue until the end of
#  the line.  Most fields are represented with a keyword followed by
#  its value.  The value  must be followed with a semi-colon.  Some fields
#  are grouped into records.  Records begin with a keyword followed
#  by a list enclosed withing curly braces {}.
#
#  Addresses may be given in dotted decimal notation, such 10.11.12.192,
#  or with host names.  It is recommended that IP addresses be used.
#  Names are more convenient to use and easier to remember, but if the
#  DNS name server is unreachable or down, then the aitmd server will not
#  be able to convert the name to an address and the configuration will
#  fail.  Using IP addresses directly makes the system immune to DNS
#  failures.
#
#  Numbers in this sample file are all in normal decimal notation.
#  Numeric values like the netmask size, card number, and port number may
#  also be entered in hexidecimal by using the '0x' prefix.  For example
#  0x4c would be the decimal value 76.  If you prefer octal, prefix the
#  number with a '0', like 0377 for decimal 255.
```

Foreign agent parameters

Each foreign agent that can connect to this GRF home agent is defined by two configuration parameters:

– **addr**, a routable IP address assigned to this foreign agent, the home agent uses this as the destination address in the GRE header.

– **password**, the password that validates this foreign agent to the GRF home agent, it is also configured in the mobile node's RADIUS profile as the "Ascend-Home-Agent-Password".

Here is the example of a foreign agent record from `/etc/aitmd.conf`:

```
foreign_agent {
   addr 172.20.5.100;    # IP address of the foreign agent
   password TrustNoOne;  # shared password
}
```

Home network parameters

A foreign agent connects to an individual home network. Therefore, each foreign agent record requires a home network record.

Here is the example of a GRF home agent record based on `/etc/aitmd.conf`:

```
home_network  {
   name Iowa;                         # text string name. no more than 31 characters
   home_agent_addr 10.200.7.6;            # IP address of home agent on the GRF
                                      # Only one home_network may use this address
   interface {
      name gs0a1;                     # HSSI card in slot 10, port 0
      vpn_addr 10.10.10.21;           # VPN address
      vpn_netmask_size 26;            #  bits in VPN netmask
      ripv2 {
         enabled yes;          #  Send RIPv2 multicasts on this interface
         metric 2;             #  metric for advertised routes
      }
   }

   mtu_limit 1200;             # pre-fragment packets before they enter the tunnel
   force_fragmentation yes;    #ignore IP DF bit when pre-fragmenting,
                               # if necessary
   bad_source_notification no;  #  drop packets from unknown source address
}
```

A home network is defined by this set of parameters:

– **name**, a unique name expressed as a text string of up to 31 characters that is used by the foreign agent to identify the home network.

– **home_agent_addr**, each home network maps to a different local IP address on the GRF, this is the home agent address.

The foreign agent uses this address as the destination address in its IP encapsulation header. The GRF home agent uses it as the source address in its IP encapsulation header.

The following parameters define the logical GRF interface on which the home agent is configured and which also connects to the home network:

- **interface name** ***gx0yz***, the interface name in standard GRF `gx0yz` format that indicates media, slot, and logical interface number.
- **vpn_addr** ***xx.xx.xx.xx***, the customer-assigned address for the home agent on the home network (must be in the private network address space).
- **vpn_netmask_size** ***number***, number of bits in the netmask for **vpn_addr**.

Note: Configuration parameters for the home agent have changed since 1.4.6:

1. The older “**circuit**” section is replaced by the “**interface name**” entry.

However, backward compatibility is maintained and **circuit** entries will be parsed. BUT — if you leave **circuit** entries in the configuration file, DO NOT include the new **interface name** entry, the home network entry will be rejected.

2. The **netmask_size** parameter is no longer used and is now ignored.

RIPv2 parameters

These parameters are required for RIPv2 transmission.

- **ripv2 enabled yes** or **no**

Sets **aitmd** to send RIPv2 packets across the circuits to connected home networks. The packets carry the routes of all mobile nodes assigned to this home agent. The routes connect the home agent **vpn_addr** with the mobile node’s private address.

- **ripv2 metric** ***number***

Specifies the metric advertised with each route. **metric** is ignored if **enabled no** is specified.

In addition, the **vpn_addr** and **vpn_netmask_size** parameters are required for RIPv2 transmission.

Fragmentation parameters

These two parameters are optional unless you wish to enable pre-fragmentation:

- **mtu_limit** ***number***

Controls whether and how pre-fragmentation will occur.

If the value is 0 (the default), no pre-fragmentation is performed (post-fragmentation is performed). If nonzero, **mtu_limit** specifies the maximum size of the fragments, prior to encapsulation, before pre-fragmentation will occur.

If the value is **auto**, the **mtu_limit** is automatically computed for each packet so as to avoid post-fragmentation.

- **force_fragmentation**, **yes** or **no**

Controls the behavior of the GRF home agent when it receives a packet from the home network which requires pre-fragmentation, but has the IP DF (Don't Fragment) option set. It is ignored when **mtu_limit** is zero.

Definition of fragmentation parameters

Use the **mtu_limit** and **force_fragmentation** parameters to control the pre-fragmentation process on the GRF home agent. The parameters are configured for a specific home network, and apply to all traffic from that home network, on any circuit, as the traffic enters a tunnel. The GRF parameter names are the same as implemented on the MAX and TNT home agent implementation.

mtu_limit parameter

The **mtu_limit** parameter controls whether and how pre-fragmentation will occur. If the value is 0 (the default), no pre-fragmentation is performed.

If nonzero, **mtu_limit** specifies the maximum size of the fragments, prior to encapsulation, before pre-fragmentation will occur.

Here are some example values:

An **mtu_limit** value of 1472 allows datagrams to pass through an Ethernet interface without further fragmentation. The value 1472 is the Ethernet MTU (1500), less the size of the GRE and second IP headers (28 bytes).

In addition, if **mtu_limit** has the keyword value of **auto**, then pre-fragmentation occurs at the maximum size allowed by the MTU of the interface used to transmit to the foreign agent. For example:

– If the interface used to transmit to the foreign agent is an Ethernet with an MTU of 1500, then the implied **mtu_limit** value is 1472

– If the interface is HSSI with an MTU of 4352, then the implied **mtu_limit** value is 4324.

If the **mtu_limit** has a nonzero value which would cause post-fragmentation, that is, it is larger than a given outbound interface's MTU, less 28 bytes, then the GRF home agent acts as if an **mtu_limit** of **auto** were configured —but only for traffic using that interface.

force_fragmentation parameter

The **force_fragmentation** parameter has the Boolean value yes or no, and is interpreted only when the **mtu_limit** is nonzero; otherwise it has no effect.

The **force_fragmentation** parameter controls the behavior of the GRF home agent when it receives a packet from the home network which requires pre-fragmentation, but has the IP DF (Don't Fragment) option set.

If **force_fragmentation** has the value **no** (the default), then the GRF home agent rejects such packets, and issues an ICMP HOST UNREACHABLE, MUST FRAGMENT message in response. This is the normal, required behavior for a router.

If **force_fragmentation** has the value **yes**, then the GRF home agent will behave in a non-standard way when it receives such a packet:

– The GRF home agent will accept the packet, clear the DF option in the IP header, and process it as if the DF were not present.

The GRF home agent will not modify the DF option on packets which do not require pre-fragmentation.

Note: The **force_fragmentation** parameter is required to support certain IP stacks (Windows 95 in particular) which set the DF option in each transmitted packet, but do not properly process ICMP HOST UNREACHABLE, MUST FRAGMENT messages.

An unfortunate side-effect is that it effectively disables MTU Path Discovery for well-behaved hosts on the home network.

Source address notification parameter:

One parameter controls whether the GRF home agent notifies the foreign agent when packets from the mobile node have bad source addresses:

– **bad_source_notification yes** (default), packet received with non-matching source address is dropped and foreign agent is notified, tunnel is torn down.

– **bad_source_notification no**, packet received with non-matching source address is dropped, no notification is made.

Starting and checking aitmd

By default, **aitmd** is not running. To enable ATMP, the administrator creates the file `/etc/aitmd.run`. This is done in the UNIX shell with the command:

```
# touch /etc/aitmd.run
```

Save the configuration change with:

```
# grwrite
```

The daemon starts within 15 seconds and restarts automatically on future reboots. You can “remove” the file normally with **rm** and save the removal with a **grwrite**. Removing `/etc/aitmd.run` does not kill the daemon, it only prevents **aitmd** from being restarted. If the run file is in `/etc`, then **aitmd** is running.

The daemon loads the ATMP configuration from the file `/etc/aitmd.conf`. Configurations can be changed on-the-fly by editing `aitmd.conf` and then sending the process a HUP signal.

Is aitmd running ?

To check that **aitmd** is running, use the **ps** command:

```
# ps ax | grep aitmd
10199  p2  S+     0:00.02 grep aitmd
  390  00- I      0:00.04 /usr/sbin/aitmd -F
```

If **aitmd** is running, its PID is reported in the `/usr/sbin/aitmd` line of output. If there is no such line, **aitmd** is not running.

What is configured ?

You can also check what is configured using the **kill -INFO** command.

This example returns the **aitmd** configuration information for home agent “`agent5`”. You can see statements for 1 configured Foreign Agent and 1 configured Home Agent.

```
# ps ax | grep aitmd
  474  00- I      0:05.41 /usr/sbin/aitmd -F
# kill -INFO 474

# Aug  5 20:21:38 agent5 aitmd: INFO: Current status Wed Aug  5
   20:21:38 1998
Aug  5 20:21:38 agent5 aitmd: 1 configured Foreign Agents
Aug  5 20:21:38 agent5 aitmd: FA: 206.146.160.181, <NULL>
Aug  5 20:21:38 agent5 aitmd: circuit: state=Up, slot=2, port=0, s0=15,
  s1=511 ifname="ga027f" ifindex=40
Aug  5 20:21:38 agent5 aitmd:   vpn_addr=10.20.2.237/30
Aug  5 20:21:38 agent5 aitmd: HN: 0, "vpnrip", 221.1.1.4, FA peer cnt=1
Aug  5 20:21:38 agent5 aitmd:    tunnels:
Aug  5 20:21:38 agent5 aitmd: TunN: tid=633, addr=10.20.2.121,
  mask_size=29, fa=206.146.160.181, hn=vpnrip
Aug  5 20:21:38 agent5 aitmd: TunN: tid=753, addr=10.20.2.241,
  mask_size=29, fa=206.146.160.181, hn=vpnrip
Aug  5 20:21:38 agent5 aitmd: ripv2: enabled=yes, metric=1
Aug  5 20:21:38 agent5 aitmd: 1 configured Home Agent
```

Overview of home agent configuration

For each mobile node, you have five areas to configure on the GRF home agent:

1 Use ATMP parameters in the `/etc/aitmd.conf` file to describe the foreign agent and home network.

2 Set up a tunneled circuit from the home agent to the mobile node's home network, this can be done through either a HSSI Frame Relay link/PVC or an ATM OC-3c PVC.

3 On the home network router, complete the other end of the connection. Create a routed circuit from the home network router back to the home agent.

4 Give the home network a path to the mobile node via static or dynamic routing:

a. Set up static route(s) on the home network router to the mobile node(s) via the home agent.

OR:

b. Enable RIPv2 transmission on the GRF home agent and the connecting home network routers/servers. RIPv2 multicast packets will advertise routes for registered mobile nodes to the home network.

5 Exchange / provide configuration information to / for the TNT foreign agent.

The next sections briefly describe the configuration file entries and tasks for the five areas listed above. File entries are based on the ATMP example shown in Figure 12-8.

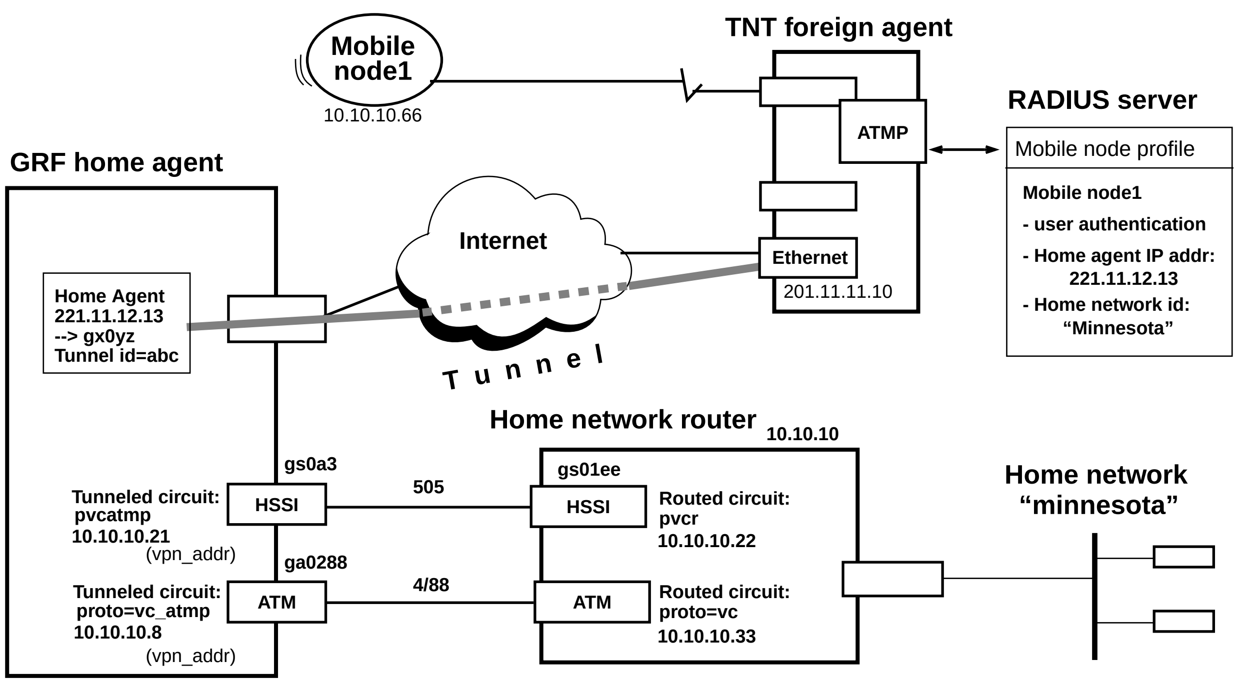


Figure 12-8. Sample ATMP components described in configuration overview

Task 1. Configure GRF ATMP parameters in */etc/aitmd.conf*

The `/etc/aitmd.conf` file tells the ATMP daemon, **aitmd**, about the home networks and foreign agents to which the GRF can connect.

From the CLI, establish a UNIX shell:

```
super> sh
#
```

In the shell, use a UNIX editor to edit `/etc/aitmd.conf` configuration files.

A copy of the `/etc/aitmd.conf` file is in the *GRF Reference Guide*, you can also refer to its man page (**man aitmd**.**conf**).

The `/etc/aitmd.conf` file has two types of record entries, one for foreign agents, one for home networks. In this file you identify which foreign agents the GRF will recognize and to which home networks the GRF connects.

Note: In the `aitmd.conf` file, keywords are entered in lower case, variables can use both upper and lower case. You must use curly braces and semi-colons as shown.

1 Identify the foreign agent by IP address or host name, and supply the password the GRF and the foreign agent will use as part of tunnel negotiation. Host names can be used to identify the foreign agent but because they are dependent upon DNS, they are less reliable than IP addresses.

- Specify the IP address with the `addr` keyword.

- Specify `password` using no more than 20 alphanumeric characters. The password is a "shared secret" that must also be entered in the mobile node's RADIUS profile used by the TNT foreign agent.

Here are the entries for the foreign agent: assigned to mobile node 2:

```
# /etc/aitmd.conf
#
foreign_agent {
   addr 201.11.11.10;          # IP address of the foreign agent
   password dont_tell;         # password used also by mobile node2
}
```

2 The home network is identified in greater detail, including information about the home agent.

Note: Configuration parameters for the home agent have changed since 1.4.6:

1. The older "**circuit**" section is replaced by the "**interface name**" entry.

However, backward compatibility is maintained and **circuit** entries will be parsed. BUT — if you leave **circuit** entries in the configuration file, DO NOT include the new **interface name** entry, the home network entry will be rejected.

2. The **netmask_size** parameter is no longer used and is now ignored.

Here are the entries for the home network assigned to mobile node 2:

```
home_network  {
   name “minnesota”;                # home network based in minnesota
   home_agent_addr 221.11.12.13;    # home agent address
     interface {        # home agent interface to home network
        name gs0a3;            # HSSI card in slot 10, port 0
        vpn_addr 10.10.10.21;       # GRF's address on the VPN
        vpn_netmask_size 26;        # size of VPN netmask
        ripv2 {
            enabled no;              # no RIP multicast of routes
            metric 2;               # ignored in this configuration
      }
   }
   mtu_limit 1472; #pre-frag path to foreign agent is Ethernet
   force_fragmentation yes;         # ignore IP DF bit
}
```

Fragmentation parameters

These two parameters are optional unless you wish to enable pre-fragmentation:

– **mtu_limit** ***number***

Controls whether and how pre-fragmentation will occur.
If the value is 0 (the default), no pre-fragmentation is performed (post-fragmentation is performed). If nonzero, **mtu_limit** specifies the maximum size of the fragments, prior to encapsulation, before pre-fragmentation will occur.
If the value is **auto**, the **mtu_limit** is automatically computed for each packet so as to avoid post-fragmentation.

– **force_fragmentation**, **yes** or **no**

Controls the behavior of the GRF home agent when it receives a packet from the home network which requires pre-fragmentation, but has the IP DF (Don't Fragment) option set. It is ignored when **mtu_limit** is zero.

More information about fragmentation options is provided in the section “Using the /etc/aitmd.conf parameters.”

Task 2. Connect home agent to the home network

This section describes configuration steps to connect a HSSI or an ATM OC-3c interface to a home network. Use either type of connection to the home network.

2a. HSSI Frame Relay connection to home network

Overview

Given that you want to establish a connection to the home network using a Frame Relay link on the HSSI card interface `gs0a3`, here are the steps:

1. Configure the Frame Relay link in the *Link* section of `/etc/grfr.conf`.
2. Configure the tunneled circuit in the *PVCATMP* section of `/etc/grfr.conf`.
3. For ATMP, specify the circuit as a blank interface in `/etc/grifconfig.conf`.

From the CLI, establish a UNIX shell to edit the configuration files:

```
super> sh
#
```

A copy of the `/etc/grfr.conf` file is in the *GRF Reference Guide*, you can also refer to its man page (**man grfr.conf**).

Configuration

There are two configuration tasks in the `/etc/grfr.conf` file. Repeat these tasks for each home network connected to the GRF.

1 In the *Link* section of the `/etc/grfr.conf` file, create a link on the physical HSSI port. Specify the `gs0a3` chassis slot, HSSI physical port 0 or 1, and any optional link management parameters:

```
#    Slot  Port  Optional Parameters
#    ====  ====  ==================
link 10     0     Name="link_to_minn" Linktype=UNI-DCE
```

Optional link parameters

These definitions of optional parameters are from the `/etc/grfr.conf` file.

- Name= link name, up to 31 characters, an alphanumeric string, default = "".
- Enabled= Y | N, enable/disable link, default = Y.
- LMIType= None | AnnexA | AnnexD, default is None.
- N391= 1..255: polling intervals per full status message, default = 10.
- N392= 1..10: Error Reporting Threshold. Default = 3.
- N393= 1..10: Measurement Interval for mN2. Default = 4.
- T391= 5 | 10 | 20 | 25 | 30: Heartbeat Poll Interval. Default = 5.

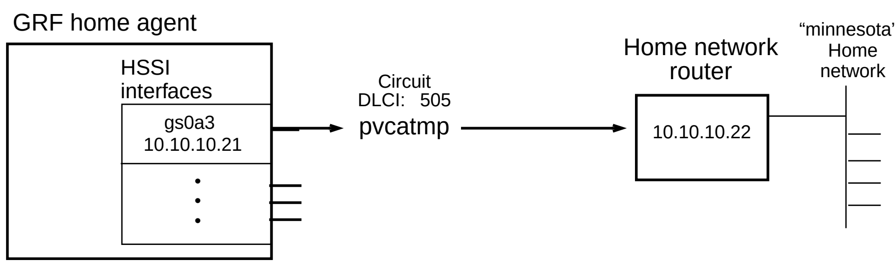
- T392= 5 | 10 | 15 | 20 | 25 | 30: Poll Verification Timer. Default = 15.
- Linktype= UNI-DTE | UNI-DCE | NNI. Default is UNI-DTE.
- AutoAddGrif= This parameter is ignored for ATMP.

2 In the *PVCATMP* section of the `/etc/grfr.conf` file, establish the PVCATMP to the home network router.

This connection is a special type of PVC, a PVCATMP. Specify the logical interface name for the GRF interface, the DLCI name for the link to the home network, and the IP address of the station at the other (not the home agent) end of the link.

Here is a sample entry in the *PVCATMP* section of `/etc/grfr.conf` based on the figure below:

```
#        lif   DLCI Peer IP Address  Optional Parameters
#        ===   ==== ===============  ===================
pvcatmp gs0a3  505  10.10.10.22      Name="link_to_minn"
```



- The `lif` is the logical interface name of the GRF home agent interface on which the PVCATMP is being configured.
- The DLCI is the value assigned to the PVCATMP circuit going to the home network.
- The peer IP address is that of the station at the other end of the PVCATMP circuit. As shown, this station can be an intermediate connecting router. If there is no connecting router, this station is the home network router.
- Optional parameters can also be specified, they are described in the next section.

Save the `/etc/grfr.conf` file and use **grfr** commands to activate the link and the PVCATMP you have just configured:

```
# grfr -c ccl -s 10 -l 0                        # activates link
LINK Defined: Slot 10, link 0
#
# grfr -c ccp -s 10 -l 0 -i 505                 # activates PVCATMP
PVC Defined: 10, link 0, dlci 505
#
```

Optional PVCATMP parameters

These are the optional parameters you can assign a PVCATMP:

- Name: Quoted string: PVC name, default = "" (up to 31 characters)
- Enabled= Y | N, enable/disable PVC, default = Y.
- CIR=integer Committed Information Rate, default = 55000000 bits/second
- Bc=integer Committed Burst Size, default = 55000000 bits/second
- Be=integer Excess Burst Size, default = 0 bits/second

CIR, Be, and Bc are traffic shaping parameters. Their defaults have proven to be problematical for generic Frame Relay applications. The HSSI media card, for example, can more efficiently handle a different set of values. The recommended bit values for HSSI cards are as follows:

- CIR value = 22000000
- Bc value = 22000000
- Be value = 0

3 ATMP requires that the PVCATMP interface be configured as a blank or inactive in `/etc/grifconfig.conf`.

Based on the example above, here is the required entry:

```
# /etc/grifconfig.conf
# name  address     netmask     broad_dest   arguments
gs0a3 - - - up         # atmp config pvcatmp
```

Large packets through tunnel

You may see a problem with large packets not getting through ATMP tunnels that go over Ethernet connections. This can be caused by the HSSI card enforcing the traffic limits specified in `/etc/grfr.conf`, or by the terms of your network subscription service. If possible, adjust the default values on the ATMP gateway circuit in the *PVCATMP* section of `/etc/grfr.conf`.

In this *PVCATMP* example, CIR, Bc and Be are assigned the recommended values that will remove HSSI restrictions through the tunnel:

```
#       lif  DLCI  Peer IP Address  Optional Parameters
#       ===  ====  ==============  ===================
pvcatmp gs0a3 505 10.10.10.21  Name="iowa" Cir=55000000 Bc=55000000 Be=0
```

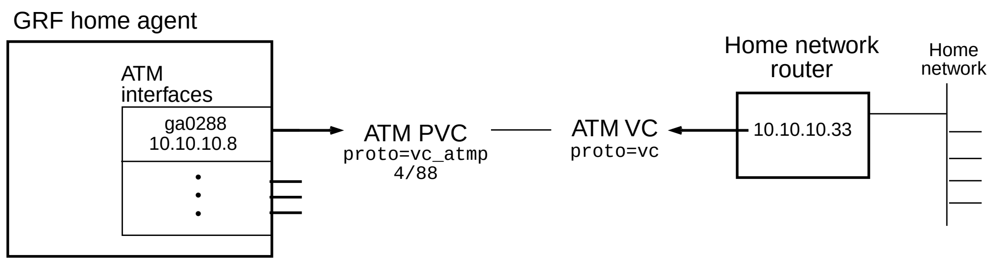
2b. ATM OC-3c circuit to home network

Overview

In this option, the GRF ATM connection to a home network is made across a PVC from an ATM OC-3c card. The home network router connects to the GRF ATM PVC through an ATM VC.

There are four steps to configure an ATM circuit to the home agent. Repeat these tasks for each home network connected to GRF ATM interfaces:

1. Specify the traffic shape name in the *Traffic Shaping* section of `/etc/gratm.conf`.
2. Configure the ATMP interface in the *Interfaces* section of `/etc/gratm.conf`.
3. Configure the tunneled circuit in the *PVC* section of `/etc/gratm.conf`.
4. Configure the tunneled circuit as a blank interface in `/etc/grifconfig.conf`.
5. Because this is a `vc_atmp` PVC, you need to make an entry in `/etc/grarp.conf` to specify an ARP entry for the home agent's peer. In this case, the home network router.



Configuration

From the CLI, establish a UNIX shell to edit the configuration files:

```
super> sh
#
```

1 In the `/etc/gratm.conf` file, set traffic shaping name and quality of service parameters in the *Traffic Shaping* section.

Using any string, set a name for each type of service that will be assigned. Text in the `/etc/gratm.conf` file describes how to specify a range of traffic shaping parameters.

This example uses `h_s_h_q` as a name to represent high_speed_high_quality.

```
# Traffic shaping parameters
# Lines beginning with the keyword "Traffic_Shape" define
# traffic shapes which may be used to configure the performance
# characteristics of ATM Virtual Circuits.
#
Traffic_Shape name=h_s_h_q \
       peak=155000 sustain=155000 burst=2048 qos=high
```

2 In the *Interfaces* section of the `/etc/gratm.conf` file, specify the logical interface name of the circuit connecting to the home network and assign it one of the `traffic_shape` names you defined in step 1:

```
# Interfaces
Interface ga0288 traffic_shape=h_s_h_q
```

3 In the *PVC* section of the `/etc/gratm.conf` file, specify the logical interface name of the circuit connecting to the home network.

You can configure the circuit to use LLC encapsulation or be VC-based multiplexed when you specify the ATMP protocol, llc_atmp or vc_atmp.

You must also assign it a VPI/VCI, specify the ATMP protocol, and assign the same `traffic_shape` name you gave the logical interface:

```
# PVCs
PVC ga0288 4/88 proto=vc_atmp traffic_shape=h_s_h_q
PVC ga052e 9/122 proto=llc_atmp traffic_shape=h_s_h_q
```

4 ATMP requires that the PVC interface be defined in `/etc/grifconfig.conf`.

Here is the required entry:

```
# /etc/grifconfig.conf
# name   address   netmask   broad_dest   arguments
ga0288  - - - up         # atmp requirement vc_atmp
ga052e  - - - up         # atmp requirement llc_atmp
```

5 Make an entry in `/etc/grarp.conf` that specifies an ARP entry for the home network interface's peer, in this case, it is the home network router.

```
# /etc/grarp.conf
#ifname  hostname  phys_addr    [temp]   [pub]   [trail]
ga0288   10.10.10.33  4/88
```

The `hostname` is the peer address, the address of the home network router. The `phys_addr` is the VPI/VCI of the PVCATMP.

Note: LLC supports Inverse ARP so an `/etc/grarp.conf` entry is not needed for `llc_atmp` PVCs.

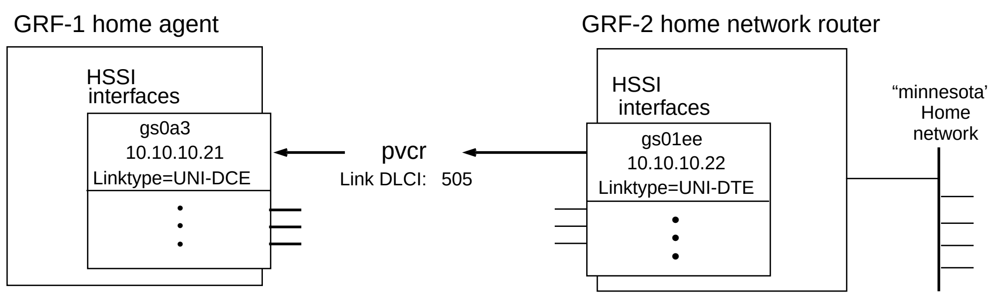
Task 3. Connect home network router to home agent

To forward packets from the home network destined for the mobile node, this router must have a routed PVC to the home agent.

If the home network router is a GRF, the interface to the home agent is either a HSSI or ATM OC-3c media card. A home agent connection from a HSSI card is a Frame Relay PVCR. A home agent connection from an ATM card is a null-encapsulated IP virtual circuit, VC (proto=vc).

Overview

Assume the connection from a GRF home network router is made across a HSSI Frame Relay circuit as shown in this diagram:



Here are the tasks:

1. Configure a Frame Relay link in the *Link* section of `/etc/grfr.conf`.
2. Configure a routed circuit in the *PVCR* section of `/etc/grfr.conf`.
3. Configure the HSSI card interface in `/etc/grifconfig.conf`.

Configuration

1 In the GRF-2 `/etc/grfr.conf` file, create a link for the physical HSSI port in the *Link* section.

Specify the GRF chassis slot, HSSI physical port, and any optional link management parameters:

```
#    Slot  Port  Optional Parameters
#    ====  ====  ===================
link 1     1     Name="to_agent1" Linktype=UNI-DTE
```

Optional link parameters

These definitions of optional parameters are from the `/etc/grfr.conf` file.

– Name= link name, up to 31 characters, an alphanumeric string, default = "".

– Enabled= Y | N, enable/disable link, default = Y.

– LMIType= None | AnnexA | AnnexD, default is None.

– N391= 1..255: polling intervals per full status message, default = 10.

- N392= 1..10: Error Reporting Threshold. Default = 3.
- N393= 1..10: Measurement Interval for mN2. Default = 4.
- T391= 5 | 10 | 20 | 25 | 30: Heartbeat Poll Interval. Default = 5.
- T392= 5 | 10 | 15 | 20 | 25 | 30: Poll Verification Timer. Default = 15.
- Linktype= UNI-DTE | UNI-DCE | NNI. Default is UNI-DTE.
- AutoAddGrif= This parameter is ignored for ATMP.

2 In the *PVCR* section of the `/etc/grfr.conf` file, establish the PVCR to the home agent. This connection is a routed circuit.

Here is the entry in the *PVCR* section of the router's `/etc/grfr.conf` file:

```
#     lif    DLCI Peer IP Address  Optional Parameters
#     ===    ==== ===============  ===================
pvcr gs01ee  505  10.10.10.21     Name="to_agent1"
```

- The `lif` is the logical interface name of the GRF home network router interface on which the PVCR is being configured.
- The DLCI is the value assigned to the PVCR circuit going to the home agent.
- The peer IP address should be on the same subnet as the PVCATMP.
- Optional parameters can also be specified, they are described in the next section.

Save the `/etc/grfr.conf` file and use **grfr** commands to activate the link and the PVCR you have just configured:

```
# grfr -c ccl -s 1 -l 1                        # activates link
LINK Defined: Slot 1, link 1
#
# grfr -c ccp -s 1 -l 1 -i 505                 # activates PVCR
PVC Defined: 1, link 1, dlci 505
#
```

Optional PVCR parameters

These are the optional parameters you can assign a PVCR:

- Name: Quoted string: PVC name, default = "" (up to 31 characters)
- Enabled= Y | N, enable/disable PVC, default = Y.
- CIR=integer Committed Information Rate, default = 55000000 bits/second
- Bc=integer Committed Burst Size, default = 55000000 bits/second
- Be=integer Excess Burst Size, default = 0 bits/second

CIR, Be, and Bc are traffic shaping parameters. Their defaults have proven to be problematical for generic Frame Relay applications. The HSSI media card, for example, can more efficiently handle a different set of values. The recommended bit values for HSSI cards are as follows:

- CIR value = 22000000
- Bc value = 22000000
- Be value = 0

3 Configure the logical interface as usual in `/etc/grifconfig.conf`:

```
# /etc/grifconfig.conf
# name  address     netmask     broad_dest   arguments
gs01ee  10.10.10.22  255.255.255.0
```

Task 4. Specify path to mobile node for home network

The home network needs a path to the mobile node. Usually you would set up a static route on the home network router. Or, if you are running RIPv2 transmission, you enable RIPv2 on the home agent and the home network router.

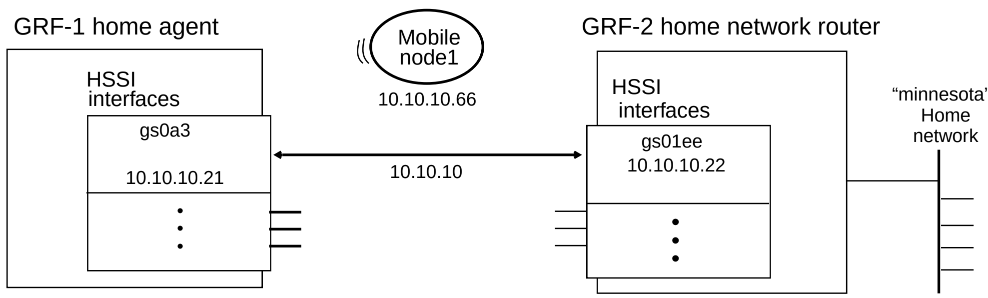
You can either:

a. Set up static route(s) on the home network router to the mobile node(s) via the home agent.

OR:

b. Enable RIPv2 transmission on the GRF home agent and the connecting home network routers/servers. RIPv2 multicast packets will advertise routes for registered mobile nodes to the home network.

The following examples for static and RIPv2 route configuration assume the home network router is a GRF.



Static route

Configure a static route in the GRF home network router's `/etc/gated.conf` file that connects the mobile node VPN address with the associated interface (gateway) on the GRF home agent.

```
static{
   10.10.10.66 masklen 32 gateway 10.10.10.21 retain;
];
```

This works because part of the configuration for the routed circuit back to the home agent includes a PVCR statement in the home network router's `/etc/grfr.conf` file:

```
#     lif    DLCI Peer IP Address  Optional Parameters
#     ===    ==== ===============  ===================
pvcr gs01ee  505  10.10.10.21     Name=”to_agent1”
```

This statement connects the gateway address to a DLCI.

RIPv2 route

On the home agent

Enable RIPv2 routing in the GRF home agent `/etc/aitmd.conf` file under the `interface` subsection:

```
interface {             # home agent interface to home network
   name gs0a3;                 # HSSI card in slot 10, port 0
   vpn_addr 10.10.10.21;    # GRF's address on the VPN
   vpn_netmask_size 26;     # size of VPN netmask
   ripv2 {
      enabled yes;  # Send RIPv2 multicasts on this interface
      metric 1;     # metric for advertised routes
```

On the home network router

Enable RIPv2 routing in the `/etc/gated.conf` file:

```
rip yes {
   interface gs01ee ripin version 2;
};
```

Task 5. Configuration links to the TNT foreign agent

The initial (pre-tunnel) communication from a TNT foreign agent to a GRF home agent is via a normally-routed IP connection across Ethernet, HSSI, or ATM OC-3c interfaces. The TNT does not have ATM, but the GRF can communicate to the TNT across an ATM-based WAN.

The foreign agent sends the tunnel request to the GRF home agent. Routed messages are exchanged to negotiate the tunnel. No special ATMP-related configuration is needed for negotiation.

Because a unique IP address, the home agent address, is assigned to the home agent for each attached home network, the foreign agent “sees” one home agent per home network, no matter how many home agents actually exist.

Mobile node RADIUS profile

As part of TNT ATMP configuration, a RADIUS user profile is created for each mobile node.

The RADIUS server has two profile databases for configuring ATMP: clients and users. The clients profile defines which hosts may access the RADIUS database. The users profile defines configuration attributes for a particular user. The RADIUS server stores both profiles in the `/etc/raddb` directory. The mobile router profile is in the `/users` database.

The user profile uses the mobile node name as a key and defines a series of attribute=value pairs for that node. The user profile password is the one which the foreign agent presents to the home agent when it requests a tunnel on behalf of the mobile node. The password must match the foreign agent password specified in `/etc/aitmd.conf` on the GRF home agent. The home agent is authoritative for this exchange. The IP address in the foreign agent record must match the address of the interface the foreign agent will use to contact the home agent.

An example of a user profile for mobile node 1 is shown here. This is the RADIUS user profile for mobile nodes running TCP/IP:

```
node1 Password="dont-tell"
   Ascend-Metric=2,
   Framed-Protocol=PPP,
   Framed-Address=10.10.10.66,
   Framed-Netmask=255.255.255.0,
   Ascend-Primary-Home-Agent=221.11.12.13,
   Ascend-Secondary-Home-Agent=221.11.12.43,
   Ascend-Home-Network-Name=minnesota,
   Ascend-Home-Agent-Password="dont-tell",
   Ascend-Home-Agent-UDP-Port=5150,
   Ascend-Idle-Limit=20
```

Five entries in the RADIUS user profile relate to GRF home agent configuration:

– Ascend-Primary-Home-Agent=

– Ascend-Secondary-Home-Agent=

– Ascend-Home-Agent-Password=

– Ascend-Home-Agent-UDP-Port = 5150

– Ascend-Home-Network-Name =

Ascend-Primary-Home-Agent

The `Ascend-Primary-Home-Agent=` entry must be the unique address for that home agent.

TNT point-of-view: This is the first home agent the foreign agent tries to reach when setting up an ATMP tunnel for this mobile node, it is the home agent address pointing to the target home network.

```
Ascend-Primary-Home-Agent=221.11.12.13,
```

This is the corresponding IP address entry in the GRF's `/etc/aitmd.conf` configuration file for the home network named `"minnesota"`:

```
home_network  {
   name "minnesota";         # home network based in minnesota
   home_agent_addr 221.11.12.13;   # home agent IP addr
```

Specify the home agent address in dotted decimal notation. IP addresses are recommended rather than domain names because the domain name server can fail.

GRF point-of-view: This IP address is the home agent address on the GRF home agent. This is the address to which the foreign agent sends encapsulated traffic via the tunnel through the Internet. Each home network needs to see the GRF as a different IP entity, hence a different home agent address for each home network. These addresses allow the GRF to connect to multiple home networks.

The home agent address is used by the operating system and must not be entered by a user in the `/etc/grifconfig.conf` file.

To check which IP addresses are assigned to `atmp0` after the **aitmd** daemon loads a home network configuration, use the **netstat -i** command:

```
# netstat -i
atmp0   1536  <link4>                     0     0     0    0     0
atmp0   1536  172.30.1.9   2,0,100,0     0     0     0    0     0
atmp0   1536  0/32        172.30.1.9     0     0     0    0     0
atmp0   1536  221.1.1.2    2,1,101,0     0     0     0    0     0
atmp0   1536  0/32         221.1.1.2     0     0     0    0     0
```

In this example, 172.30.1.9 and 221.1.1.2 are addresses for two home agents. The GRF supports multiple home agents and connects to multiple home networks, you see a corresponding number of `atmp0` addresses reported by **netstat -i**.

Ascend-Secondary-Home-Agent

TNT point-of-view: This is the address of the alternate home agent the foreign agent tries to reach when setting up an ATMP tunnel for a mobile node. If the foreign agent is unable to negotiate an ATMP tunnel to the primary home agent, then it will attempt to negotiate a tunnel with the secondary home agent.

The address in this parameter becomes the new address pointing to the target home network.

```
Ascend-Secondary-Home-Agent=221.11.12.43,
```

Ascend-Home-Agent-Password

This is the password that the TNT foreign agent sends to the GRF home agent during an ATMP negotiation. It must match the GRF ATMP password entered into the foreign agent record in the `/etc/aitmd.conf` configuration file. Use a text string of up to 20 characters, the default value is null.

```
node1 Password="dont-tell"
```

This is the corresponding entry in the GRF's `/etc/aitmd.conf` configuration file:

```
foreign_agent {
   addr yyy.yyy.yyy.yyy;    # IP address of the foreign agent
   password dont-tell;      # shared secret password
}
```

Ascend-Home-Agent-UDP-Port = 5150

The GRF home agent uses the same UDP port as the TNT, 5150. Port number 5150 is "hardwired" in the GRF operating software. Please leave the TNT RADIUS profile setting at the default of 5150.

Ascend-Home-Agent-Name

This is the entry for the name assigned the home network in the RADIUS profile. Use a text string of up to 31 characters.

```
Ascend-Home-Network-Name=minnesota
```

This is the corresponding entry in the GRF home agent's `/etc/aitmd.conf` configuration file:

```
home_network  {
   name minnesota;   #text string name. no more than 31 characters
```

Monitoring ATMP activity on the GRF

This section describes ways to verify that ATMP configuration and addresses are correct.

Check aitmd configuration first

To verify or troubleshoot an ATMP configuration, start with **aitmd**. Use the **kill -INFO** command to see what **aitmd** "thinks" is configured:

```
# ps ax | grep aitmd
  474  00- I      0:05.41 /usr/sbin/aitmd -F
# kill -INFO 474
```

Remember, when there is no response to **kill -INFO**, **aitmd** is hung and must be restarted.

netstat -i command

The next place to look when troubleshooting is the system. The **netstat -i** command displays what the kernel "knows" about the current home agent configurations.

The **netstat -i** command returns a display from which you can verify several home agent and home network configuration parameters.

Home agent address

Interface to home network

The slot, port, DLCI or VPI/VCI numbers for the circuit to the home network

link

```
# netstat -i
Name    Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
de0     1500  <link1>     00:c0:80:86:16:e2 627493     0   381326     0  7777
de0     1500  206.146.160 agent5            627493     0   381326     0  7777
rmb0    620   <link2>     00:00:00:00:00:00 587033 10649   564760     0     0
rmb0    620   <GRIT>      0:0x40:0          587033 10649   564760     0     0
lo0     1536  <link3>                         6139     0     6139     0     0
lo0     1536  <GRIT>      0:0x48:0            6139     0     6139     0     0
atmp0   1536  <link4>                           10     0       10     0     0
atmp0   1536  205.1.1.1          1,0,100,0      10     0       10     0     0
atmp0   1536  0/32        205.1.1.1             10     0       10     0     0
atmp0   1536  221.1.1.3          2,1,15,511     10     0       10     0     0
atmp0   1536  0/32        221.1.1.3             10     0       10     0     0
gl000   1496  <link5>                            0     0        0     0     0
gs000*   4352  <link33>                       57902     0    57901     0     0
gs017f  4352  <link15>                           0     0        0     0     0
gs017f  4352  <vpn3707830531> 10.20.2.221        0     0        0     0     0
ga02f1  9180  <link16>    00:c0:80:fa:cd:f1      0     0        0     0     0
ga02f1  9180  210.1.1     210.1.1.150            0     0        0     0     0
         .
         .
         .
```

Figure 12-9. ATMP entries as reported in netstat -i

The introductory entry in the list of ATMP (`atmp0`) entries always has an associated Link number. This is the SNMP interface index that is assigned when an interface is created:

```
atmp0   1536  <link4>                10     0      10     0     0
```

An ATMP entry is a pair of lines. The nest four `atmp0` lines shown in the excerpt report on two home networks. The entry in the “Network” column contains the home agent address that is entered in the mobile node’s RADIUS profile. The “Address” entry on that same line contains slot, port, and DLCI or VPI/VCI information that identifies the physical interface to the home network. Although the specific interface name is not reported with the address information, you can use the `/etc/aitmd.conf`, `/etc/grfr.conf`, or `/etc/gratm.conf` files to verify the displayed configuration.

If you configure a VPN address under the interface name of the home network record in `/etc/aitmd.conf`, that private network address will be displayed. In this example, it is 10.20.2.221. A unique identifier for the private network, in this case, vpn3708730531, is also included, but it reflects internal information not useful for checking a configuration.

Note: An asterisk (*) indicates an inactive interface.

Using maint commands

The next area to troubleshoot is ATMP at the media card level.

Two **maint** commands, **maint 70** and **maint 73**, provide useful information about the ATMP components configured on each media card.

To use **maint** commands, you start the **grrmb** program on the target media card that you want to run the **maint** command on. From the CLI prompt or the UNIX shell, enter **grrmb**:

```
# grrmb
GR 66>
```

Now change the prompt number to the slot number of the target card. For the card in slot 2, enter **port 2** and then you can enter a **maint** command to act on that media card:

```
GR 66> port 2
Current port card is 2
GR 2> maint 70
```

List home networks configured per HSSI or ATM card

The **maint 70** command lists the home agents and home network circuits associated with a media card. The `State` column also reports configuration status per home agent and home network circuit. Use this command to verify your configuration parameters.

The **maint 70** columns are as follows:

- **FRT-index** (Foreign agent Route Table)
  is an arbitrarily-assigned home network index, not the tunnel ID, but the number you use in the **maint 73** command to display the tunnel ID.
- **S/P/s0/s1**
  is the slot, port, and DLCI or VPI/VCI number of the tunneled circuit to the home network, depending upon whether the card is HSSI Frame Relay or ATM.

- **State** indicates configuration and functional status depending upon the type/role of card.
  - On the home agent card (usually an Ethernet card acting as the home agent)

  The foreign agent uses an IP address on this card for tunnel negotiation. A state of `HomeAgent` indicates a home agent is properly configured to this card in `aitmd.conf`.
  - On any HSSI or ATM card on a GRF home agent, State can have three values:

  1. `HomeAgent` indicates that the home agent is properly configured in `/etc/aitmd.conf`, but this is not the circuit to the home network.
  2. `LocalCirc` indicates that this circuit is not part of ATMP configuration.
  3. `LocCir,HA` = the circuit to the home network is configured on this media card.
- **Address** is the IP address of the associated home agent.
- **VPN Address** is the private network address the customer assigns to the interface that has the circuit to a home network, it only appears if entered in `/etc/aitmd.conf` file.
- **VPN Netmask** is the netmask for the VPN address.

Please ignore the “`Rx: packets Received, BRx: Bytes Received`” and “`RTx packets transmitted, BTx: Bytes transmitted`” headings, they no longer apply.

maint 70 - Ethernet card example

This is the Ethernet card with the GRF home agent IP address that the foreign agent uses to negotiate a tunnel. The `S:P:s0:s1` column points to the interface which has the circuit to the home network, in the example it is an ATM card. In a correctly configured system, "HomeAgent" is the state you will see on the Ethernet card. This indicates that the Ethernet card “knows” the home agent exists in `aitmd.conf`. The Address column shows the home agent IP address assigned to the Ethernet card. The next two columns show the VPN address and netmask assigned in `aitmd.conf` to the interface which has the circuit to the home network.

```
# grcard
 0      HSSI_V1        running
 1      HSSI_V1        running
 2      ATM_OC3_V2     running
 3      ETHER_V1       running

Current port card is 3
GR 3> maint 70
[RX]
[RX] H O M E  N E T W O R K  T B L  :
[RX] =================================
[RX] S: Slot, P: Port, Rx: packets Received, BRx: Bytes Received
[RX]                  RTx packets transmitted, BTx: Bytes transmitted
[RX]
[RX]FRT-index  S:P:s0:s1     State   Address  VPN Address  VPN Netmask
[RX]--------   ---------     -----   -------  -----------  -----------
[RX]   0   02:00:0015:0511  HomeAgent 221.1.1.4  10.20.2.237 255.255.255.252
[RX]
[RX] Entries: 1
GR 3>
```

maint 70 - ATM or HSSI card with circuit to home network

The circuit to the home network is indicated by `LocCir,HA` in the State column:

```
GR 3> port 2
Current port card is 2
GR 2> maint 70
[RX]
[RX] H O M E  N E T W O R K  T B L :
[RX] =================================
[RX]
[RX] S: Slot, P: Port, Rx: packets Received, BRx: Bytes Received
[RX]                  RTx packets transmitted, BTx: Bytes transmitted
[RX]
[RX]FRT-index  S:P:s0:s1     State     Address   VPN Address  VPN Netmask
[RX]--------   ---------     -----     -------   -----------  -----------
[RX]   0   02:00:0015:0511  LocCir,HA 221.1.1.4  10.20.2.237 255.255.255.252
[RX]
[RX] Entries: 1
```

maint 70 - ATM or HSSI card with circuit, possible ATMP problem

In this example, only a circuit is configured in `/etc/grfr.conf`, but the home agent is not set up in `/etc/aitmd.conf` or **aitmd** may not be running.

```
GR 1> maint 70
[RX]
[RX] H O M E  N E T W O R K  T B L :
[RX] =================================
[RX]
[RX] S: Slot, P: Port, Rx: packets Received, BRx: Bytes Received
[RX]                  RTx packets transmitted, BTx: Bytes transmitted
[RX]
[RX]FRT-index  S:P:s0:s1       State     Address   VPN Address  VPN Netmask
[RX]--------   ---------       -----     -------   -----------  -----------
[RX]     0    01:00:0101:0000  LocalCirc 0.0.0.0
[RX]
[RX] Entries: 1
```

Display tunnel information - maint 73

The **maint 73 *tunnel_number*** command shows tunnel information.

The **maint 73** columns are as follows:

- mobile node non-routable IP address
- number of bits in the address netmask
- route flags, currently ignored
- foreign agent routable IP address
- tunnel ID
- slot, port, and DLCI or VPI/VCI number of the tunneled circuit to the home network

Obtain the tunnel number using the **maint 70** command. The tunnel number is the entry under FRT-Index.

In this example, we look at index 0 and index 1:

```
GR 0> maint 73 0
[RX]
[RX] Mobile node tree list
[RX] Mobile Node/Mask  Flags  Foreign Agent   Tunnel Id  Slot:Port:s0:s1
[RX] 10.20.2.120/29     0 =>  206.146.160.181 0x00000279  2:0:0015:0511

maint 73 1
[RX]
[RX] Mobile node tree list
[RX]    No home network found at index 1
GR 0>
```

This **maint 73 0** command shows a tunnel for the mobile node using address 10.20.2.120 with 29 bits of netmask, connecting to a foreign agent at address 206.146.160.181. The tunnel ID is 0x279. The ATMP gateway circuit is on slot 2, port 0, VPI/VCI 15/511.

The “`no MN tree`” message usually indicates that there are no tunnels currently active. If you suspect a problem, use **maint 70** to check the configuration:

```
GR 1> maint 73 1
[RX]
[RX] Mobile node tree list
[RX]    Home network at index 1 has no MN tree
GR 1>
```

tcpdump

tcpdump can decode GRE-encapsulated packets.

ATMP statistics - grstat commands

grstat display commands return useful information about ATMP circuits on HSSI and ATM OC-3c media cards.

Common IP statistics

Look at IP and ATMP packet counts per logical interface:

The **grstat ipstat *interface*** command displays IP and ATMP packet counts for that interface. This example shows information for interface `ga027f`, an interface from the **netstat -i** example with a vpn address:

```
# grstat ipstat ga027f
ga027f
  ipstat
       count description
       21913 total packets received
       21913 packets forwarded normally
       21913 packets ATMP encapsulated
```

Look at ATMP packet counts per logical interface:

When the card is a home network gateway, the ATMP-related IP counts are reported:

```
# grstat ipstat ge031
ge031
  ipstat
       count description
   622199438 total packets received
   311099718 packets dropped
           2 packets forwarded to the RMS
   311099718 packets ATMP decapsulated
```

Look at packets dropped per media card

```
# grstat ipdrop 1
card 1 (2 interfaces found)
  ipdrop totals
       count description
   311099718 ATMP err: bad GRE header
```

Look at packets dropped per interface

```
# grstat ipdrop ge031
ge031
  ipdrop
                        last            last
       count     source addr       dest addr reason
   311099718       205.1.1.1       205.1.1.2 ATMP err: bad GRE header
```

Look at the IP counts and layer 2 statistics

```
# grstat ipstat ge034
ge034
  ipstat
       count description
        7971 total packets received
        7969 packets forwarded normally
           2 packets forwarded to the RMS

# grstat l2 ge034
ge034
  Layer 2 statistics
    physical port 4
               count description
                8397 RX packets
            11293604 RX bytes
                   1 CRC errors
                8281 TX packets
              513891 TX bytes

# grstat switch ge034
ge034
  Switch statistics
               count description
                7857 RX packets
              629312 RX bytes
           311107691 TX packets
        479104990712 TX bytes
                   1 Switch receiver reset
```

Note: The following IP forwarding statistics are maintained for all encapsulated ATMP packets:

– number of packets forwarded normally (*without* pre- or post-fragmentation)

– number of packets fragmented (*with* pre- or post-fragmentation)

– number of fragments created (*with* pre- or post-fragmentation)

Fragmentation statistics

When pre-fragmentation is specified, the **grstat** command returns related information. Pre-fragmentation statistics are incorporated into the existing IP statistics reported by **grstat**. These include:

– number of packets fragmented before encapsulation

– number of force-fragmented packets

Common IP statistics that reflect pre-fragmentation functions include:

– number of packets rejected because the DF bit was set

– number of packets fragmented, including those pre-fragmented

– number of fragments created

These statistics enable users to distinguish unfragmented, pre-fragmented, and post-fragmented packets.

Look at pre-fragmentation counts per logical interface:

When pre-fragmentation is enabled, the **grstat** command returns related information.

```
# grstat ipstat ga027f
ga027f
  ipstat
       count description
       21913 total packets received
       21913 packets forwarded normally
       21913 packets ATMP encapsulated
```

Look at pre-fragmentation counts per card:

```
# grstat ipstat 1
card 1 (2 interfaces found)
  ipstat totals
     count description
     36778 total packets received
     20290 packets dropped
     16488 packets forwarded to the RMS
     16488 multicast packets received
     16488 multicast packets forwarded to the RMS
       493 packets ATMP encapsulated with pre-fragmentation
       247 packets ATMP encapsulated with pre-fragmentation and
              mtu_limit override
         4 packets ATMP encapsulated with pre-fragmentation, clearing DF
```

Refer to the *GRF Reference Manual* or check the **grstat** man page for more information about using the **grstat** command.

Frame Relay ATMP statistics - grfr commands

The **grfr** command has display commands that return useful information about ATMP on HSSI Frame Relay circuits.

Display PVC statistics

This command displays the statistics for configured Frame Relay PVCs.

Enter: `grfr -c dps`

```
C O N F I G U R E D   P V C s   S T A T s:
=========================================

(S=Slot, P=Port, R=receive, T=Transmit)
(TP=Transmitted Packets, TO=Transmitted Octets)

Name      S/P/DLCI  Type    RPackets  R-Octets  T-Packets T-Octets  TP-Dropped TO-Dropped
----      --------  ----    --------  --------  --------- --------  ---------- ----------
2:0:0     02:0:0    Switch  2793      81044       2791     39074       0          0
2:0:160   02:0:160  ATMP    10266     1038795     10225    573187      0          0
2:0:218   02:0:218  Route   10270     863275      10279    1327565     0          0
2:0:329   02:0:329  Switch  10711     149954      10708    257072      0          0
eth1-tst  02:0:350  Route   3         90          0        0           0          0
hss8-tst  02:0:100  ATMP    6955134   618713108   6989490  637473637  14      20706
hss9:201  02:0:201  ATMP    0         0           0        0           0          0
```

Display media card interface status

This example shows that the two physical interfaces P0 and P1 on the HSSI media card in slot 2 are UP and in the RUNNING state. There are two HSSI cards installed, but the one in slot 3 is not running.

```
# grfr -c dbs

P O R T - C A R D   H W   S T A T U S :
=======================================

Slot   Type   State     P0   P1   P2   P3   P4   P5   P6   P7
----   ----   -----     --   --   --   --   --   --   --   --

0      ----
1      ----
2      HSSI   RUNNING   UP   UP   --   --   --   --   --   --
3      HSSI   INACTIVE  DOWN DOWN
4      ----
5      ----
6      ----
7      ----
8      ----
9      ----
10     ----
11     ----
12     ----
13     ----
14     ----
15     ----
```

Display link configuration and status

This command shows the status of configured links and their current parameters. Enter:

```
# grfr -c dlc

C O N F I G U R E D   L I N K S :
=================================
Name:         S/P:  LMI:     Link:    Autogrif: N391: N392: N393: T391: T392: Statu:
----          ---   ---      ----     --------  ----  ----  ----  ----  ----  --
Slot 9, Sone  9 /0  ANNEX-A  UNI-DCE  Auto      6     3     4     10    15    Inactive
Sonet2        9 /1  ANNEX-A  UNI-DCE  None      6     3     4     10    15    Active
Slot 13, Upp  13/0  ANNEX-D  UNI-DCE  gs0d1     6     3     4     10    15    Active
Slot_13_Lowe  13/1  ANNEX-A  UNI-DCE  gs0d80    6     3     4     10    15    Active
Total: 4 links configured
```

Display configured PVCs

```
# grfr -c dpc

C O N F I G U R E D   P V C s   :
=================================
(A* = Autoadded, D* = Deleted)
Name          Slot  Port  DLCI   Type    CIR   Bc    Be    State     EPs/ISIS
----          ---   ----  ----   ----    ----  ---   ---   -----     --------

0:0:0         0     0     0      Switch  55K   55K    0K   Active    0:0:0
Headroom-pub  0     0     200    Route   55K   55K    0K   Inact     NO-ISIS

0:1:0         0     1     0      Switch  55K   55K    0K   Active    0:1:0
wg-atmp       0     1     401    ATMP    30M   30M   30M   Active
wg-route      0     1     402    Route   22M   22M    0K   Active    NO-ISIS

2:0:0         2     0     0      Switch  22K   22K    0K   Active    2:0:0
HN1-eth-dumm  2     0     32     Route   22K   22K    0K   Active    NO-ISIS
HN1-eth       2     0     100    ATMP    22K   22K    0K   Active
vpn:201       2     0     201    ATMP    30M   30M    0M   Active
vpn:202       2     0     202    ATMP    30M   30M    0M   Active
vpn:203       2     0     203    ATMP    30M   30M    0M   Active
vpn:204       2     0     204    ATMP    30M   30M    0M   Active
vpn:205       2     0     205    ATMP    30M   30M    0M   Active
vpn:206       2     0     206    ATMP    30M   30M    0M   Active
vpn:207       2     0     207    ATMP    30M   30M    0M   Active
vpn:208       2     0     208    ATMP    30M   30M    0M   Active
vpn:209       2     0     209    ATMP    30M   30M    0M   Active
vpn:210       2     0     210    ATMP    30M   30M    0M   Active

2:1:0         2     1     0      Switch  55K   55K    0K   Active    2:1:0
wg20          2     1     20     Route   30M   30M   30M   Active    NO-ISIS
wg22          2     1     22     Route   30M   30M   30M   Active    NO-ISIS
HN2-hssi      2     1     101    ATMP    55K   55K    0K   Active

Total  9 PVCs configured
       5 Routed PVCs
       4 Switched PVCs
       0 Multicast PVCs
       13 ATMP PVCs
```

Display system configuration and status

```
# grfr -c dsc

S Y S T E M   P A R A M E T E R S :
===================================
Name:.......................... X
Time and Date compiled ........ Thu Jul 23 03:32:41 CDT 1998
Compiled from source in ....... /A1_4_10/BSDI/usr.sbin/fred
Start Time .................... Sat Aug  2 17:45:55 CDT 1997
Up-time ....................... 41 days, 5 hours, 44 mins, 15 secs
Configuration File ............ /etc/grfr.conf
grif Configuration File ....... /etc/grifconfig.conf
Debug Level.................... 1
Statistics Interval............ 10
Portcard Heartbeat Interval.... 10
Media Types Supported.......... HSSI, SONET-OC3
Boards configured ............. 2
Links  configured ............. 4
PVCs  configured .............. 22
   Routed PVCs  configured .... 5
   Switched PVCs  configured .. 4
   Mcasted  PVCs  configured .. 0
   ATMP PVCs configured ....... 13
Active Links .................. XX
Active PVCs ................... XX
```

Display configured interfaces

This command shows the list of configured interfaces and their current parameters. Enter:

```
# grfr -c dic

C O N F I G U R E D   I N T E R F A C E S:
==========================================
gr-interface: gs020, if_num: 0x0, slot = 2
gr-interface: gs021, if_num: 0x1, slot = 2
gr-interface: gs022, if_num: 0x2, slot = 2
Total: 3 interfaces configured
```

Adding/deleting PVCs on-the-fly

You can add or delete PVCs without resetting the media card by editing the `/etc/grfr.conf` file and then using a **grfr -c ccp *slot link dlci*** command to add or a **grfr -c crp *slot link dlci*** command to delete.

To add a PVC to the card in slot 13, start the UNIX shell and first edit `/etc/grfr.conf`:

```
super> sh
# vi /etc/grfr.conf
```

Make the PVC entry as usual:

```
#   lif    DLCI Peer IP Address Optional Parameters
#   ===    ==== ============== ===================
pvc gs0d0  606  0.0.0.0   Name="test606"
```

Save the file and exit **vi**.

Enter the **grfr -c ccp** command to add a PVC. The configuration file and the PVC slot, link, and DLCI must be specified:

```
# grfr -c ccp -f /etc/grfr.conf -s 13 -l 0 -i 606
```

Here is the response:

```
grfr Adding type 1, lif=gs0d0, dlci=606, peer_ip=0.0.0.0
        Slot =13, link =0, name=test606
PVC slot 13, link 0, dlci 606 defined
```

To delete (disable) a PVC, you do not need to edit the `/etc/grfr.conf` file, the **grfr -c crp *slot link dlci*** command is sufficient.

Specify the target DLCI to be disabled:

```
# grfr -c crp -s 13 -l 0 -i 600
```

Here is the response:

```
PVC slot 13, link 0, dlci 600 deleted
```

Transparent Bridging

13

Chapter 13 describes the implementation of transparent bridging on the GRF. This includes configuration information and the use of `/etc/bridged.conf`.

Chapter 13 describes the following topics:

GRF bridging implementation

The GRF implements IEEE 802.1D transparent bridging on GRF Ethernet and FDDI interfaces, and on ATM OC-3c interfaces using RFC 1483 encapsulated bridging over PVCs.

Transparent bridging provides a mechanism for interconnecting stations attached to physically separate Local Area Networks (LANs) as if they are attached to a single LAN. This interconnection happens at the 802 MAC layer, and is transparent to protocols operating above this boundary in the Logical Link Control (LLC) or Network layers. Participating stations are unable to identify that peers are on anything other than the directly-attached physical media.

The GRF implementation consists of the transparent bridging function described in 802.1D, and does not include any capability for Source Route or Source Route Transparent (SRT) bridge operation.

Feature summary:

- bridging on FDDI, Ethernet, and ATM OC-3c per the 802.1D standard
- participation in 802.1D spanning tree protocol
- layer-2 transparent bridging of MAC frames through the GRF from one interface to another.
- conversion of frames between Ethernet and FDDI formats as necessary
- fragmentation of IPv4 frames if necessary
- simultaneous bridging and routing over the same interface
  (a GRF interface participating in a bridge group can still route normally)
- routing IP to or from a bridge group from any GRF media
- RFC 1483 encapsulated bridging over ATM OC-3c PVCs with either VC-based multiplexing or LLC encapsulation
- multiple independent bridge groups per GRF
- up to 255 GRF interfaces per bridge group

Specifications

The GRF bridging implementation reflects the following documents:

- International Standard ISO/IEC 10038: 1993;
  ANSI/IEEE Standard 802.1D, 1993 edition
- International Standard ISO 8802-2;
  ANSI/IEEE Standard 802-2, 1989 edition
- RFC 1483, J. Heinanen,
  Multiprotocol Encapsulation over ATM Adaptation Layer 5, 07/20/1993.
  Available via ftp at: `ftp://nic.ddn.mil/rfc/rfc1483.txt`

Simultaneous routing and bridging

Ascend’s transparent bridging does not preclude the use of IP packet routing on the same physical interface.

Bridging as well as IP version 4 (IPv4) routing can both be enabled on the same physical interface. In this circumstance, the GRF exchanges traffic between bridging domains and routing domains that exist on the same physical media.

A GRF interface may simultaneously bridge layer-2 frames and route layer-3 packets--that is, forward frames destined to a system attached to another LAN at the MAC layer, but still receive IP packets destined for a remote system attached to a non-broadcast GRF interface and route those packets at the IP layer.

This unique capability eliminates the need for separate pieces of routing equipment to transport packets inter-domain.

To perform the simultaneous functions, the GRF bridging interface examines the destination MAC address of each arriving frame. If the address is *other than* a GRF MAC address for any interface participating in the assigned bridge group, the packet is submitted to the bridging engine for forwarding. When the MAC address is a GRF MAC address, the packet is forwarded to the GRF protocol forwarding engine for routing at the protocol layer. Multicast and broadcast frames are submitted to both engines.

Configuration options

The GRF supports the configuration items specified in 802.1D. A GRF functioning as a bridge will interoperate with other bridges, including equipment of vendors in conformance with the IEEE 802.1D standard, to allow forwarding of frames across multiple LAN hops.

Additionally, the GRF supports up to 64 independent 802.1D bridge groups, and separates traffic between groups. For example, on a GRF with six attached FDDI rings, rings A, B, and C could form one bridge group, rings D and E could form a second bridge group, and ring F could stand alone, using only IP routing for its packets.

A GRF functioning as a bridge also will interoperate with other bridges to forward frames from one bridge to the other over ATM. This will allow two independent bridged LANs at remote locations to function as one logical network transparently connected by ATM. This encapsulated bridging follows the Internet standard specification in RFC 1483.

Interoperability

FDDI - Frame forwarding is compatible with any station sending and receiving FDDI LLC frames.

Ethernet - Frame forwarding is compatible with any station using either DIX Ethernet or IEEE 802.3 frames.

ATM OC-3c - Frame forwarding is compatible with any remote bridge using RFC 1483 bridging encapsulation.

Spanning tree - GRF transparent bridging will interoperate with any other bridge (including other GRFs) compliant with the IEEE 802.1D spanning tree protocols.

Spanning tree

The GRF implementation supports the full Spanning Tree Algorithm specified in the IEEE 802.1D standard.

Using the Spanning Tree, network topologies can contain cycles that can be used as redundant or back-up links. The Spanning Tree controls the bridge's flow of traffic over all potential links to prevent packet storms (bridges repeating a packet or packets to each other, without end).

Consistent with basic GRF architecture, the Spanning Tree Algorithm and all controlling configuration and bridging information is maintained on the control board. A copy of the bridging filtering table is maintained on each media card.

Bridge filtering table

Media card bridge ports forward new MAC source addresses to the operating system for insertion in the global bridge filtering table that is maintained on the control board. Each bridging media card type (FDDI, Ethernet, and ATM OC-3c) also has a copy of this table. Batches of table updates are sent out to all bridging media cards in the same way IP route table updates are dispersed to the media cards.

Bridge ports also "age" entries according to the 802.1D protocol. When no activity is associated with a MAC address for the specified time-out interval, the interface sends the operating software a delete request and the address is removed first from the global bridge filtering table and then, via the update packets, from media cards' tables.

Fragmentation

IPv4 frames are fragmented as necessary, as when bridging an FDDI frame of more than 1500 bytes to an Ethernet interface.

A frame may be too large for the maximum transmission unit of the sending GRF interface. One example is when forwarding a 4500-byte frame from FDDI to an Ethernet interface with an MTU of 1500 bytes. The GRF bridge will attempt to break such a frame into fragments that will fit the sending interface. This is possible if the frame contains an IP datagram; then the GRF may use the fragmentation rules of IP to split the frame. Otherwise, the GRF must drop the frame.

Spamming

Spamming occurs when a bridging interface forwards a frame to all active interfaces in the bridge group. On the GRF, spamming is done when a broadcast or multicast address is received, or when a frame arrives whose destination address is not in the bridge filtering table.

GateD

GateD treats a bridge group interface as a single interface. Individual member interfaces are not considered in GateD operation.

Bridging components

Bridging daemon – bridged

The bridging daemon, **bridged**, configures and manipulates bridging interfaces on the GRF. It operates the spanning tree algorithm specified in IEEE 802.1D and ensures interoperability with other 802.1D bridges.

bridged reads the `/etc/bridged.conf` configuration file to build an initial bridging topology. The `bridged.conf` file is read whenever **bridged** is restarted. Refer to the **bridged** man page for more information.

bridged is started by the system script `/etc/grstart`. This script monitors the **bridged** daemon and restarts it if **bridged** stops. **bridged** is run from its installed location `/usr/sbin/bridged`.

Configuration file – bridged.conf

The bridging configuration file is `/etc/bridged.conf`. A utility, **bredit**, is used to access the file and create bridge groups and bridging settings.

Parameters in `bridged.conf` can be set to:

– name bridge groups

– assign interfaces (bridge ports) to a group

– assign priority, root path cost, and forwarding addresses to individual interfaces

– assign hello time and forwarding delay values, priority, maximum age, and discard addresses to individual groups

A copy of the `/etc/bridged.conf` file is in the *GRF Reference Guide.*

Editing utility – bredit

The **bredit** utility is used to access and edit the `bridged.conf` configuration file.

bredit opens the configuration file in the **vi** editor. After you make changes, you exit the file with the **vi** exit file **:wq** command.

At this point **bredit** asks if you want to make the changes permanent. You also have the option of signaling **bridged** to re-read the updated file immediately. When this option is taken, **bridged** restarts as if it was stopped and restarted for the first time. If you change the file in **vi** but do not choose either of the options, **bredit** tells you that your changes were not committed.

Management tools

A set of tools are provided to manage bridging, primarily through **bridged**. Brief descriptions are provided here, more detail is given in the *Examining and debugging bridge configurations* section near the end of this chapter.

These tools include:

- **brstat**, displays relevant **bridged** status and bridging information
- **brinfo**, displays relevant kernel-based bridging information

brstat

The **brstat** command provides a snapshot of state information directly from **bridged**. A short lag occurs between the time a request is made and when an active **bridged** returns the information.

```
super> brstat
```

brinfo

The **brinfo** command is used to retrieve bridging interface information for administrative debugging and other situations where a simple checking of bridge group or bridge port information is needed.

super> brinfo *bridge_group* | all

or

super> brinfo *bridge_port* | all

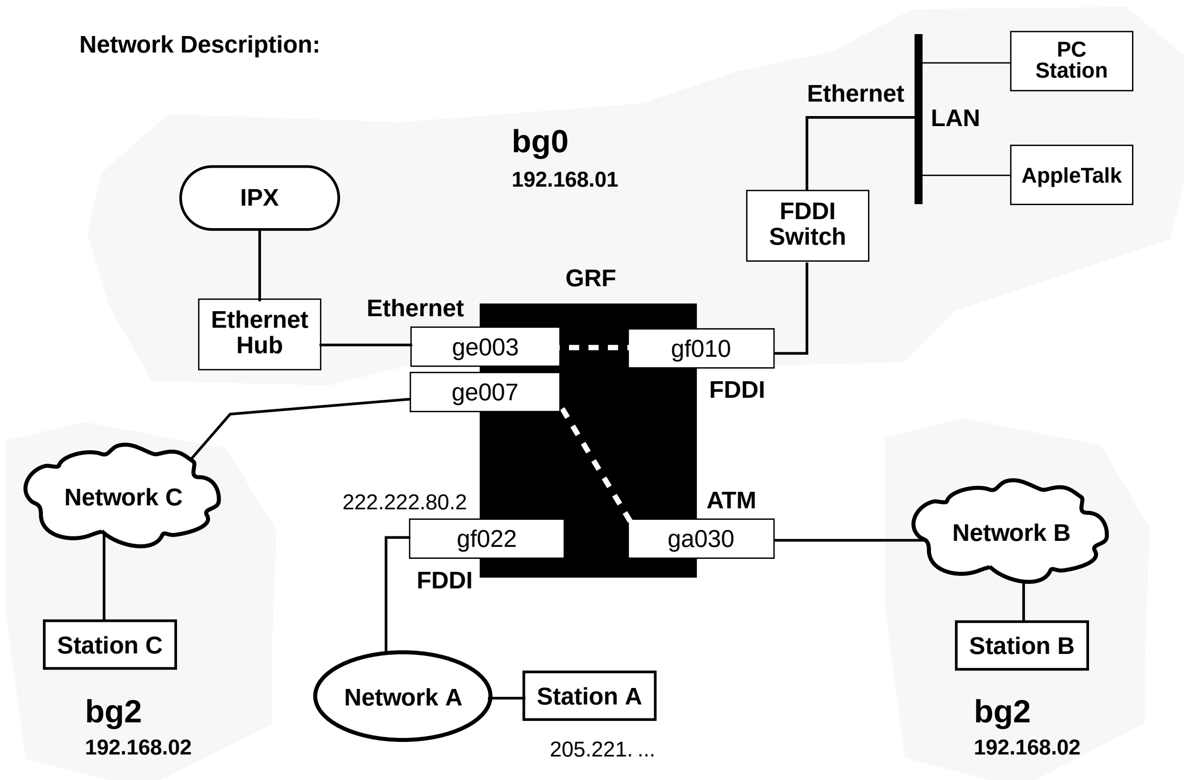
If a bridge group is specified, **brinfo** prints information about the group and the bridge ports (underlying interfaces) that are members of the specified group. If a bridge port (interface name) is specified, **brinfo** displays the specified interface. If no parameters are specified, all groups are reported on by default.

brinfo gets its information directly from the BSD kernel whereas **brstat** gets its information from **bridged**.

Bridging example

In this example, bridge group `bg0` is a single shaded area. Two GRF interfaces, one Ethernet and one FDDI (`ge003` and `gf010`), form the bridge between the IPX services and the Ethernet LAN. Bridge group `bg2` has two LANs, each with a GRF interface, one Ethernet (`ge007`) and one ATM (`ga030`). Interface `gf022` is IP routing only. Station A can route to any station in either bridge group.

Network Description:



Configuration tasks:

1. Enter IP addresses in `grifconfig.conf`:

```
bg0     192.168.01
bg2     192.168.02
gf022   222.222.80.2
```

2. Define bridge groups in `bridged.conf`:

```
bridge_group bg0        bridge_group bg2
port ge003, gf010       port ge007, ga030
```

3. Set Network A for normal routed network.
4. Configure a bridging PVC on `ga030` in `gratm.conf`.

Figure 13-1. Bridging example diagram

The GRF currently supports up to 64 bridge groups with as many as 255 logical GRF interfaces assigned to each group. A logical interface can be a member of only one bridge group.

From a GRF perspective, a bridge group equals a virtual LAN.

Configuration file and profile overview

When a new GRF system is installed or a site upgrades to a bridging software release, the bridging daemon, **bridged**, is automatically started.

These are the steps to configure bridging interfaces and parameters:

1 Create bridge groups in bridged.conf

Run **bredit** to access and edit the `/etc/bridged.conf` configuration file. Create and name the bridge groups, and assign bridge ports and parameters to each.

2 Assign an IP address to each bridge group

Step 2 is necessary only if you want to do simultaneous bridging and routing.

Edit `/etc/grifconfig.conf` to identify each bridge group by assigning:

– an IP address

– the GRF interface name

– a netmask, required

– a broadcast address, as required

Execute a **grifconfig** command for each group.

Note: Members of bridge groups are not assigned IP addresses in `/etc/grifconfig.conf`. In the example from the preceding page, only the FDDI interface, `gf022`, and the bridge groups, `bg0` and `bg2`, are assigned IP addresses.

3 Create ATM OC-3c PVCs for encapsulated bridges

To configure an encapsulated bridge on an ATM circuit, edit the `/etc/gratm.conf` file to create a PVC on the ATM OC-3c logical interface.

1. Create bridge groups in bridged.conf

The **bredit** utility is used to access and edit the `bridged.conf` configuration file.

bredit opens the configuration file in the **vi** editor. After you make changes, you exit the file with the **vi** exit file **:wq** command.

Here are the syntax conventions in the `/etc/bridged.conf` file:

{ - the brace after the group name indicates the start of the group's parameters

; - a semi-colon indicates the end of the arguments for a statement

};- a closing brace and semi-colon indicate the end of the block

The format of a group name is:

```
bridge_group bgA { arguments } ;
```

where `A` is a decimal number from 0 through 63

The only required parameter is the list of GRF interfaces you are assigning to the group.

The format of the group list is:

```
bridge_group bgA {
port interface_name;
};
```

where *`interface_name`* is in the standard GRF interface name format `gx0yz` that uniquely describes a logical FDDI, Ethernet, or ATM OC-3c interface.

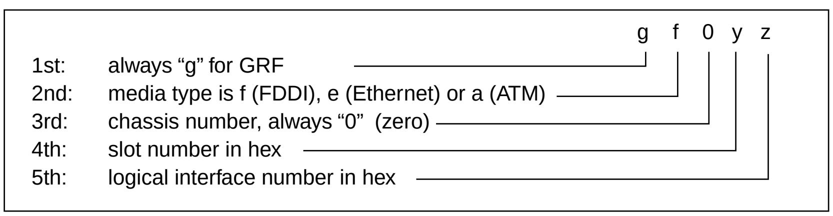


Figure 13-2. Interface name for FDDI, Ethernet, and ATM OC-3c interfaces

List one port on a line, or list them all on one line as shown in these examples:

```
port ge003;
port gf010;
```

or

```
port ge003 gf010;
```

A simple bridge group entry is:

```
bridge_group bg0 {
port ge003;
port gf010;
};
```

An empty bridge group is defined in this way:

```
bridge_group bg5 {
            ;
};
```

2. Assign IP addresses to bridge groups

To do simultaneous bridging and routing, assign an IP address to each bridge group in the `/etc/grifconfig.conf` file.

These are the entries in `grifconfig.conf` for bridge group `bg0` and `bg2` and the non-bridging interfaces shown in the example in Figure 13-1:

```
# name    address          netmask            broad_dest        argument
#
bg0       192.168.01.1   255.255.255.0
bg2       192.168.02.1   255.255.255.0

gf022     222.222.80.2   255.255.255.0
```

A netmask entry is required for each bridge group.

Finally, you must execute a **grifconfig** command for each bridge group you have assigned an IP address. In this example, you execute two commands:

```
# grifconfig bg0
# grifconfig bg2
```

3. Create an ATM PVC for an encapsulated bridge

Bridging over ATM can be configured in two ways:

– LLC Encapsulation (RFC 1483, section 4)

– VC Based Multiplexing (RFC 1483, section 5)

When LLC Encapsulation is used, a single PVC is configured to carry all traffic.

When VC Based Multiplexing is used, multiple PVCs are defined for the logical interface. Each PVC carries a specific type of traffic. For example, one PVC carries Ethernet PDUs while another carries FDDI.

Configuration in /etc/gratm.conf

Configuration over ATM also requires that new entries be made to three sections of the regular ATM configuration file, `/etc/gratm.conf`.

The next three steps describe ATM bridging configuration requirements and options. Examples of configured PVCs follow.

1 In the Traffic Shaping section of the `/etc/gratm.conf` file, set traffic shaping name and quality of service parameters, use any string. Set a name for each type of service that will be assigned.

The `/etc/gratm.conf` file itself describes how to specify a range of traffic shaping parameters.

```
# Traffic shaping parameters
# Lines beginning with the keyword "Traffic_Shape" define
# traffic shapes which may be used to configure the performance
# characteristics of ATM Virtual Circuits.
#
Traffic_Shape name=high_speed_high_quality \
       peak=155000 sustain=155000 burst=2048 qos=high
```

2 To configure a logical interface for bridging, you create an Interface entry in the Interface section of the `/etc/gratm.conf` file. This entry must include the intended bridging method, specify this with the `bridge_method=` keyword.

Here is a sample Interface entry:

```
Interface ga030 traffic_shape=high_speed_high_quality \
bridge_method=vc_multiplexed,broute_to_ether
```

You can specify two types of bridging methods, VC multiplexed or LLC encapsulated:

- VC Based Multiplexing, `bridge_method=vc_multiplexed`

  The configuration must include one or more PVCs for this interface specified in the PVC section and defined (as described below) with `proto=vcmux_bridge`.

  VC multiplexed requires an entry in `/etc/grarp.conf`:

```
#/etc/grarp.conf
ga030 0/50 xx.xx.xx.xx  ## xx.xx.xx.xx is destination IP addr
```

- LLC Encapsulation, `bridge_method=llc_encapsulated`

The configuration must include one PVC for this interface specified in the PVC section and defined with `proto=llc,bridging`.

Restrictions

Media and transmission restrictions are specified for both types of bridging methods using the `broute_to_ether`, `ether_only`, `broute_to_fddi`, or `fddi_only` keyword.

In this list, `xxxx` represents either the `llc_encapsulated` or `vc_multiplexed` bridging method:

– bridge_method=xxxx,broute_to_ether

  IP and ISO datagrams are transmitted as Ethernet frames.

– bridge_method=xxxx,ether_only

  All frames except BPDUs (routed datagrams and all bridged LAN frame types) are transmitted as Ethernet frames.

– `bridge_method=xxxx,broute_to_fddi`

  IP and ISO datagrams are transmitted as FDDI frames.

– bridge_method=xxxx,fddi_only

  All frames except BPDUs (routed datagrams and all bridged LAN frame types) are transmitted as FDDI frames.

If an interface cannot be used to transmit a particular frame type directly, the GRF attempts to translate the frame to a permitted type.

For example, if an interface is defined to send Ethernet frames only and the GRF has an FDDI frame to transmit, the GRF translates the frame to an Ethernet frame first. Similarly, if the GRF has a routed IP datagram to transmit, the GRF adds an Ethernet header and transmits the datagram as an Ethernet frame.

3 One or more Permanent Virtual Circuits (PVCs) must be defined in the PVCs section for each logical interface specified for bridging in the Interfaces section.

A bridging PVC is assigned a protocol value. This value must be consistent with the bridging method defined for the logical interface. Bridging PVCs are assigned either one of these protocol values:

– proto=llc,bridging

– proto=vcmux_bridge,yyyy

proto=llc,bridging

This type of PVC is used for logical interfaces defined with `bridge_method=llc_encapsulated`. The PVC uses LLC encapsulation for each PDU.

For example, this PVC entry enables bridging on an LLC PVC:

```
PVC ga030 0/32 proto=llc,bridging traffic_shape=high_speed_high_quality
```

proto=vcmux_bridge,yyyy

This type of PVC is used only for logical interfaces defined with `bridge_method=vc_multiplexed`. The PVC carries bridged traffic of a single type.

The `yyyy` represents a second protocol qualifier required for the `proto=` parameter, either `ether_fcs`, `ether_nofcs`, `fddi_fcs`, `fddi_nofcs`, or `bpdu`. The second qualifier defines the type of bridged traffic the PVC can carry.

Traffic types include:

– proto=vcmux_bridge,ether_fcs

  Specifies that each PDU is an Ethernet frame, including a Frame Check Sequence.

– proto=vcmux_bridge,ether_nofcs

  Specifies that each PDU is an Ethernet frame, without a Frame Check Sequence.

– proto=vcmux_bridge,fddi_fcs

  Specifies that each PDU is an FDDI frame, including a Frame Check Sequence.

– proto=vcmux_bridge,fddi_nofcs

  Specifies that each PDU is an FDDI frame, without a Frame Check Sequence.

– proto=vcmux_bridge,bpdu

  Specifies that each PDU is an 802.1D Bridge Protocol Data Unit.

PVC configuration examples

LLC encapsulated, restricted to Ethernet

Here is a sample LLC Encapsulated configuration, restricted to Ethernet. Note that any IP or ISO routed traffic transmitted on the PVC will be encapsulated as an Ethernet frame.

```
# Traffic shape
Traffic_Shape name=high_speed_high_quality peak=155000 sustain=155000
burst=2048 qos=high
# Logical interface
Interface ga030 traffic_shape=high_speed_high_quality
bridge_method=llc_multiplexed,broute_to_ether
# PVC
PVC ga030 0/32 proto=llc,bridging
```

VC-based multiplexing options

Here is a sample VC Based Multiplexing configuration. Note that routed IP or ISO datagrams are encapsulated as Ethernet frames.

```
# Traffic shape
Traffic_Shape name=high_speed_high_quality peak=155000 sustain=155000
burst=2048 qos=high
```

```
# Logical interface

Interface ga030 traffic_shape=high_speed_high_quality
bridge_type=vc_multiplexed,broute_to_ether

# PVCs for bridging

PVC ga030 0/32 proto=vcmux_bridge,ether

PVC ga030 0/33 proto=vcmux_bridge,ether_fcs

PVC ga030 0/34 proto=vcmux_bridge,bpdu
```

Installing configuration changes

When you enter configuration information or make changes, you must do a **grwrite** command to save the `/etc` directory to permanent storage. In the CLI, or from the CLI UNIX shell, enter:

```
# grwrite -v
```

That command saves the `/etc` directory. You can find out at any time if there are unsaved files in that directory, use this version of **grwrite** to get a list of unsaved files:

```
# grwrite -vn
```

You must also reset the media card for the configuration changes to take place. Enter:

```
# grreset <slot_number>
```

Packet translation

This section provides packet translation formats for various types of frames.

Ethernet packet formats

An Ethernet frame can be in an Ethernet II, Ethernet 802.2, or Ethernet SNAP format. The formats are illustrated in the figures below:

Ethernet II

Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Field	Destination MAC Addr						Source MAC Addr						Ethertype	
	Ethernet MAC Header													

Figure 13-3. Ethernet II frame format

Ethernet 802.2

Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Field	Destination MAC Addr						Source MAC Addr						Length		SSAP	DSAP	CTL
	802.3 MAC Header														802.2 LLC Header		

Figure 13-4. Ethernet 802.2 frame format

Ethernet SNAP

Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Field	Destination MAC Addr						Source MAC Addr						Length		AA	AA	03	00-00-00			Ethertype	
	802.3 MAC Header														802.2 LLC Hdr			SNAP Header				

Figure 13-5. Ethernet SNAP frame format

FDDI packet formats

An FDDI frame can be in either FDDI 802.2 or FDDI SNAP format. The formats are illustrated in the figures below:

FDDI 802.2

Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FC	Destination MAC Addr						Source MAC Addr						DSAP	SSAP	CTL
	FDDI MAC Header													802.2 LLC Header		

Figure 13-6. FDDI 802.2 frame format

FDDI SNAP

<table>
<tr><td>Octet</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr>
<tr><td rowspan="2">Field</td><td>FC</td><td colspan="6">Destination MAC Addr</td><td colspan="6">Source MAC Addr</td><td>AA</td><td>AA</td><td>03</td><td colspan="3">00-00-00</td><td colspan="2">Ethertype</td></tr>
<tr><td colspan="13">FDDI MAC Header</td><td colspan="3">802.2 LLC Hdr</td><td colspan="5">SNAP Header</td></tr>
</table>

Figure 13-7. FDDI SNAP frame format

Default frame translation

Normally, the GRF does frame translation based solely on the packet format, without examining the protocol carried in each frame.

The translations are done as follows:

- from Ethernet II to FDDI SNAP
- from Ethernet 802.2 to FDDI 802.2
- from Ethernet SNAP to FDDI SNAP
- from FDDI 802.2 to Ethernet 802.2
- from FDDI SNAP to Ethernet II

IPX frame translation

A Novell client or server can be configured to use a nonstandard Ethernet packet format:

Ethernet 802.3 “Raw” (Novell)

This format is identical to Ethernet 802.2, except that the LLC header is replaced with FF-FF-FF-FF.

<table>
<tr><td>Octet</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18...</td></tr>
<tr><td rowspan="2">Field</td><td colspan="6">Destination MAC Addr</td><td colspan="6">Source MAC Addr</td><td colspan="2">Length</td><td colspan="4">FF-FF-FF-FF</td><td>...</td></tr>
<tr><td colspan="14">802.3 MAC Header</td><td colspan="5">IPX Packet...</td></tr>
</table>

Figure 13-8. Ethernet 802.3 Raw frame format

To successfully translate these packets to FDDI, the GRF must be explicitly configured with the type of FDDI packet format to use, FDDI 802.2 or FDDI SNAP. In addition, each bridge port can be configured with the packet type to use for all IPX frames that are translated from other media types. The configuration is done for the outbound interface, i.e, an FDDI interface is configured with the packet type to use when an Ethernet packet is translated to FDDI for transmission. (Note that the configuration does not apply to frames that do not need translation.)

IPX translation configuration is accomplished with the `ipx_translate_to_ethernet` and `ipx_translate_to_fddi` keywords in `bridged.conf`. The `ipx_translate_to_ethernet` keyword applies only to Ethernet and ATM interfaces. The `ipx_translate_to_fddi` keyword applies only to FDDI and ATM interfaces. For example:

```
bridge_group bg0 {
   port ge020 { ipx_translate_to_ethernet ethernet_ii; };
   port gf000 { ipx_translate_to_fddi fddi_snap; };
   port ga031 {
    ipx_translate_to_ethernet ethernet_ii;
    ipx_translate_to_fddi fddi_snap;
  };
};
```

The following keyword combinations specified in bridged.conf are translated to formats as shown in Table 13-1.

Table 13-1.Keyword combinations with resulting packet formats

Keyword Combinations		**Transmitted Packet Format**
`ipx_translate_to_ethernet`	`ethernet_ii`	Ethernet II
`ipx_translate_to_ethernet`	`ethernet_802.2`	Ethernet 802.2
`ipx_translate_to_ethernet`	`ethernet_snap`	Ethernet SNAP
`ipx_translate_to_ethernet`	`ethernet_802.3_raw`	Ethernet 802.3 "Raw"
`ipx_translate_to_fddi`	`fddi_802.2`	FDDI 802.2
`ipx_translate_to_fddi`	`fddi_snap`	FDDI SNAP

IPX translation performance

Please note that packet translations between an 802.2 format (Ethernet 802.2 or FDDI 802.2) and any other format are much slower than other translations. These translations involve adding or removing an odd number of bytes in the header, and should only be used for low-bandwidth or temporary configurations.

Table 13-2 shows the packet translation rates between the Ethernet and FDDI formats.

Table 13-2.Ethernet and FDDI packet translation rates

From format	**To format**	**Translation rate**
Ethernet II	FDDI 802.2	s l o w
	FDDI SNAP	*fast*
Ethernet 802.3	FDDI 802.2	*fast*
	FDDI SNAP	s l o w
Ethernet SNAP	FDDI 802.2	s l o w
	FDDI SNAP	*fast*

Table 13-2.Ethernet and FDDI packet translation rates (continued)

From format	To format	Translation rate
Ethernet 802.3 "Raw"	FDDI 802.2	s l o w
	FDDI SNAP	*fast*
FDDI 802.2	Ethernet II	s l o w
	Ethernet 802.2	*fast*
	Ethernet SNAP	s l o w
	Ethernet 802.3 "Raw"	s l o w
FDDI SNAP	Ethernet II	*fast*
	Ethernet 802.2	s l o w
	Ethernet SNAP	*fast*
	Ethernet 802.3 "Raw"	*fast*

Sources of bridging data

Bridging trace log

The **-d level** option for the **bridged** command controls the type of messages collected in the `/var/tmp/bridged.trace` log.

The output shown here reflects level 5, the default, and adequate for most debugging. Enter:

```
# cd /var/tmp
# cat bridged.trace
```

```
1998.04.02.10:58:09.083 0n NOTICE main.c:210 main() started
1998.04.02.10:58:09.090 0n NOTICE br_init.c:423 add_modify() adding group 'bg0'
1998.04.02.10:58:09.137 0n NOTICE br_init.c:921
    initialize_bridge_unique_ids() bg0 MAC set to 00:c0:80:1c:05:81
1998.04.02.10:58:09.140 0n NOTICE br_init.c:463 add_modify() adding 'gf001' to 'bg0'
1998.04.02.10:58:09.141 0n WARNING support.c:153 support_attach_port() gf001 add port
    failed
1998.04.02.10:58:09.141 0n NOTICE br_init.c:463 add_modify() adding 'ge010' to 'bg0'
1998.04.02.10:58:09.142 0n WARNING support.c:153 support_attach_port() ge010 add port
    failed
1998.04.02.10:58:09.142 0n NOTICE standard.c:1112 std_initialisation() bg0 root [me]
1998.04.02.10:58:14.893 5s NOTICE signal.c:352 signal_reread_config() reread_config
    signal received
1998.04.02.10:58:14.895 5s INFO signal.c:335 reread_config() complete
1998.04.02.10:58:24.189 12t NOTICE support.c:141 support_attach_port() gf001 disabled --
    not running
1998.04.02.10:58:24.190 12t NOTICE br_reconfig.c:541 try_reattach() gf001 added to bg0
1998.04.02.10:58:24.410 15r NOTICE br_reconfig.c:264 check_bport_flags() gf001 running
1998.04.02.10:58:24.415 15r NOTICE standard.c:663
    std_become_designated_port() gf001 desig bridge [me] port 128/1
1998.04.02.10:58:24.415 15r NOTICE standard.c:1161 std_initialize_port() bg0.gf001
    Blocking(4)
1998.04.02.10:58:24.416 15r NOTICE standard.c:741 std_make_forwarding() bg0.gf001 For-
warding(3)
1998.04.02.10:59:34.252 58t NOTICE support.c:141 support_attach_port() ge010 disabled --
    not running
1998.04.02.10:59:34.254 58t NOTICE br_reconfig.c:541 try_reattach() ge010 added to bg0
1998.04.02.10:59:35.204 60r NOTICE br_reconfig.c:264 check_bport_flags() ge010 running
1998.04.02.10:59:35.204 60r NOTICE standard.c:663
    std_become_designated_port() ge010 desig bridge [me] port 128/2
1998.04.02.10:59:35.204 60r NOTICE standard.c:1161 std_initialize_port() bg0.ge010
    Blocking(4)
1998.04.02.10:59:35.205 60r NOTICE standard.c:741 std_make_forwarding() bg0.ge010
    Forwarding(3)
```

Figure 13-9. Output from bridging trace file

Bridge group information

brinfo returns configuration information about a bridge group and each of its member ports. The number of ports in a group is stated in the `Ports:` line. Enter:

```
# brinfo bridge_group

# brinfo bg0
Bridge Daemon: Running
Bridge_group: bg0
     Flags: (0x43) up broadcast running
     Ports:  2
     port ge003
          State: (0xf):Forwarding
          Flags: 0x9143 up broadcast running promisc link0 multicast
          Bridging media: ethernet bpdu
          MAC Address:  0:c0:80:00:55:d1

     port gf010
          State: (0xf):Forwarding
          Flags: 0x9143 up broadcast running promisc link0 multicast
          Bridging media: fddi bpdu
          MAC Address:  0:c0:80:00:55:d2
```

Low-level state information

brstat obtains low-level state information from **bridged**. Enter:

```
# brstat


    Bridged Information:
       Debug Level: 5, Trace Mask: 0xffffffff
       Log File: "/var/tmp/bridged.trace", Config File: "/etc/bridged.conf"
       bridged started at: Fri Jan 9 14:39:58 1998

    Bridge Group bg12
       Spanning Tree:   Enabled
       Root Bridge:     32768 00:c0:80:0c:65:53
       Bridge ID:       32768 00:c0:80:83:43:f9

       Root Port: ge066, Root Path Cost: 10
       Topology Change Detected:  No
       Root   Max Age: 20, Hello Time:  2, Forward Delay: 15
       Bridge Max Age: 20, Hello Time:  2, Forward Delay: 15,  Hold Time: 1

                                 Path  Desig  Desig              Desig
    Interface Port ID Con State     Cost  Cost   Bridge             Port
    --------  ------  --- -------- ----- -----   ------------------------
    gf080    128 1    No  Disabled   10
    gf081    128 2    No  Disabled   10
    gf082    128 3    No  Disabled   10
    gf082    128 4    No  Disabled   10
    ge065    128 5   Yes Blocking   10   0   32768 00:c0:80:0c:65:53  128 5
    *ge066   128 6   Yes Listening  10   0   32768 00:c0:80:0c:65:53  128 6

Dump snapshot finished at Fri Oct 9 14:40:01 1998
```

Route trees and filtering table

The **netstat -rn** command returns the bridging filtering table of MAC addresses and other related information. Be careful with this command if your route table has a large number of entries. Use **netstat -rn | wc -l** to first check the number of routes in the route table.

Since the filtering table itself tends to be lengthy, pipe the **netstat** command with **more** to view it easily. Enter:

```
#  netstat -rn

Routing tables

Internet:
Destination      Gateway          Flags     Refs      Use  Interface
198.174.11       198.174.11.33    U           22    21575  ef0
198.174.11.33    127.0.0.1        UGH          2      765  ef0
204.221.156      204.221.156.33   U            1        5  gh010

Bridging:
Destination      Gateway            Flags  Refs   Age     Interface
00:c0:f2:00:1e:a0  00:00:00:00:00:00 UHD     0     1:02     gf081
          •                •          •      •       •        •
          •                •          •      •       •        •
          •                •          •      •       •        •
          •                •          •      •       •        •
          •                •          •      •       •        •
          •                •          •      •       •        •
```

Source MAC address *Port interface name*

```
Bridging ARP:
Destination      Gateway            Flags    Refs    Use  Interface
198.174.59.101 00:03:01:80:62:82   UHD     0      0            bg0
#
```

Note: The age displayed for bridging routes is the time since the address was first learned. Age is expressed in the following format:

:15	seconds
1:27	1 minute, 27 seconds
3:14:02	3 hours, 14 minutes, 2 seconds
4d	4 days

Bridging sockets

The command **netstat -f bridge** displays all active bridging sockets. Sockets are used by **bridged** to transmit and receive Bridge Protocol Data Units:

```
$ netstat -f bridge
Active bridging sockets
Proto Recv-Q Send-Q  Group                  Port                (flags)
bridg      0      0  bg0                     *                   3
```

Kernel bridging statistics

The command **netstat -s -f bridge** displays kernel statistics for bridging:

```
$ netstat -s -f bridge
  bridging:
    37 packets received
    0 packets received before bridging configured
    0 packets received for unknown interface(s)
    0 packets dropped for pullup failures
    0 packets dropped for socket full
    0 packets dropped for no endpoint
    37 packets delivered
    0 packets dropped for no memory
    147052 packets sent
    169701 output packets dropped for interface down
    0 output packets dropped for link down
    44119 output packets unicast
    102933 output packets multicast
    0 output packets dropped for no memory
    9536 output packets copied
    93397 output packet copy avoided
    1080 copied output packets dropped for no memory
    0 output packets with too many copies
    0 learning/forgetting messages processed
    0 learning/forgetting messages dropped for pullup failures
    0 learning/forgetting messages for unknown interface(s)
    0 learning/forgetting messages with unknown opcodes
    0 learning messages processed
    0 forgetting messages processed
    0 learning messages failed
    0 forgetting messages failed
    0 forced routing table push(es)
```

Examining and debugging bridge configurations

Introduction

There are several places to start debugging bridging problems. To begin, it is probably most useful to try to determine which piece of the system seems to have the problem.

Three pieces of software need to work together for bridging to work correctly:

– **bridged** software (user space)

– GRF kernel software

– media card software

This section describes how tools such as **brinfo** and **brstat** can be helpful in debugging **bridged** and the GRF kernel software. If a problem cannot be isolated using these tools, or if the tools indicate the problem to be elsewhere, then debugging on the media card side needs to be pursued.

This section also describes how to gather traces from **bridged** to help diagnose possible problems. In specific bridging configurations, traces can help to understand **bridged** behavior based on the IEEE 802.1D standard.

Before attempting to debug bridging software, it is helpful to have read the IEEE 802.1D standard, especially those sections describing the behavior of the spanning tree. This bridging implementation uses the spanning tree functionality described in that standard.

Information needed by Ascend support

When you need to send Ascend support information about bridging problems, please include the following:

– a complete description of the problem

– **brinfo** output

– **brstat** output

– a description of the network (text or picture)

– **bridged** trace file(s)

– contents of `/etc/bridged.conf`

– contents of `/etc/gratm.conf`

– contents of `/etc/grifconfig.conf`

Enabling traces via bridged command

bridged writes traces to the `/var/tmp/bridged.trace` file.

This file is periodically archived and saved off in a compressed form to files in the form: `/var/tmp/bridged.trace.x.gz` in which `x` is 0–5.

By default, **bridged** runs with minimal tracing enabled. This saves the system overhead of writing every trace entry and the disk space used by the log file.

Sometimes it is necessary to gather additional **bridged** trace information for a given problem. When this needs to be done, edit `/etc/bridged.conf` using **bredit**, and change the **debug_level** line to read:

```
debug_level 6 ;
```

When this change is committed and **bridged** is reconfigured, additional traces are written to the **bridged** trace file. Once the error condition has been recreated, save the traces and then change the traces level back to 5.

The debug levels correspond to the following list:

- Level 0 (LOG_EMERG): Unusable
- Level 1 (LOG_ALERT): Action must be taken immediately
- Level 2 (LOG_CRIT): Critical conditions
- Level 3 (LOG_ERR): Error conditions
- Level 4 (LOG_WARNING): Warning conditions
- Level 5 (LOG_NOTICE): Normal but significant condition
- Level 6 (LOG_INFO): Informational (basic internal logic)
- Level 7 (LOG_DEBUG): Debugging (low-level internal logic)

Debug level 5, the default, provides more than enough information to resolve most **bridged** issues. Levels 6 and 7 are rarely used.

Note: Error conditions (fatal and non fatal) are always traced.

Displaying useful information

Two commands, **brinfo** and **brstat**, display a majority of the available bridging information.

brinfo queries the kernel to determine state and topology information about the current bridge groups and their operating environment.

brstat queries **bridged** for a superset of this and other information.

Using brinfo

In the example, two bridge groups are configured in the kernel: "`bg0`" and "`bg1`". While `bg1` has no interfaces defined, `bg0` has two interfaces defined, `gf080` and `gf081`.

Here is the **brinfo** output for the two groups:

```
Bridge group name: bg1
Flags:(0x43) up broadcast running
Ports : 0

Bridge group name:  bg0
Flags:(0x43) up broadcast running
Ports : 2

    Port gf080 : State (0) Blocking
    Flags : (0x9343) : up broadcast running promisc link0 multicast
    Bridging media: fddi bpdu
    MAC address: 0:c0:80:0:55:d1

    Port gf081 : State (0X) Forwarding
    Flags : (0x9343) : up broadcast running promisc link0 multicast
    Bridging media: fddi bpdu
    MAC address: 0:c0:80:0:55:d3
```

Each interface is in one of the following states:

- **disabled**, usually by configuration, or if there is no connection on this port
- **blocking**, by spanning tree logic
- **listening**, spanning tree intermediate state
- **learning**, spanning tree intermediate state
- **forwarding**, spanning tree stable state

The flags correspond to the flags seen when the **ifconfig *interface*** command is used. Flags tell us about the state of the interface from the kernel's perspective. From a bridging perspective, the flags shown in the example are the flags that should be set for normal bridge operations.

All flags should get set automatically when **bridged** starts and interfaces are configured in the **bridged** configuration file.

If the `link0` flag is not set, as in:

```
Flags : (0x9343) : up broadcast running promisc multicast
```

then there is no connection at this interface. Either no wire is connected to the interface, or no host is on the other end of the wire.

State information - brstat

The **brstat** command signals **bridged** to dump out its internal state into the file `/var/tmp/bridged.dump`. This file is massaged by **brstat** to display information of interest. See the **bridged** man page for details about the debug level, log file, and configuration files.

Here is an example of **brstat** output:

```
# brstat


Bridged Information:
   Debug Level: 5, Trace Mask: 0xffffffff
   Log File: "/var/tmp/bridged.trace", Config File: "/etc/bridged.conf"
   bridged started at: Thu Apr 27 18:43:12 1997


Bridge Group bg0
   Spanning Tree:   Enabled
   Root Bridge:     7 08:00:2b:b6:38:80
   Bridge ID:       27 00:c0:80:00:55:d1

   Root Port: gf081, Root Path Cost: 10
   Topology Change Detected: No
   Root   Max Age: 20, Hello Time:  2, Forward Delay: 15
   Bridge Max Age: 20, Hello Time:  2, Forward Delay: 15, Hold Time: 1

                                 Path  Desig Desig                  Desig
Interface Port ID Con State      Cost  Cost  Bridge                 Port
--------- ------- --- ---------- ----- ----- ---------------------- -------
gf080     128 1   Yes Disabled   10    0     32768 08:00:2b:b6:38:80 128 4
*gf081    138 2   Yes Forwarding 10    0     32768 08:00:2b:b6:38:80 128 6


Dump snapshot finished at Fri Apr 28 15:20:00 1997
```

The configuration information starts at the Bridge Group section. The Designated Root line shows the MAC address of the root bridge.

In the example above, the root bridge is transmitting BPDUs with a priority of 7. The GRF (`bridge ID 00:c0:80:00:55:d1`) is transmitting BPDUs at a priority of 27. If the priorities were equal, the MAC address would be used to determine the root bridge.

The MAC address of the GRF bridge is selected as the numerically lowest MAC address of all the ports in the bridge group, or the MAC address of the maintenance Ethernet port.

The Root Path Cost and other values displayed in the second set of descriptors are spanning tree values that describe spanning tree configuration variables. See the IEEE 802.1D standard for more information about these variables.

The example also shows two configured interfaces, `gf080` and `gf081`. Each interface displays a priority, a unique id, a state, a status, and spanning tree variables associated with the 802.1D standard. Port priority is used to set up redundant bridge connections to the same LAN. An asterisk (*) indicates the root port

Collect data via grdinfo

With a single command, **grdinfo** collects the output from **brinfo -all** and statistics from **brstat**, and compresses it in a log file. Refer to the “Management Commands and Tools” chapter in this manual for more information.

MAC addresses and bridge IDs via netstat -in

Use the **netstat -in** command to check which interfaces have bridge IDs. An excerpt from **netstat** output is shown below. Bridge IDs are listed as **`<Bridge>`** in the network column:

```
# netstat -in
Name    Mtu   Network     Address            Ipkts  Ierrs   Opkts Oerrs  Coll
de0     1500  <link1>     00:c0:80:0c:65:53    58217      0   231408 0    287
de0     1500  206.146.164 206.146.164.9        58217      0   231408 0    287
rmb0    616   <link2>     00:00:00:00:00:00  3177659 15624  3278448 0      0
rmb0    616   <GRIT>      0:0x40:0           3177659 15624  3278448 0      0
lo0     1536  <link3>                          61134      0    61134 0      0
lo0     1536  127         127.0.0.1            61134      0    61134 0      0
lo0     1536  <GRIT>      0:0x48:0             61134      0    61134 0      0
atmp0*  1536  <link4>                              0      0        0 0      0
bg0     1500  <link147>                        22410      0        0 0      0
bg0     1500  <Bridge>    00:c0:80:0c:65:53    22410      0        0 0      0
bg0     1500  222.222.169 222.222.169.9        22410      0        0 0      0
bg0     1500  47.0000.8000.0900.0900.0900      22410      0        0 0      0
gl000*  1496  <link5>                              0      0        0 0      0
gf000*  4352  <link142>   00:c0:80:89:24:0e        0      0        0 0      0
gf001*  4352  <link137>   00:c0:80:89:24:0f        0      0        0 0      0
gf002*  4352  <link136>   00:c0:80:89:24:10        5      0      116 0      0
gf002*  4352  <Bridge>    00:c0:80:89:24:10        5      0      116 0      0
gf003*  4352  <link140>   00:c0:80:89:24:11        0      0        0 0      0
ge020   1500  <link141>   00:c0:80:89:09:9d        0      0    35831 0      0
ge020   1500  <Bridge>    00:c0:80:89:09:9d        0      0    35831 0      0
ge021   1500  <link130>   00:c0:80:89:09:9e     1029      0    35763 0      0
ge021   1500  <Bridge>    00:c0:80:89:09:9e     1029      0    35763 0      0
```

The Address field provides the MAC address. Note that the bridge port and the associated link for that interface have the same address.

Restarting bridged during debug

The **brsig** command provides a way to signal **bridged**. **brsig** takes the following parameters:

0 (zero)
USR1
USR2
HUP

This command is not needed for normal operations. It is a low-level debug tool, and should be used carefully. When a signal is received by **bridged**, you see a message similar to this:

```
# bridged signalled successfully.
```

Use **brsig 0** to verify that **bridged** is running.

Use **brsig USR1** to cause **bridged** to write a dump file, `/var/tmp/bridged.dump`. This dump contains detailed information about the state of internal timers and bridging configuration. **bridged** rewrites the dump file each time it receives **USR1**.

Use **brsig USR2** to cause **bridged** to check the state of a bridging interface from the kernel's perspective.

Use **brsig HUP** to cause **bridged** to reread the `bridged.conf` configuration file or an alternate file specified in the **bridged** command line (usually reserved for debugging purposes). The **bredit** command sends a **HUP** to **bridged**.

IP Packet Filtering

14

Chapter 14 describes how to configure IP packet filters for GRF media interfaces and how to use the set of filtering **maint** commands to display statistics and configuration data.

Chapter 14 covers these topics:

GRF filtering implementation

The GRF can perform straightforward IP packet filtering on any configured logical interface.

Filtering can be done on:

- source IP addresses
- destination IP addresses
- TCP, UDP, and ICMP addresses
- additionally, UDP addresses can be filtered against a destination port
- TCP addresses can also be filtered against a destination port or an established session

Two components comprise the filtering capability:

- filters, comprised of rules
- bindings

Filtering can be configured on all GRF media cards except the ATM OC-12c card, these are: HIPPI, FDDI, ATM OC-3c, HSSI, Ethernet, and SONET. The **filterd** man page does not list Ethernet and SONET support even though it is provided.

Configuration daemon

Filtering is done at the card level. The **filterd** daemon controls filtering but does not directly process IP packets. Messages from **filterd** are sent to the `gr.console` log.

filterd acts as an information mechanism between the kernel and the media card. It handles configuration and status requests from the cards.

Refer to the **filterd** man page for additional information about internal filter components and implementation.

Configuration file

The template for the filtering configuration file resides in `/etc/filterd.conf.template.`

Copy this file to `/etc/filterd.conf` and edit it to define the filtering process. A copy of the file template is included in this chapter and in the *GRF Reference Guide*. Refer to the `filterd.conf` man page for additional information about internal filter components and implementation.

CLI access to /etc/filterd.conf

The **gredit** command accesses the `/etc/filterd.conf` file from the command-line interface (CLI) and opens the file in the **vi** editor.

Enter:

```
super> gredit filterd
```

maint command set

A set of GRF **maint** commands are available to monitor and retrieve filtering information.

The **maint** commands for filtering on the receive side of a card are **maint 50** through **maint 58**, and keep the same set of command numbers across each type of media card. For all except FDDI media cards, the **maint** commands for filtering on the transmit side of a card are **maint 150** through **maint 158**. FDDI transmit **maint** commands range from **maint 70 50** through **maint 70 58**. **maint** commands are described at the end of this chapter.

Creating a filter

Each filter is given a unique name. Filter names are descriptive, usually less than 64 characters.

A filter's ability to discard packets is defined by a rule or rules.

Here is the beginning of an example of a filter to be used for a site's mail server:

```
filter mail_server_allow {
     implicit deny;
```

The `implicit deny` statement enables the `mail_server_allow` filter to discard those packets which do not match the rules, i.e., which are not in a specified set of addresses. The default case is `deny`.

In this example, a `permit` statement accepts those packets that do not match the rules:

```
filter mail_server_allow {
     implicit permit;
```

The `implicit permit` statement enables a filter to accept those packets that do not match the rule(s), i.e., that are not in a specified set of addresses.

Rules

A rule specifies a match against a packet. If a rule matches, the filter either permits or denies the packet based on the rule prefix “`permit`” or “`deny`”.

Rules should be carefully ordered. The first rule matched by a packet governs the way in which the packet is treated. Correct ordering maximizes the filter’s efficiency and minimizes the effects filtering enacts upon packet throughput rates.

Here is an example of a permit rule to govern access to the mail server at port 25:

```
permit {
     to 222.222.222.1 0.0.0.0;
     ipv4protocol tcp {
          port 25;
     }
}
```

Applying a mask

Note that `222.222.222.1` is an IP address, and `0.0.0.0` is a mask that applies to the IP address.

Where there are 0s in the mask, permitted packets must have an IP address that exactly matches the given IP address. In the example, a bitwise mask of `0.0.0.0` only permits packets bearing a single IP address, that of `222.222.222.1`. This is because four 0s require an exact match in all four sections of the address.

This example permits only those packets bearing IP addresses `222.222.222.1`:

```
filter mail_server_allow {
     implicit deny;
     permit {
          to 222.222.222.1 0.0.0.0;
          ipv4protocol tcp {
               port 25;
          }
     }
```

Non-zero values in a mask indicate a specific match is not required in that section of the IP address. This is informally known as a “don’t care” value because non-zero values mark those bits that you don’t care about matching.

Using the example, a mask of `0.0.0.255` on address `222.222.222.1` permits packets bearing IP addresses `222.222.222.0` through `222.222.222.255`. .
In effect, that mask permits all packets from the Class C network `222.222.222`.

Similarly, the mask `0.0.0.3` applied to `222.222.222.0` permits only those packets bearing IP addresses `222.222.222.1` and `222.222.222.2`.

This filter permits only those packets bearing IP addresses:

```
222.222.222.0
222.222.222.1
222.222.222.2
222.222.222.3
```

```
filter mail_server_allow {
     implicit deny;
     permit {
          to 222.222.222.0 0.0.0.3;
          ipv4protocol tcp {
               port 25;
          }
     }
```

Applying a filter

The same filter can be applied to logical interfaces on one or several media cards, the cards can be the same or different media. A filter is applied using a bind statement, binding is discussed in this chapter.

Refer to the `/etc/filterd.conf` man page for additional information about internal filter components and implementation.

Filters for service ports

Ports are an IP convention to describe a logical entity at which a network or RMS host service resides. The function of a port is synonymous with the UNIX socket convention.

In the filtering example used here, port 25 is the logical “place” where the `sendmail` service resides. TCP/IP has a list of standard ports for network and host services, refer to the `/etc/services` file for a list of GRF ports and services.

A port number can be used in a filter even if the port is not included in the `/etc/services` file.

Specifying port numbers

As you define a filter in `/etc/filterd.conf` that acts upon a port, you can specify ports in two ways such that

1,2,3,10 is the same as 1..3,10

Also, you can specify ranges of port numbers using:

`Gt`	to mean greater than
`Lt`	to mean less than
`Le`	to mean less than or equal to
`Ge`	to mean greater than or equal to

To specify services at ports numbered higher than 86, and to include port 86,
the port statement is:

```
port Ge 86;
```

To specify services at ports numbered lower than 24, but not including port 24, the port statement is:

```
port Lt 24;
```

Bindings

Once the filters are defined, a binding is used to apply a filter to the intended logical interface or interfaces.

Binding statements identify:

- the target media card using the card's media type and chassis slot number
- the target logical interface(s) by the interface's number
- direction, inbound and/or outbound, of packets to be examined
- action(s) to be performed on the selected packets

This binding example shows how the media card in slot 3 is specified, and how the mail server filter is applied to logical interface 0 to which the mail server connects:

```
media fddi 3 {
     # filter
     bind mail_server_allow {
          vlif 0;           # DAS interface
          direction out;    # outbound traffic
          action filter;
     }
}
```

Logical interface number (vlif)

The logical interface is identified by the virtual logical interface number statement, `vlif <number>`, in which `number` is expressed in decimal.

The number of logical interfaces varies among media cards:

ATM OC-3c

- `vlif <number>` ranges between 0 and 255, as in `vlif 0..18`, or `vlif 1,20,23..25`, or `vlif 222`.

HIPPI, SONET

- `vlif <number>` is always `vlif 0` since a HIPPI card has a single logical interface.

HSSI

- `vlif <number>` ranges between 0 and 255 when the framing protocol is Frame Relay, and between 0 and 1 when PPP or HDLC are running.

FDDI

- `vlif <number>` ranges between 0 and 3, depending upon the combination of SAS and DAS connections.

Ethernet

- `vlif <number>` ranges between 0 and 7 for eight ports.

In the `vlif <number>` statement, you can specify all the logical interfaces on a particular media card even if some interfaces are not configured or defined with an IP address. This is a way to apply a standard filter across system interfaces, or to ensure automatic future compliance with a filtering requirement.

Direction

Traffic direction is specified to be `in`, `out`, or `into_me`.

Direction `in` places a filter at the inbound logical interface. This option filters packets coming from a source external to the GRF that are on their way to the switch or the RMS.

Direction `out` places a filter at the outbound logical interface. This option filters packets coming from the switch or the RMS before they are transmitted out to a destination external to the GRF.

Direction `into_me` places an internal filter for the router manager system (RMS). This option supports a filter that affects performance less since the filter is not active in a data path.

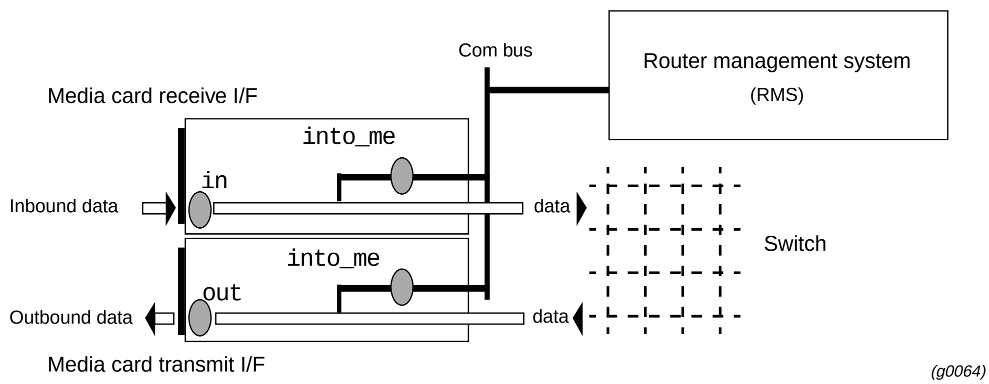


Figure 14-1. Placing filter with the direction option

More than one direction can be specified in a single binding:

```
media fddi 3 {
     # filter
     bind mail_server_allow {
          vlif 0;           # DAS interface
          direction out;    # outbound traffic
          direction in;     # inbound traffic
          action filter;
     }
}
```

Media type

Use these names to specify media type:

```
hippi
fddi
atm     (OC-3c)
hssi
ether
sonet   (OC-3c)
```

Actions

This statement describes the activity initiated by the filter.

Currently available actions include:

- `filter`, for dropping a "matched" packet
- `log`, for logging a packet header or portion of a packet
- `class` *`class_value`*, used with Controlled-Load to mark packets so they are not dropped by SPD, refer to the *Integrated Services* chapter

Filtering states

Operational

In the `Operational` state, a filter is loaded and ready to function.

Fastop

In the `Fastop` state, the filter is loaded and ready to function. In addition, because it is a single filter applied to a given interface and direction, this filter makes use of a faster algorithm.

Dependent

In the `Dependent` state, the filter is waiting for a filter code or action information. `Dependent` is a transitional state, and if the state is seen to persist, a fault has occurred.

Pend Delet

In the `Pend Delet` state, the filter is in the process of being removed, but is otherwise operational and continues to function. When the change/update completes, the filter is removed from the system. `Pend Delet` is a transitional state, and if the state is seen to persist, a fault has occurred.

Packet header logging

Logging is a recording activity in which a copy of the header on the filtered packet is sent to a logging host. Logging allows you to monitor a packets sent to a specified address, and can be used to monitor intrusions to a specific address.

When the filtering action is `log`, then a packet that matches the filter is detected, its header (or a specified portion) is logged, but the packet is not dropped.

You can log up to 512 bytes of information at a UDP-named port on the target host. Remember that the rate at which packets are examined/logged readily affects performance.

This is the format for the logging configuration:

```
action log {
      target ip_address port;
      rate n;          # opt to change, default = 1
      buffer ;
      size n;          # opt to change, default = 20 bytes
}
```

`target` *`ip_address`* -
- the IP address of the target host you want to send logged information to.

`port`
- the UDP port on the target host that gets the information.

`rate` *`n`*
- the rate at which packets are examined/logged, when *n* is 20, every 20th packet is logged. Default = 1, all packets logged.

`buffer`
- a keyword, if you include `buffer` in the list, logged packets are buffered.

`size` *`n`*
- specifies the number of bytes to be taken from each logged packet, the default for *n* is 20 bytes, the basic size of an IP header

Note: If you require that a type of detected packet be both logged and dropped, a configuration example is provided in the "Logging and dropping packets" section later in this chapter.

Logging to the administrative Ethernet

Packet header logging will not forward packets to the IP address of the administrative Ethernet (`de0` or `ef0`), or to an IP address out that interface.

To log packet headers to the RMS itself, use the IP address of the interface on which the filter is configured.

Logging loops

A logging loop is created when a logged packet causes multiple logged packets to be generated, and the card processing activity is taken up with cycles of handling logging packets.

Filters on the receive side

Logging filters assigned on the downstream (transmit) side may create logging loops. There is no risk of such loops if logging filters operate on the receive or upstream side.

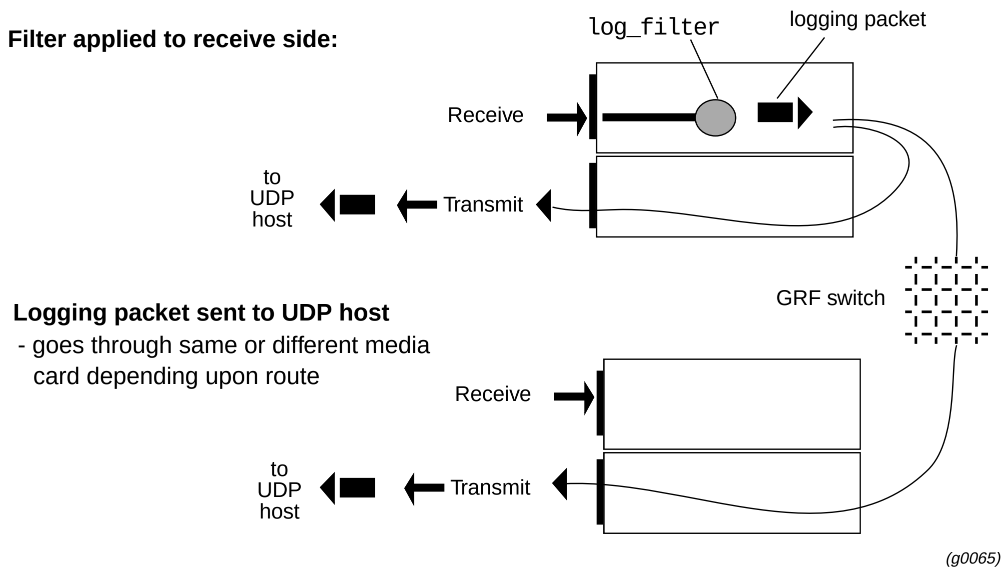


Figure 14-2. Receive-side logging filters do not loop

The logging packets are sent to the appropriate transmit data path.

Filters on the transmit side

With a filter applied to the transmit or outbound port, a packet marked for logging causes the transmit side to request that the receive side send the logging packet to the UDP host.

If the route to that host is through the transmit port, markers in the newly-created logging packet can cause another logging request to the receive side. Media card activity can become bogged down in this cycle of logging and requesting.

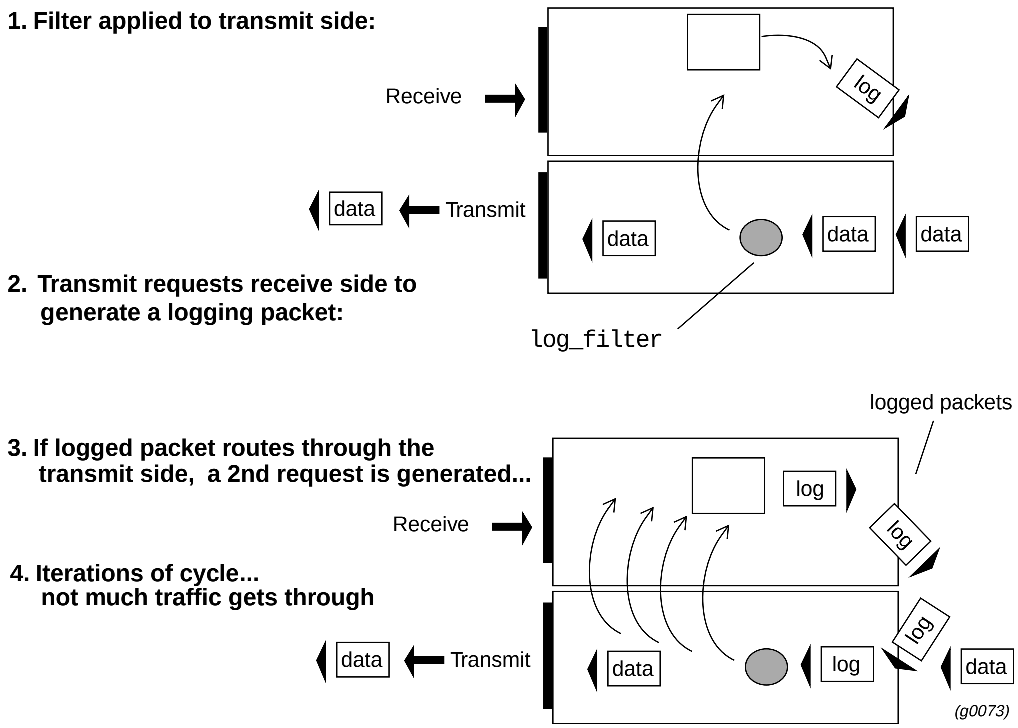


Figure 14-3. How logging loops can occur

Loops caused by ICMP messages

In a specific situation, a packet header logging filter can cause loops in which ICMP port unreachable messages are generated unnecessarily.

If a packet header logging filter is configured on the receive side of a media card to log ICMP messages, and the filter is also configured to log packet headers to a system out the same media card, a loop can occur if there is no application listening on the UDP port configured to receive the logged packet header.

This loop occurs because the logging system must send back an ICMP port unreachable message to the GRF when there is no application running. This ICMP message will be selected by the filter and cause another attempt to log a packet header, which will in turn cause another ICMP port unreachable message to be sent to the filter on the GRF.

Avoid this situation by configuring the logging filter to ignore ICMP messages from the logging host to the media card's configured IP addresses.

Controlling access to the internal system

This section describes blocking IP traffic into the internal router management system software, the RMS. It does not refer to the administrative Ethernet LAN access (`de0` or `ef0`, at the EXT connector).

To prevent unauthorized access to the internal system, the following information is critical to the way you set up an RMS block.

In a normal case, an administrator would deny access to the RMS over a given media card interface. The following filter and binding (using the `into_me` direction) prevent access to the RMS across the FDDI card in slot 1:

```
# For fddi, our RMS interface is 222.222.41.4
filter block rms {
     implicit permit;

     deny {
          to 222.222.41.4 0.0.0.0;
     }
}

media fddi 1 {
     bind block rms {
          vlif 0;
          direction into_me;
          action filter;
     }
}
```

This method only works when there is a single interface card, an unlikely configuration for a router.

Assume there also is an ATM OC-3c card with the address of 222.221.41.4. For this example, the ATM is a secure interface, and does not have an RMS block on it. (However, this example applies even if there is an RMS block on the ATM card.)

With the `block_rms` filter in place, a hostile user's attempt to come *in* over the external FDDI link to 222.222.41.4 is blocked. However, if the attack is on the ATM card, 222.221.41.4, in over the internal link to the FDDI interface, the packets fail to match the deny and are passed on. Since the destination is the RMS, the FDDI card passes the traffic directly to the RMS, thus allowing access.

To prevent the indirect access, any filter that blocks access to the RMS by using the destination address, must list *all* GRF interfaces.

The next example shows how the `deny` list expands to include the other installed interfaces:

```
# For fddi, our RMS interface is 222.222.41.4,
# ATM is 222.221.41.4, and
# the RMS administrative ethernet is 222.220.41.4.

filter block rms {
     implicit permit;
```

```
          deny {
               to 222.222.41.4 0.0.0.0;
               to 222.221.41.4 0.0.0.0;
               to 222.220.41.4 0.0.0.0;
               to 192.168.2.1 0.0.0.0;
          }
     }

     media fddi 1 {
          bind block rms {
               vlif 0;
               direction out;
               action log {
                    target 192.168.2.1 100;
          }
     }
```

Note that the same filter can also be bound to the ATM interface to secure it.

Filtering configuration process

When the filtering plan is complete, these are the step to create and start up GRF filtering:

1 Copy the `/etc/filterd.conf.template` file into `/etc/filterd.conf`

2 Edit `/etc/filterd.conf` to create and assign each filter:

– create the filter(s) and specify rule(s)

– specify the binding(s)

The template for the `/etc/filterd.conf` configuration file appears on the next several pages and is also included in the *GRF Reference Guide*.

3 Start the **filterd** agent

Either send a HUP signal:

```
% kill -HUP ‘cat /tmp/filterd.pid‘
```

or reset the GRF:

```
% shutdown r now
```

Verify that filters are applied to the correct media cards and logical interfaces using the **maint** commands.

Changing filters on-the-fly

You can update and change filters on the fly. The system or media cards do not need to be reset to put changes into effect.

Sample filters

The following examples demonstrate several filter applications.

Filtering against ping

This example configures a filter that prevents GRF 3 from pinging the `ge031` interface in GRF 1, but allows GRF 3 to ping any other interface shown.

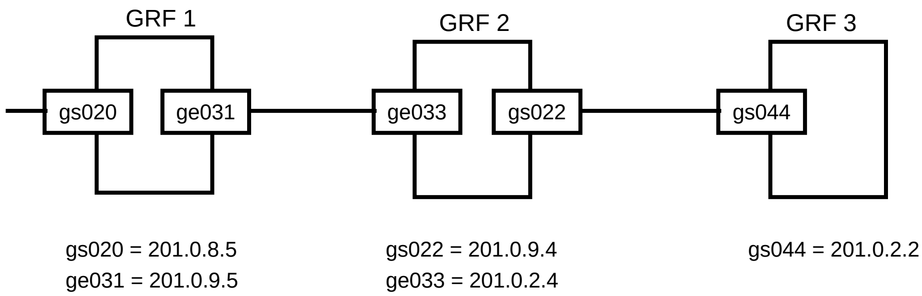


Figure 14-4. Example: filtering against ping

The filter is configured and applied in GRF 2:

```
filter stopicmp {
    implicit permit;
    deny {
    to 201.0.9.5 0.0.0.0;
    ipv4protocol icmp;
    }
}

media hssi 2 {
     bind stopicmp {
          vlif 0;
          direction in;
          action filter;
     }
}
```

After you define the filter in `/etc/filterd.conf`, kill and restart the daemon using the **kill -HUP** command.

With the filter in place in GRF2, ICMP packets from GRF3 to 201.0.9.5 are blocked:

```
# ping 201.0.9.5
    ICMP ECHO 201.0.9.5 (201.0.9.5):  56 data bytes
    --- 201.0.9.5 ICMP ECHO statistics ---
    4 packets transmitted, 0 packets received, 100% packet loss
#
```

But ICMP packets from GRF3 to other destination IP addresses are forwarded:

```
# ping 201.0.8.5
     ICMP ECHO 201.0.8.5 (201.0.8.5):  56 data bytes
```

```
     64 bytes from 201.0.8.5: icmp_seq=0 ttl=254 time=1.007 ms
     64 bytes from 201.0.8.5: icmp_seq=1 ttl=254 time=0.887 ms
     --- 201.0.8.5 ICMP ECHO statistics ---
     2 packets transmitted, 2 packets received, 0% packet loss
        round-trip min/avg/max = 0.887/0.940/1.007 ms
#
```

Filter against network spoofing

This filter prevents the network 192.100.100 from being spoofed by preventing packets coming to address 192.100.100.0 (0.0.0.255) from reaching the network.

```
filter spoof  {                     # Declare the filter spoof
     implicit permit;               # Allow for everything

     deny {                         # Prevent network from being spoofed
          from 192.100.100.0  0.0.0.255;
          to 192.100.100.0  0.0.0.255;    # not to network 192.100.100.0
     }
}

filter tcp_test {                        # Declare the filter tcp_test
     implicit permit;                    # Allow for everything

     deny {         # Deny ntwks 198.1.x.x from ftping to 198.100.100.1
           from 198.1.1.0  0.0.0.255;      # Network 1
           from 198.1.2.0  0.0.0.255;      # Network 2
           from 198.1.3.1  0.0.0.0;        # Network 3
           to 198.100.100.1  0.0.0.255;    # Ftp server
     }

      ipv4protocol tcp {      # Define protocol
            port 21 ;       # Specify ftp port
     }
}

media atm 3 {                              # Define the media card type
       bind tcp_test {         # Bind tcp_test to atm logical interface
             vlif 0..8;       # First logical interfaces on phys port 0
            direction in;         # In bound packets
            action filter;
       }
 }
```

Provide services for an intranet

The purpose of this filter is to allow certain TCP/IP services to reach specified workstations on a FDDI-based intranet. The `gf020` FDDI card serves as a gateway to this intranet.

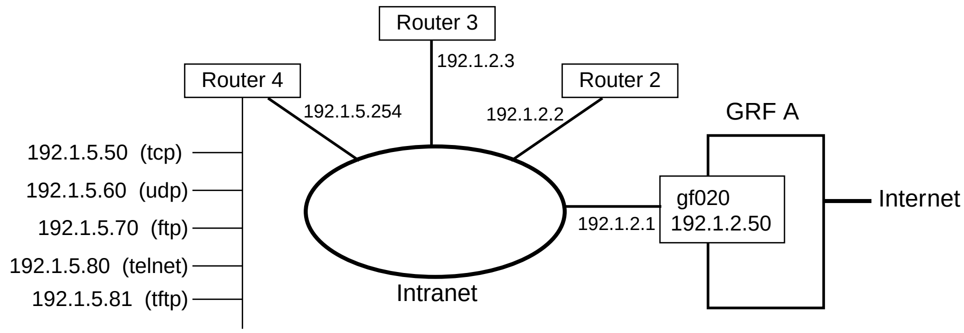


Figure 14-5. Example: controlling services in an intranet

1 Create the filter "intranet" and assign an implicit deny:

```
filter intranet {
     implicit deny;
```

2 Then allow access to a specific host for each service:

```
     permit {
          to 192.1.5.50  0.0.0.0;
          ipv4protocol  tcp {
               port  gt 1024;              # All non defined ports
               ESTABLISHED;                # Existing connections
          }
     }

     permit {
          to 192.1.5.60  0.0.0.0;
          ipv4protocol  udp {              # Domain Server access
               port 53;
          }
     }

     permit {
          to 192.1.5.70  0.0.0.0;
          pv4protocol tcp {                # FTP Server access
              port 21;
          }
     }

     permit {
          to 192.1.5.80  0.0.0.0;
```

```
          ipv4protocol tcp {                # Telnet access
               port 23;
          }
     }

     permit {
          to 192.1.5.81  0.0.0.0;
          ipv4protocol  udp {               # TFTP Server access
               port 69;
          }
     }
```

3 Now bind the filter “`intranet`” to all four logical interfaces on the FDDI card in slot 2:

```
media fddi 2 {
     bind intranet {
          vlif 0..3 ;
          direction in;
          action filter;
     }
}
```

Filtering configuration file – filterd.conf

This is the `/etc/filterd.conf` file template. Refer to the `filterd.conf` man page for other examples and related information.

```
#
# Template file to give an example of how filtering is setup.
#
#

# The first step is to define filters that you want to have.  These are
# general items that can be applied to more than one media card/interface or
# don't have to be applied at all.  If you are not using a filter,
# however, comment it out using /* filter.... */ to allow for faster
# load time.
#
#
# The following is a fairly complex filter example that denies access to the
# world but lets in some "things".  A definition of each of these "things" is
# included in a comment by each rule.
#
# In GR filtering lingo, the following is a "filter":
#

/* ------
filter example_in_1 {
        implicit deny;          # if no rules matched, filter DENIES

        #
        # Make our first match rule.  In this case allow packets from
        # ports > 1023 and returning packets in established TCP connections.
        #
        # In GR filtering lingo, this is a "rule".
        permit {
                ipv4protocol tcp {
                        established;
                        port gt 1023;
                }
                ipv4protocol udp {
                        port gt 1023;
                }
        }

        #
        # Ok, now lets do something to allow people to talk to our
```

```
        # mail server.
        #
        permit {
        }
}

#
#
# The following is a second filter, which we will put on transmit side of
# our "internal" network. The filter prevents someone from dropping packets
# from the outside that look like they come from the inside.
#
# Note that it may make more sense to place this on the upstream side
# of all the "unsafe" interfaces instead of on the downstream side.  This
# is especially true if there are more than one interface on the "safe"
# side that move traffic amongst themselves.  Putting the filter on the
# downstream side in this case causes safe traffic to take the performance
# hit of blocking traffic on the upstream side.
#

filter spoof_block {
        # By default, let everything through.  Our upstream side will catch
        # the stuff it wants to weed out.
        implicit permit;

        # our "internal" network is 222.222.222.*.
        deny {
                from 222.222.222.0 0.0.0.255;
                to 222.222.222.0 0.0.0.255;
        }
}

#
# And now, an unusual filter to show how some other things work, but
# would never actually be created.
#
filter weird_filter {
        implicit deny;

        permit {
                from 128.101.101.101 0.0.0.0;   # from this one
                from 128.101.101.102 0.0.0.0;   # OR from this one
                to 220.220.220.0 0.0.0.255;     # AND to this one
                ipv4protocol icmp;              # ICMPs allowed
                ipv4protocol tcp {
```

```
                        # note that allow port 0, even though it is not
                        # a valid port.  By including it, range checking
                        # becomes more efficient in many cases.
                        port 0..3, 10..20, gt 1023;
                }
                ipv4protocol udp;               # all udp through
        }
}

#
#
# Once all filters are declared, we can declare what is called a "binding"
# This means attaching the above filters to interfaces.
#

media fddi 11 {
        # ok, what filter?
        bind example_in_1 {
                vlif 0;                 # do the first interface
                direction in;           # inbound traffic only
                action filter;          # route/don't route
        }

        # and we have another interface that we are going to pretend
        # routes to our internal net.
        bind spoof_block {
                vlif 1;                 # the other interface (DAS)
                direction out;          # outbound traffic only
                action filter;
        }
}
#
# And another card for our _other_ "internal" network.  We attach the
# weird filter to it.
media atm 3 {
        bind weird_filter {
                vlif 0..255;            # all interfaces
                direction in;           # in
                direction out;          # AND outbound
                action filter;
        }
}
-----*/
```

Filter grammar reference

This section provides the grammar of filter components. More information at this level of detail is available in the `/etc/filterd.conf` manpage.

```
configuration   :       /* empty */
                |       configuration configurations
                ;

configurations  :       filtercomponent
                |       filterentry
                |       ';'
                ;

filtercomponent :       FILTER_COMPONENT identifier '{' filterbody '}'
                |       ';'
                ;

filterentry     :       FILTER identifier '{' filterentrybody '}'
                ;

filterentrybody :       /* empty */
                |       implicit_permission filterbody
                |       filterbody
                ;

implicit_permission :   IMPLICIT PERMIT ';'
                |       IMPLICIT DENY ';'
                ;

filterbody      :       /* empty */
                |       filterbodies filterbody
                |       filteruse filterbody
                ;

filteruse       :       USE identifier ';'
                ;

filterbodies    :       filter_permission '{' filterinnards '}'
                ;

filter_permission :     PERMIT
                |       DENY
                ;

filterinnards   :       /* empty */
                |       filterinnard filterinnards
                |       ';'
                ;


filterinnard    :       FROM IPV4ADDR IPV4ADDR ';'
                |       FROM ALL ';'
                |       TO IPV4ADDR IPV4ADDR ';'
                |       TO ALL ';'
                |       IPV4PROTOCOL filterprotos
```

```
                |       ';'
                ;

filterprotos    :       TCP '{' tcp_options '}'
                |       TCP ';'
                |       ICMP ';'
                |       UDP '{' udp_options '}'
                |       UDP ';'
                |       ALL ';'
                ;

tcp_options     :       /* empty -> implies 'all' */
                |       tcp_option tcp_options
                ;

tcp_option      :       PORT filterportset ';'
                |       ESTABLISHED ';'
                ;

udp_options     :       /* empty -> implies all */
                |       PORT filterportset ';' udp_options
                ;

filterportset   :       filterport
                |       filterportset ',' filterport
                ;

filterport      :       NUMBER
                |       RANGE
                |       filteroperator NUMBER
                ;

filteroperator  :       GT
                |       GE
                |       LE
                |       LT
                ;

identifier      :
                |       IDENTIFIER
                ;

mediaentry      :       MEDIA hw_type slot '{' mediabody '}'
                |       ';'
                ;

mediabody       :       BIND identifier '{' bindbody '}'
                |       ';'
                ;

bindbody        :       /* empty */
                |       bindinnard bindbody
                ;

bindinnard      :       VLIF vlif_set ';'
                |       DIRECTION direction_key ';'
                |       ACTION actions
```

```
                |       ';'
                ;

vlif_set        :       vlif
                |       vlif_set ',' vlif
                ;

vlif            :       NUMBER
                |       RANGE
                ;

actions         :       FILTER ';'
                |       ';'
                ;

hw_type         :       ATM
                |       FDDI
                |       HIPPI
                |       HSSI
                |       ETHER
                |       SONET
                ;

slot            :       NUMBER
                ;

direction_key   :       IN
                |       INBOUND
                |       OUT
                |       OUTBOUND
                ;
```

Using the filtering maint commands

The filtering **maint** commands display a range of filter information and statistics.

maint commands are executed at the GR> prompt.

Invoking the maint prompt

To switch to the **maint** prompt, use the **grrmb** command, enter:

```
# grrmb
```

The **maint** `GR` *n*> prompt appears. The number is the current port the **maint** command will act on, 66 is the number of the control board:

```
GR 66>
```

Change the prompt port to the media card you are working with. For example, if you are working with a card in slot 2, enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 2
GR 2>
```

To leave the **maint** prompt, enter **`quit`**.

Filtering command set

The filtering capability adds the set of **maint** commands 50–58 to each media card’s set.
A media card with two CPUs has two sets of filtering commands, one on the receive side, one on the transmit side.

- Use **maint** 50–58 commands for filters configured on the receive [RX] side of a media card, as in **maint 55**.
- Use **maint** commands 150–158 for filters configured on the transmit side [TX] of a media card, as in **maint 155**.
- **maint** commands for the FDDI media card preface the set of filtering commands with 70, as in **maint 70 55**.

Here is the list of filtering **maint** options:

```
[RX]  50:    Filtering filter list: [detail_level [ID]]
[RX]  51:    Filtering filter list: [detail_level [IF]]
[RX]  52:    Filtering action list: [detail_level [ID]]
[RX]  53:    Filtering action list: [detail_level [IF]]
[RX]  54:    Filtering binding list: [detail_level [ID]]
[RX]  55:    Filtering binding list: [detail_level [IF]]
[RX]  56:    Display filtering statistics: [IF#]
[RX]  57:    Reset filtering statistics: [IF#]
[RX]  58:    Show filter protocol statistics
[RX]          note, IF/ID may be '-1' to indicate all of the given item
[RX]          while detail level is 0|1|2.
```

- `detail_level` is an optional parameter that specifies how much information is returned, useful levels are 0 and 1.
- `IF` is an optional parameter that specifies the interface number.
- `ID` is an optional parameter that specifies the filter ID randomly assigned by **filterd**.
- Odd-numbered commands 51–55 return information based on filter ID.
- Even-numbered commands 50–54 return information based on interface number.

This series of commands shows how media cards in slots 11, 6, and 5 are checked for filters assigned to the receive side:

```
# grrmb
GR 66> port 11
Current port card is 11
GR 11> maint 50 ge00b3
GR 11> [RX]
[RX]
[RX] No filters found.

GR 11> port 6
Current port card is 06
GR 06> maint 50
GR 06> [RX]
[RX]
[RX] No filters found.

GR 06> port 5
Current port card is 05
GR 05> maint 50 gf052
GR 05> [RX]
[RX]   filterID     type   status  access
[RX]   00000022   ctable (loaded)    0002
GR 05>
```

The card in slot 5 has filter 00000022 applied.

Translating filterIDs to actual names

Use the **grfutil -f** command to translate the **maint**-assigned filterID number you see in a **maint** display into the filter name you created:

```
# grfutil -f <filterID_number>
```

You must exit the **grrmb** mode to use **grfutil**:

```
GR 05> maint gf052
GR 05> [RX]
[RX]  filterID     type   status  access
[RX] 00000022   ctable (loaded)    0002
[RX]
GR 05> exit
super> sh
#
# grfutil -f 22
  filterID  temp  BPF SPARC SSPARC C30 CTABLE fname
  00000022    No  Yes    No     No  No    Yes mail_server_allow
```

Use **maint 54** to find out which interfaces have the filter applied:

```
# grrmb
GR 05> maint 54 22
GR 05> [RX]
[RX] vlif   BindID       state location  filterID  action_cnt
[RX] 0000 00000040      FastOp     IPin       22          1
[RX] 0000 00000041      FastOp    IPout       22          1
[RX] 0001 00000042      FastOp     IPin       22          1
[RX] 0001 00000043      FastOp    IPout       22          1
```

The same filter (22) is applied to logical interfaces 0 and 1.

IDs are abbreviated in some fields.

For example:

Bind ID 00000022 changes to 22,
Bind ID 00000000 changes to 0,
Bind ID 00000313 changes to 313.

The binding ID is given per assignment. Two IDs mean there are two binding statements that apply to this media card.

The state shown above is `FastOp`, other states are `Operational`, `Dependent`, and `Pend Delet`.

Location tells where the filter is applied, at the incoming IP flow or before the outgoing IP flow.

These are the two binding statements reflected in this example:

```
media FDDI 5 {

     #filter
     bind mail_server_allow {
          vlif 0;           # DAS interface
          direction in;     # inbound traffic
          direction out;    # outbound traffic
          action filter;
     }
}
```

and:

```
media FDDI 5 {

     # filter
     bind mail_server_allow {
          vlif 1;           # DAS interface
          direction in;     # inbound traffic
          direction out;    # outbound traffic
          action filter;
     }
}
```

maint 155 1 0 gives more detail to a binding list on the transmit side of interface 0:

```
GR 05> maint 155 1 0
[TX]
[TX]   vlif   BindID       state location filterID action_cnt
[TX]   0000 00000041      FastOp     IPin       22          1
[TX]   filterID     type   status  access    address  data_addr
[TX]   00000022   ctable (loaded)    0002 0x010e9c90 0x010ea0b0
[TX]   ActionID     type   status  access    address  data_addr
[TX]   00000039   filter (loaded)    0002 0x010ea050 0x00000000
```

Display list of actions

Use **maint 152** to display a list of filter actions configured on the transmit side of cards in slots 5 and 2. These examples show two actions, filtering and logging:

```
GR 05> maint 152
[TX]
[TX]  ActionID     type   status  access
[TX]   00000039   filter (loaded)    0002


GR 05> port 2
Current port card is 2
[TX]
[TX]  GR 02> maint 152
[TX]   ActionID     type   status  access
[TX]   00000020      log (dependent)    0001
```

Display filtering statistics

Use **maint 156** to read statistics for filtered packets, the example shows a transmit-side filter:

```
GR 05> maint 156 gf052
[TX]
[TX]  Inum   loc packets  [filtered   sniffed  forwarded]
[TX]     0  IPin       0          0         0          0
[TX]     0 IPout       0          0         0          0
[TX]     0  IPme       0          0         0          0
[TX]     1  IPin       9          0         9          9
[TX]     1 IPout      34          0        34          0
[TX]     1  IPme       0          0         0          0
[TX]     2  IPin       0          0         0          0
[TX]     2 IPout       0          0         0          0
[TX]     2  IPme       0          0         0          0
[TX]     3  IPin       0          0         0          0
[TX]     3 IPout       0          0         0          0
[TX]     3  IPme       0          0         0          0
```

Clear statistics

Use **maint 57** to clear filtering statistics.

```
GR 05> maint 57
[RX]
[RX] All filtering statistics cleared.
```

Show protocol statistics

Use **maint 58** to review the filtering protocol statistics. These statistics are not cleared by **maint 57**.

```
GR 05> maint 58
[RX]
[RX] -------libfilter->filterd protocol statistics------
[RX] Bad end points on ACK packets:       0
[RX] Bad end points on request packets:   0
[RX] Out of sync ack with none queued:    0
[RX] Out of sync ack with queue:          0
[RX] Out of sync request:                 0
[RX] Retranmitted packets:                0
[RX] Recieved packets:                    14
[RX] Transmitted packets:                 14
```

Configuring Frame Relay

15

This chapter describes how to configure Frame Relay on GRF HSSI and SONET OC-3c media cards. Frame Relay configuration consists of media card tasks and protocol tasks.

Section 1: Introduction to Frame Relay

This section briefly describes the GRF implementation of Frame Relay. Both UNI, routed Frame Relay, and NNI, switched Frame Relay, can be configured.

Section 2: Media card configuration

This section provides a brief overview of media card tasks in which you configure the card in the appropriate Card and System profiles, and also configure logical interfaces in the `/etc/grifconfig.conf` file. Please refer to the HSSI and SONET chapters in this manual for details about media card tasks, only an overview is presented here.

Section 3: Frame Relay logging

This section describes how to set up Frame Relay logging the first time you configure Frame Relay. The Frame Relay daemon, **fred**, has its own log file.

Section 4: Configuring Frame Relay links

This chapter describes how to configure Frame Relay links using the `/etc/grfr.conf` file and the **grfr** command options.

Section 5: Configuring Frame Relay circuits

This chapter describes the Frame Relay circuit options and how to configure Frame Relay circuits using the `/etc/grfr.conf` file and the **grfr** command options.

Section 6: On-the-fly configuration

This release of Frame Relay supports on-the-fly configuration of links and PVCs without requiring the media card to be reset. The **grfr** command has options to add and delete, enable and disable, and modify links and PVCs.

Section 7: Verifying Frame Relay configurations

This section describes the monitoring information you can obtain from **grfr** display commands.

The following topics are covered:

Introduction to Frame Relay

In a Frame Relay network, each physical connection is called a link. A link is a point-to-point physical connection to another piece of Frame Relay equipment, such as a switch or router.

A virtual circuit is a path from an endpoint through one or more frame relay switches to another endpoint. A circuit goes across one or more links.

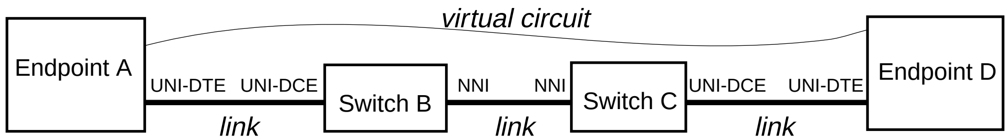


Figure 15-1. Frame Relay virtual circuit and links

Link types

You can specify a Frame Relay link to be UNI-DTE, UNI-DCE, or NNI.

UNI = User to Network Interface
NNI = Network to Network Interface
DTE = Data Terminal Equipment
DCE = Data Communications Equipment

UNI-DTE link

A UNI-DTE device is the device at the edge of a Frame Relay network. It connects to a UNI-DCE device inside the network. Only routing is performed on UNI-DTE links.

UNI-DCE link

A UNI-DCE device is the externally-connecting device at the edge of a Frame Relay network. It connects to a UNI-DTE device outside the network. Both switching and routing can be performed on UNI-DCE links.

NNI link

An NNI link is the link between two Frame Relay switches inside the network. Switching and routing can both be performed on NNI links.

In the example below, endpoint A views the link to B as a UNI-DTE link. Switch B views the link to A as a UNI-DCE link, and the link to C as NNI.

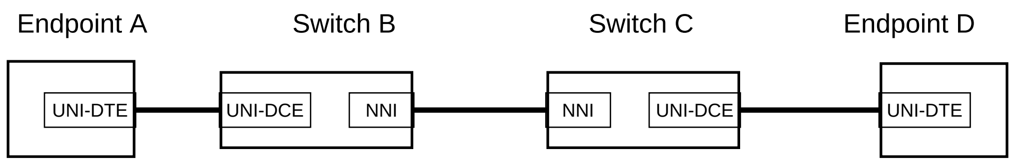


Figure 15-2. NNI link components

PVCs

A PVC (Permanent Virtual Circuit) is a logical path through a Frame Relay network from one endpoint to another endpoint. The path goes across links that connect network devices.

There is a segment for each link between the endpoints. Each segment of a circuit is identified with a Data Link Circuit Identifier (DLCI). The DLCI field is only 10 bits wide, for a maximum of 1024 circuits per link. It is important to note that DLCIs have local significance only. Otherwise, the entire network would be limited to 1024 circuits.

Each link on the path must have a unique DLCI number, as shown in Figure 15-3.

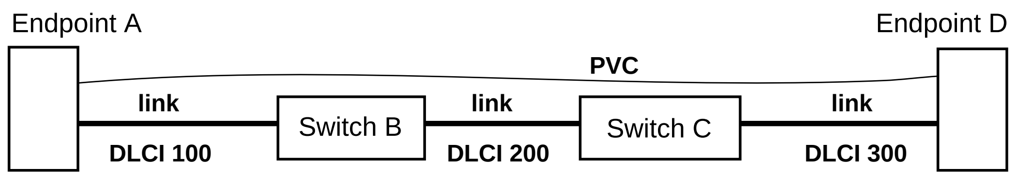


Figure 15-3. Components of a Frame Relay circuit

The circuit from endpoint A to endpoint D passes through two switches, B and C

- from A’s point of view, the circuit to endpoint D uses DLCI 100
- from D’s point of view, the circuit to endpoint A uses DLCI 300

For the switches, each circuit connects two link/DLCI pairs.

- for switch B, the circuit connects DLCI 100 on the A link to DLCI 200 on the C link
- for switch C, the circuit connects DLCI 200 on the B link to DLCI 300 on the D link

A circuit pair is assigned the same DLCI in each direction. The PVC configured from endpoint A to switch B is assigned DLCI 100. The PVC configured from switch B to endpoint A is also assigned DLCI 100.

Each packet has a Frame Relay header. One of the fields in the header is the DLCI. The DLCI determines which circuit a packet is traveling on, and allows a switch to forward the packet to the appropriate next link.

The switches change the DLCI value in the header as the packet crosses the network.

At the entry endpoint, a packet is encapsulated in a Frame Relay header and then placed on a link. Conversely, when the exit endpoint receives a packet from a link, it removes the Frame Relay header before processing. A router is a typical endpoint. An entry endpoint is where a packet enters a Frame Relay network. An exit endpoint is where a packet will leave the Frame Relay network and continue to its destination.

From the router’s point of view, at the end of each circuit is another host with an IP address. From a switch’s point of view, a circuit is merely a connection between two DLCIs on two links. In the example, switch C views the circuit as: “Link B-200 goes to Link D-300”.

All the stations in a Frame Relay network (endpoints and switches) communicate with each other using Local In-Channel Signaling (LICS). A DLCI on each link is reserved for this purpose. LICS messages convey the status of circuits throughout the Frame Relay network. LICS is also referred to as LMI. On the GRF, the LICS messages are processed on the RMS.

Specifications

The Frame Relay Forum is the primary source of Frame Relay specifications.

These specifications are available at:

`http://www.frforum.com/5000/5000index.html.`

GRF implementation features

A GRF router can be simultaneously configured as a Frame Relay router and as a Frame Relay switch. In addition to routing IP and IS-IS traffic over Frame Relay, the GRF provides switching and multicast service features.

The GRF implementation does not support forward explicit congestion notification (FECN) or backward explicit congestion notification (BECN), and does not provide an automated re-routing capability. Frame Relay MIB (RFC 1315) is not supported. Standard LMI (revision 1) is not supported.

Routing

Standard IP routing is supported across Frame Relay links via route circuits (PVCs). IS-IS is also supported across Frame Relay PVCs.

Switching

The Frame Relay switching feature enables a GRF to function as a switch. When a GRF router functions as a Frame Relay switch, it performs layer-2 switching and forwards incoming data from incoming circuits to the appropriate out-going circuits without touching the payload of the data packets.

Frame Relay switching is supported on the HSSI and SONET media cards. The incoming and outgoing endpoints of a switched circuit can be configured on different media types (HSSI and SONET). A complete frame relay network can be built by connecting GRF routers using high-speed links.

Multicast service

Multicast service enables a GRF to function as a Frame Relay multicast server. As a multicast server, the GRF receives multicast data messages from one incoming circuit and forwards the data messages to a group of outgoing circuits.

(Note that Frame Relay multicast is not the same as IP multicast.)

Link options

A link is a HSSI or SONET interface (port). Each link can be configured as:

- UNI-DTE (a router link)
- UNI-DCE (an access switch)
- NNI (a switch, internal to a Frame Relay network or between Frame Relay networks)

Each port on a card can have a different link type.

HSSI and SONET media cards support the configuration of both switch and route circuits on the same Frame Relay link. Switch and route circuits can both be configured on either a UNI-DCE or NNI link. Only route circuits can be configured on a UNI-DTE link.

Circuits

HSSI cards provide two physical interfaces, the SONET card provides a single physical interface. Each physical interface supports 976 Data Link Circuit Identifiers (DLCIs), numbered 16 through 991. This excludes those used for Local In-Channel Signaling (LICS).

Circuits are configured to switch, to route, or to multicast.

- A circuit is configured to switch where two segments of a Frame Relay circuit come together.
- A circuit is configured to route at the endpoint of a Frame Relay circuit.
- A multicast circuit allows an inbound packet to be replicated to multiple destinations.

A circuit can be enabled or disabled, added or deleted, on-the-fly, via **grfr -c cxx** commands. Statistics are individually kept for each circuit and are also displayed using the **grfr -c dxx** commands. The **grfr** command is described later in this chapter.

Statistics

Statistics are kept individually for each PVC, in both directions. PVC statistics include packets and octets received, transmitted, and discarded (due to link congestion. The **grfr -c dps** command displays PVC statistics. Refer to the “Verifying and monitoring Frame Relay” section at the end of this chapter.

Bandwidth enforcement (traffic shaping)

Bandwidth enforcement is assigned on a per-circuit basis. Three bandwidth enforcement parameters can be configured:

- Committed Information Rate (CIR)
- Burst Excess (Be)
- Committed Burst (Bc)

Each circuit is guaranteed a certain bandwidth, the Committed Information Rate, or CIR. Each circuit is allowed to consume bandwidth beyond the CIR to a second threshold, CIR+Be (Burst Excess), above which all packets are dropped. Between the CIR and CIR+Be, packets are no longer guaranteed, and may be dropped by a congested network. These packets are considered Discard Eligible, and are marked as such with the DE bit set in the Frame Relay header. The Committed Burst Size (Bc) is the maximum amount of subscriber data the network agrees to transfer, under normal conditions, during a specified time interval. These parameters are discussed further in the “Configuring Frame Relay circuits” section.

Note: Oversubscribing is not prevented.

The GRF supports asymmetric PVCs. You can configure a different set of bandwidth enforcement parameters on each direction of a circuit.

LICS protocols

The GRF implementation supports the following Local In-Channel Signaling (LICS) protocols:

- ANSI T1.617 Annex D
- CCITT Q.939 Annex A
- LICS disabled

Here are the supported Annex D and Annex A link parameters:

- T391 - the heartbeat poll interval, default is 10 seconds
- T392 - polling verification timer, default is 15 seconds
- N391 - full status polling cycle, default is six polls
- N392 - error threshold, default is three errors
- N393 - count of monitored events, default is four events

Interoperability

The GRF interoperates with any networking equipment that supports HSSI and SONET media interfaces. The connecting device can run Annex A, Annex D, or no LICS protocols.

Note: There is no standard mechanism defined for carrying Frame Relay over SONET.

IS-IS protocol support

HSSI and SONET cards support IS-IS over Frame Relay.

IS-IS is a link state interior gateway protocol (IGP) originally developed for routing ISO/CLNP (International Organization for Standardization / Connectionless Network Protocol) packets. In ISO terminology, a router is referred to as an “intermediate system” (IS). IS-IS intra-domain routing is organized hierarchically so that a large domain may be administratively divided into smaller areas using level 1 intermediate systems within areas and level 2 intermediate systems between areas.

Refer to the *GRF GateD Manual* for more information.

SNMP support for circuit tables

The Frame Relay DTE MIB (RFC 1315) is now supported.

To look at a circuit table for a specific logical interface, you must use an SNMP application.

To begin, display the router interface table. In the very first column, this table gives you the interface index number associated with each logical interface.

Here is a portion of a representative interface table:

```
GRF site_box Interface Table
IfEntry
Index  Descr  Type   MTU  Speed    PhysAddress AdminStatus OprStatus ...
25     gs023  hssi  4352  52000000             up          up
```

Now display the Frame Relay circuit table. In that table, use the interface index number to locate the information for the specific logical interface.

Here is a portion of the information available from a sample circuit table:

```
Frame Relay Circuit Table
FRCircuitEntry
Index DLCI State RecFECNs RecBECNs SentFrames SentOctets RecFrames ...
25    403  Active  0        0      962        84556      969
26    404  Active  0        0      1002       85940      1009
```

Introduction to fred, the FR daemon

The GRF Frame Relay daemon is known as **fred**. This program is responsible for configuring, administering, and monitoring Frame Relay interfaces and circuits on the media cards. **fred** is the source or destination of all Local In-Channel Signaling (LICS) packets.

The Frame Relay link and circuit configuration file is `/etc/grfr.conf`. **fred** reads this file and other internal data structures to build PVC and link tables.

grfr is the program that changes Frame Relay configuration and also displays current Frame Relay status. Commands in the format **grfr -c cxx** add, enable, disable, or delete PVCs and links, or modify link configurations. Commands in the format **grfr -c dxx** display link and PVC status and statistics.

To configure and attach route circuits to logical interfaces, **fred** reads the GRF IP address file, `/etc/grifconfig.conf`, and the Frame Relay configuration file `/etc/grfr.conf`. Frame Relay error and event messages are collected by **fred** and sent to `/var/log/fred.log`.

PVC and link tables

fred, via **grfr**, gets configuration data from the `/etc/grfr.conf` file to build the link table and a corresponding PVC table. Then **fred** downloads a copy of the PVC table to each Frame Relay media card. **fred** updates the tables from incoming LICS messages and from media card status (link up/down) messages, and updates the copies on the media cards. The link table contains 32 entries, two ports per slot for up to 16 slots. Each link entry includes data from the Link section of the `grfr.conf` file, and has a pointer to a corresponding PVC table.

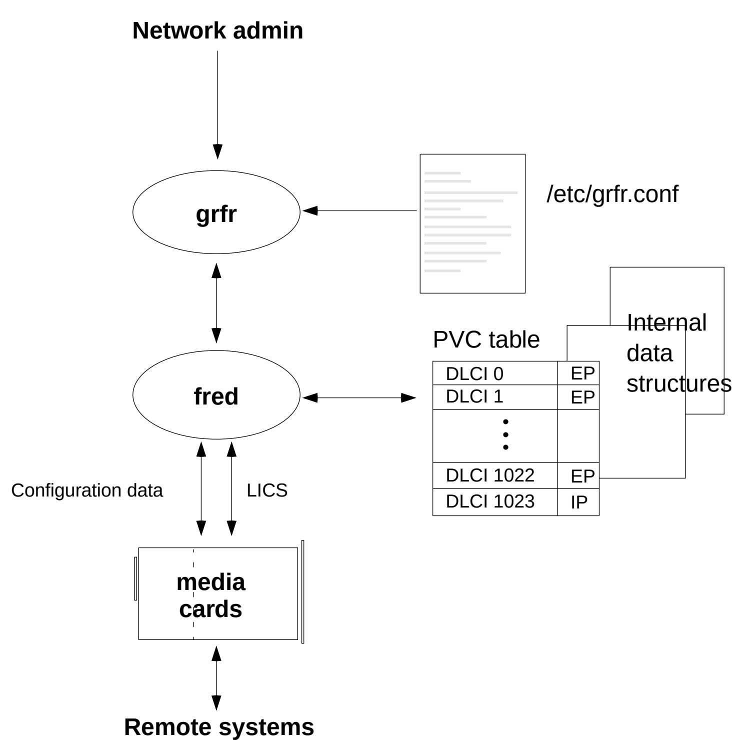


Figure 15-4. Interactions among fred, grfr, and media cards

Route circuits

Route circuits are defined using the `pvc` or `pvcr` keyword in `/etc/grfr.conf`. A route circuit may be configured on any link type, UNI-DTE, UNI-DCE, and NNI.

Switch circuits

A switch circuit is composed of two segments, each on a different link. Switch circuits are defined using the `pvcs` keyword in `/etc/grfr.conf`. A switch circuit may be configured on a UNI-DCE or NNI link. The endpoints of a switch circuit can be configured on different media types (HSSI and SONET).

Multicast circuits

A multicast circuit may be configured on a UNI-DCE or NNI link.

For One-Way multicast, a unicast circuit must already exist between the root and each of the multicast members. These multicast circuits are defined using the `pvcm1` keyword in `/etc/grfr.conf`.

For Two-Way multicast, a unicast circuit is added for each member of the multicast group. These multicast circuits are defined using the `pvcm2` keyword in `/etc/grfr.conf`.

N-Way multicast circuits are defined using the `pvcmn` keyword in `/etc/grfr.conf`.

LICS processing

Local In-Channel Signaling (LICS) processing is implemented in accordance with specifications from the Frame Relay Forum, and the Sprint Frame Relay Switch Specification (5404.03). CCITT Q.933 Annex A and ANSI T1.617 Annex D are implemented. LICS can also be disabled.

On each link, one circuit is reserved for LICS traffic. LICS procedures are performed on all link types, UNI-DTE, UNI-DCE, or NNI.

From a link configured as a UNI-DTE, **fred** sends a poll (status enquiry) to the UNI-DCE to which it is attached every *T391* seconds. Every *N391* polls, the status enquiry message is for a full status report. The event monitoring period is *N393* polls (a sliding window). If there is no response to *N392* polls during a monitoring period, the link is considered down. A full status message contains information about all the circuits on the link.

From a link configured as a UNI-DCE, **fred** responds to polls from the UNI-DTE to which it is attached. If a poll is not received within *T392* seconds of the last poll, or the sequence number is incorrect, an error is logged. The event monitoring period is *N393* polls received or missed (a sliding window). If *N392* errors are logged during an event monitoring period, the link is considered down.

From a link configured as an NNI link, **fred** both polls and answers polls from the switch to which it is attached.

fred keeps timers for each link to trigger a poll and to note missed polls.

grfr command functions

The **grfr** command provides a way to display configuration information, status and statistics of switch circuits, multicast group and modify configurations.

Functions include

- Configuration information to aid debugging includes link parameters, circuit endpoint parameters, a switch circuit, or a multicast group.
- Status information to aid debugging and provide data for analysis and reports includes statistics for a link, a switch circuit, and a multicast group.
- Temporary and minor configuration changes, such as enabling or disabling a circuit or endpoint, adding or deleting a switch circuit can be made using **grfr**. Permanent and major changes must be made via the `grfr.conf` file.

Examples of **grfr** display and configuration commands are found at the end of this chapter and in the *GRF Reference Guide*.

Debugging and log levels

Four debug levels (1 to 4) manage event logging. Level 1 logs the lowest number of debug messages and level 4 provides the highest, level 1 is set by default. Log messages are written by default to the `/var/log/fred.log` file.

You can set and change debug level on-the-fly using the **grfr** command **grfr -c csd -d *level***. These debug levels do not impact the performance of the card.

Level 1 - logs error and main transition events such as link active and inactive. Use this level for normal operations. You can change it on-the-fly.

Level 2 - logs all events related to the LMI protocols. These include sending, receiving, status enquiries, and status responses.

Level 3 - logs same events as in level 2, but provides more details and includes the contents (in hex) of all messages sent and received.

Level 4 - log messages include all activities to and from the media card.

Multicast service

Frame Relay multicast service enables a GRF router to function as a multicast server. A multicast server is a system (a GRF) or switch that receives multicast data messages from one incoming circuit and forwards the data messages to a group of out-going circuits. Multicast services are supported on switch circuits only.

Frame Relay provides the three types of multicast service defined by the Frame Relay Forum in the *Frame Relay PVC Multicast Service and Protocol Description* document.

- One-Way
- Two-Way
- N-Way

In Frame Relay multicast, one switch (a node) within the network is designated as a “Multicast Server” and provides the multicast service. Messages to be multicast are first sent to the multicast server and then, at the multicast server, the messages are replicated and sent to members of the multicast group.

Frame Relay uses the term “upstream circuit” to refer to a circuit where a multicast server *receives* multicast messages. The term “downstream circuit” refers to a circuit where a multicast server *sends* multicast messages. These terms are also applicable to a switch circuit.

In both One-Way and Two-Way Multicast, one station acts as the root station.

One-way multicast

In One-way multicast, the root station sends traffic on a special circuit that delivers the data to all the other members of the multicast group. Point-to-point Frame Relay connections are established to all leaves in the multicast group. The root station maintains

This method requires that a unicast circuit also exist between the root station and each member of the group. Each member of the group receives its multicast packets on the unicast circuit, as if it had been sent by the root on that circuit. If the members of the group wish to communicate something back to the root, they send that traffic back on the unicast circuit. The root receives this traffic on the unicast circuits.

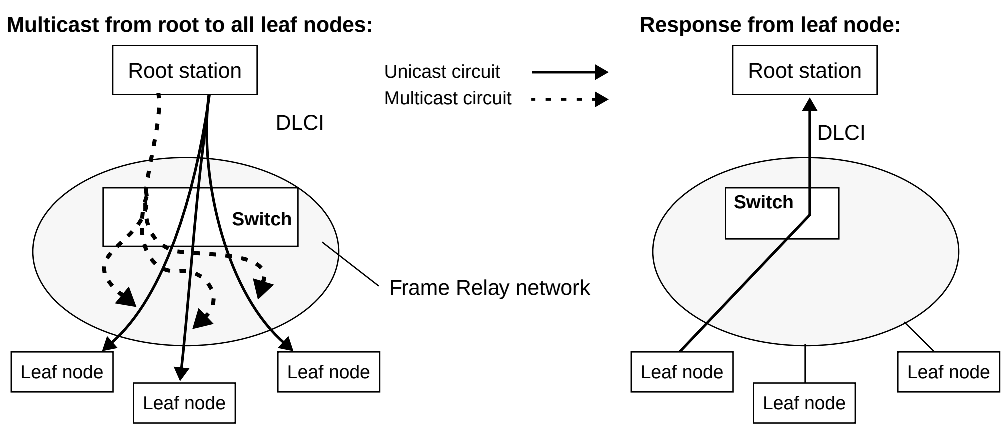


Figure 15-5. Diagram of one-way multicast circuits

Two-way multicast

In Two-way multicast, duplex transmissions are used. In one direction the data units are multicast, in the other direction they are concentrated. Unicast circuits are not required between the root station and the members of the multicast group, but such circuits are permitted.

All members and the root use a special multicast circuit. Data transmitted by the root goes to all the members. Data transmitted by the members is sent only to the root using the multicast circuits.

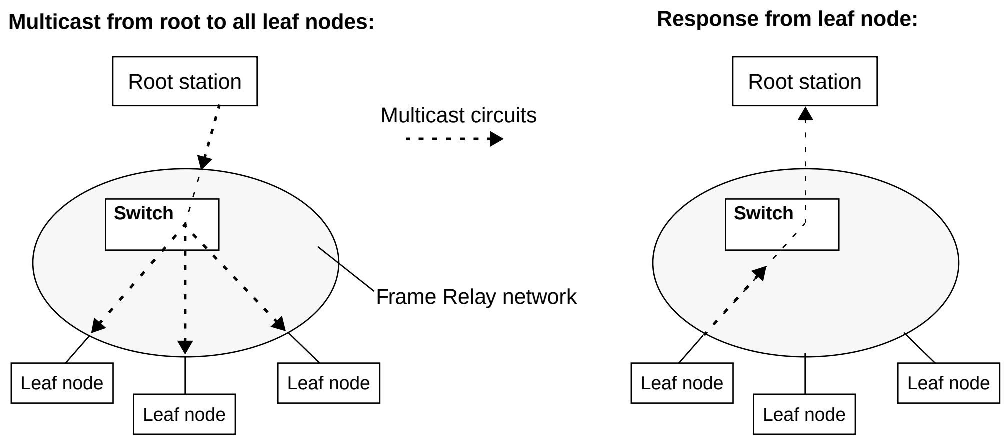


Figure 15-6. Diagram of two-way multicast circuits

N-way multicast

In N-Way multicast, all transmissions are duplex and all are multicast. All members of the group are peers and have special multicast circuits. Any data sent on these multicast circuits gets sent to all the other members of the group.

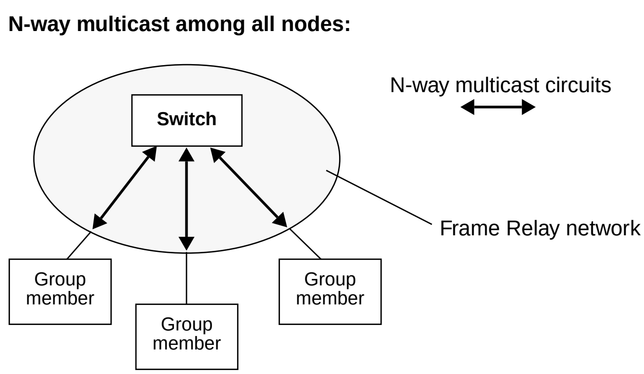


Figure 15-7. Diagram of N-way multicast circuits

Before you start...

Before you configure the Frame Relay protocol, be sure you configure the media cards themselves.

1 Card profile parameters

You must set SONET and/or HSSI parameters in the Card profile.

These parameters include framing protocol, CRC, and internal clock. Refer to the HSSI and SONET chapters in this manual for more information.

2 IP address assignment

Identify the endpoint router logical interfaces in `/etc/grifconfig.conf`.

3 Configure Frame Relay logging

You must start Frame Relay logging the first time you configure Frame Relay.

The next sections provide an overview of these tasks.

First, configure the media cards

Before you configure the Frame Relay protocol, be sure you configure the media cards themselves.

1. Specify interface names in /etc/grifconfig.conf

Identify the logical interfaces on the media card. At minimum, you must identify one logical interface for each physical port on which you will run Frame Relay.

You can assign up to 128 logical interfaces per HSSI Frame Relay port. Here are sample entries in `/etc/grifconfig.conf` for the first and last logical interfaces on each port of the HSSI card in slot 3:

```
# name   address   netmask         broad_dest   arguments
gs030   192.0.2.1   255.255.255.0  192.0.2.255
gs037f   192.0.99.1   255.255.255.0  192.0.99.255
gs0380   192.0.2.15   255.255.255.0
gs03ff   192.0.99.15   255.255.255.0
```

You can assign up to 128 logical interfaces to the SONET Frame Relay port. Here are sample entries in `/etc/grifconfig.conf` for the first and last logical interface on the SONET card in slot 6:

```
# name  address     netmask         broad_dest   arguments
go060   192.0.2.1   255.255.255.0
go067f   192.0.2.28   255.255.255.0
```

Note: Interface names are case sensitive. Always use lower case letters when defining interface names.

2. Specify card and port-level Frame Relay settings

You may have already specified the Frame Relay protocol and CRC settings earlier when you specified HSSI port parameters in the Card profile. Here is the procedure to assign protocol to the HSSI card, and set required CRC and clock values to port 0 on the card in slot 5.

Frame Relay is the default HSSI protocol, but you may want to check the setting has not been changed. If needed, change the protocol setting and do a write to save your change:

```
super> read card 5
CARD/5 read
super> list
card-num* = 5
media-type = hssi
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = Cisco-HDLC
ether-verbose = 0
ports = <{ 0{ off on 10 3} {single off}{"" "" 1 sonet internal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 1000 10 2147483647 10 10 10 }
```

Go to the HSSI settings on port 0. A “shortcut” command is used here. Set CRC to 32-bit if the MTU for the device on the other end of the wire is over 4096, otherwise, specify 16-bit: Set source clock to 1 if a null-modem cable is being used to connect directly to another DTE. Otherwise, set source clock to 0.

Boolean field, '0' or '1'

```
super> read card 5
CARD/5 read
super> list ports 0 hssi
super> list hssi
source-clock = 1
CRC-type = 16-bit
super> set CRC-type = 32-bit
super> set source = 0
super> write
CARD/5 written
```

Starting Frame Relay logging

You must start Frame Relay logging the first time you configure Frame Relay.

During site installation, system logging must be configured, it does not begin automatically. The *GRF 400/1600 Getting Started* and *GRF Configuration and Management* manual both describe how to configure logging to an external device.

These are the steps specifically required to configure Frame Relay logging.

1 Create the `fred.log` file:

```
super> sh
# cd /var/log
# touch fred.log
```

2 Edit the `/etc/syslog.conf` file to have **syslogd** log to `fred.log`:

```
# cd /
# cd /etc
# vi syslog.conf
```

The entries should look like the following:

```
*.err;*.notice;kern.debug;lpr,auth.info;mail.crit      /var/log/messages
cron.info                                  /var/log/cron
local0.info                                /var/log/gritd.packets
local1.info                                /var/log/gr.console
local2.*                                   /var/log/gr.boot
local3.*                                   /var/log/grinchd.log
local4.*                                   /var/log/gr.conferrs
local5.*                                   /var/log/mib2d.log
```

Add the following line at the end of the file:

```
local6.*                                   /var/log/fred.log
```

Save the file and exit.

3 Modify `/etc/grclean.conf` to specify a size limit for `fred.log`:

```
# vi syslog.conf
```

The file entries should look like the following:

```
size=10000
logfile=/var/log/cron
size=10000
logfile=/var/log/aitmd.log
size=10000
logfile=/var/log/fred.log

************************************************************************
```

Add a `fred.log` entry after the `var/log` entry. An example is shown below.
The file size (in K) you specify will depend upon the available memory resources.

```
size=1000
logfile=/var/log/fred.log
```

4 Save all changes and reboot:

```
# grwrite -v
# reboot -i
```

If you are upgrading software rather than doing an initial installation, you will have to signal (HUP) **syslogd** to re-read the `syslog.conf` file so the Frame Relay changes are incorporated.

Configuring Frame Relay links

Configure link parameters in the `/etc/grfr.conf` configuration file. Please see the *GRF Reference Guide* for a template of `/etc/grfr.conf` and other GRF configuration files.

Link parameters are set in the Link section of `/etc/grfr.conf`. On each link you can configure the following required and optional parameters.

Required parameters

- Link descriptors - *required for all links*
  Specify the link's slot number and port number in decimal.
- Link type - *required* for UNI-DCE and NNI links
  Specify the link to be UNI-DTE, UNI-DCE, or NNI.
  Default is UNI-DTE.
- LMI type - *required* for AnnexA and AnnexD
  Specify the link to be AnnexA or AnnexD.
  Default is none.

Optional parameters

- Link name
  Each link can be named for administrative convenience such as `link_to_chicago`.
- Enabled Y|N
  Enable the link, the default is Y.
- T391- heartbeat poll interval
  T391 represents the Link Integrity Verification timer. This link option specifies the interval (*T391 seconds*) the user device waits before sending each status inquiry message.
  Default is 5 seconds, options are 5, 10, 20, 25, or 30.
  – A UNI-DTE sends polls to the connected UNI-DCE.
  – Two NNIs send polls to each other.
  – A UNI-DCE does not send polls to a UNI-DTE.
- N391- full status polling cycle
  N391 represents the Full Status Polling cycle. This link option requests a full status report every *N391* polls. N391 usually applies to the user equipment.
  Default is 10 polls, range is `1..`255.
- T392 - polling verification timer
  T392 represents the Polling Verification timer. This link option specifies the number of seconds the network waits for an expected status inquiry message. If a poll is not received within *T392* seconds of the previous poll, a missed poll error is logged. T392 should be set to a value greater than T391.
  Default is 15 seconds.
- N392 - error threshold
  N392 represents the Error Threshold number. This link option specifies the number of missed poll errors counted in a single monitoring period which causes the link to be taken down. N392 should be set to a value lower than or equal to N393.
  Default is 3 errors, the range is `1..`10.

- N393 - monitored events count

  N393 represents the Monitored Events count. This link option determines the length of a monitoring period for a link declared inactive. Each period is actually a sliding window that is *N393* events long, where an event is a received poll or a missed poll. After a channel or user device is declared inactive, the network device waits *N393* cycles of positive poll responses before declaring the channel or device active again. If N393 is set to a value much lower than N391, a link could go in and out of an error condition without the user equipment or network being notified.

  Default is 4, the range is `1..10`.

- AutoAddGrif

  Enables remote devices to assign a PVC to a GRF interface.
  Default is no.

Port 0, interface 0 requirement for HSSI NNI

Interface 0 must be configured on port 0 for HSSI Frame Relay NNI interfaces configured on port 1 to operate.

The entry can be either an active or dummy interface. For interface `gs0380` to work, you must define the following interfaces in `/etc/grifconfig.conf`:

```
# name address  netmask broad_dest argument
gs0380 192.168.0.1 255.255.255.0
gs030 - - - up
```

If the media card has already booted, then interface 0 must be added to the `/etc/grifconfig.conf` file and the **fred** daemon must be restarted. This requirement applies only to HSSI Frame Relay switching.

Installing links with grfr commands

In earlier versions of Frame Relay, you installed a link by resetting the media card with the **grreset** command.

Now, after you configure the logical interface in `/etc/grifconfig.conf` and configure the desired link parameters in `/etc/grfr.conf`, you use a **grfr** command to install the configured link. There is no need to reset the media card.

Create and install link

To create *and* install the new link, execute a **grfr - c ccl -s *slot* -l *port*** command after you configure the logical interface in `/etc/grifconfig.conf` and configure the desired link parameters in `/etc/grfr.conf`.

```
# grfr -c ccl -s 5 -l 0
LINK Defined: Slot 5, link 0
```

The response tells you whether or not the link is successfully created.

To verify the link is there, use the **grfr -c dlc** display link configuration command:

```
# grfr -c dlc
C O N F I G U R E D  L I N K S :
=================================
Name:   S/P:  LMI:   Link:  Autogrif: N391: N392: N393: T391: T392: S:
----    ---   ---    ----   --------  ----  ----  ----  ----  ----  --
linkDD  5 /0  ANNEX-D  NNI      None    6     3     4     10    15  Ae
```

Disable a link

To disable a configured link and make all associated PVCs inactive, execute a **grfr - c cdl -s *slot* -l *port*** command:

```
# grfr -c cdl -s 5 -l 0
LINK Disabled: slot 5, link 0
```

Enable a link

To enable a configured link and its associated PVCs, execute a **grfr - c cel -s *slot* -l *port*** command:

```
# grfr -c cel -s 5 -l 0
LINK Enabled: slot 5, link 0
```

Remove a link

To remove a configured link and all assigned PVCs, execute a **grfr - c crl -s *slot* -l *port*** command:

```
# grfr -c crl -s 5 -l 0
```

```
LINK Deleted: slot 5, link 0
```

Note: The **grfr -c ccl** command does not restore the link to state before a **grfr -c crl** command is executed.

When a link is removed using the **grfr -c crl** command, the link and all underlying PVCs are removed. A subsequent **grfr -c ccl** command reestablishes the link but not the PVCs. The **fred** daemon needs to be restarted in order to reestablish all of the PVCs on the link.

Modify a link

To change the parameters for a configured link, make the desired changes in `/etc/grfr.conf`. Then, install the changes on the link using the **grfr - c ccl -s *slot* -l *port*** command:

```
# grfr -c ccl -s 5 -l 0
LINK Modified: slot 5, link 0
```

Use the **grfr -c dlc** display link configuration command to verify the change is made. The link state will be reset to "Inactive" if the link type is changed.

Link configuration example

In the example, Frame Relay links are configured between SONET and HSSI cards:

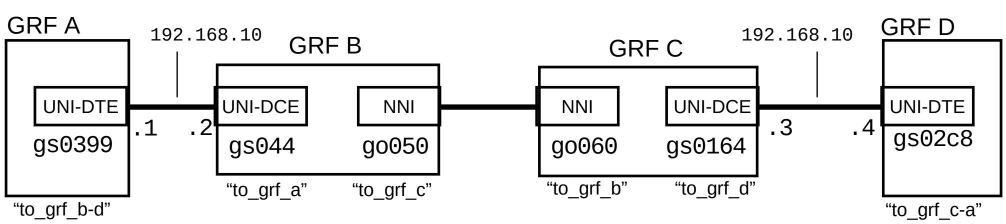


Figure 15-8. Frame Relay link configuration example

The GRF A, B, C, and D examples below show the Link section entries in `/etc/grfr.conf` and the IP interface assignments in `/etc/grifconfig.conf`.

GRF A

Port 0 in the HSSI card in slot 3 has a UNI-DTE link via logical interface `gs0399` to a HSSI card in GRF B.

1 `/etc/grfr.conf` Link entry:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 3 0 name="to_grf_b-d" LMIType=AnnexD
```

2 `/etc/grifconfig.conf` Interface entry:

```
# name address  netmask broad_dest argument
gs0399 192.168.10.1 255.255.255.0
```

3 Install link with **grfr -c ccl -s *slot* -l *port*** command:

```
# grfr -c ccl -s 3 -l 0
LINK Defined: Slot 3, link 0
```

GRF B

Port 0 in the HSSI card in slot 4 has a link via logical interface 4 to a HSSI card in GRF A.

Port 0 in the SONET card in slot 5 has a link via logical interface 0 to a SONET card in GRF C.

1 `/etc/grfr.conf` Link entries:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 4 0 name="to_grf_a" Linktype=UNI-DCE LMIType=AnnexD
link 5 0 name="to_grf_c" Linktype=NNI LMIType=AnnexD
```

2 `/etc/grifconfig.conf` Interface entries:

```
# name address  netmask broad_dest argument
gs044 192.168.10.2 255.255.255.0
go050 192.168.10.2 255.255.255.0
```

3 Install links with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 4 -l 0
LINK Defined: Slot 4, link 0
# grfr -c ccl -s 5 -l 0
LINK Defined: Slot 5, link 0
```

GRF C

Port 0 in the SONET card in slot 6 has a link via logical interface `go066` to a SONET card in GRF B.

Port 0 in the HSSI card in slot 1 has a link via logical interface `gs0164` to a HSSI card in GRF D.

1 `/etc/grfr.conf` Link entries:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 6 0 name="to_grf_b" Linktype=NNI LMIType=AnnexD
link 1 0 name="to_grf_d" Linktype=UNI-DCE LMIType=AnnexD
```

2 `/etc/grifconfig.conf` Interface entries:

```
# name address  netmask broad_dest argument
gs0164 192.168.10.3 255.255.255.0
```

3 Install links with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 6 -l 0
LINK Defined: Slot 6, link 0
# grfr -c ccl -s 1 -l 0
LINK Defined: Slot 1, link 0
```

GRF D

Port 1 in the HSSI card in slot 2 has a link via logical interface `gs02c8` to a HSSI card in GRF C.

1 `/etc/grfr.conf` Link entry:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 2 1 name="to_grf_c-a" LMIType=AnnexD
```

2 `/etc/grifconfig.conf` Interface entry:

```
# name address  netmask broad_dest argument
gs02c8 192.168.10.4 255.255.255.0
```

3 Install links with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 2 -l 1
LINK Defined: Slot 2, link 1
```

Configuring Frame Relay circuits

You automatically specify circuit type by the section of `/etc/grfr.conf` file in which you configure the circuit:

- Route circuits, PVC section
- Switch circuits, PVCS section
- 1-way multicast, PVCM1 section
- 2-way multicast, PVCM2 section
- N-way multicast, PVCMN section
- Endpoint parameters, PVCEP section

You can also configure ATMP PVCs in the PVCATMP section of `/etc/grfr.conf`. Refer to the ATMP chapter in this manual for more information.

Route circuits - PVC/PVCR section

The keywords PVC and PVCR are interchangeable, and are processed for configuration purposes in exactly the same way.

You configure route circuits and their parameters in the PVC section of `grfr.conf`:

```
#    lif    DLCI Peer IP Address   Optional Parameters
#    ===    ==== ===============   ===================
pvc   gs050  100 0.0.0.0
pvc   gs050  150 192.1.13.200
pvc   gs050  250 192.1.14.200 CIR=5600000 Bc=5600000 Be=2400
pvc   gs0e80  405 0.0.0.0
```

Required parameters

- Logical Interface name (`lif`)
  Circuits are grouped onto logical interfaces. You can have all circuits on a given link on the same logical interface, or each circuit on its own logical interface, or any grouping in between.
- Endpoint
  Specify the DLCI of this circuit, which ends here at the router.
- Peer IP Address
  Specify the IP address of the host or router at the other end of this circuit. If this parameter is set to 0.0.0.0, Inverse ARP is used to determine the IP address.

Optional parameters

- Name - *optional*
  Each route circuit can be named for convenience.
- Enabled Y|N - *optional*
  Enable circuit, default is Y.
- IS-IS Y|N - *optional*
  Enable IS-IS, default is N.

- Bandwidth enforcement parameters - *optional*

  These parameters are assigned per PVC.

  These definitions use a committed rate measurement interval, Tc. Tc is the time interval during which the user is allowed to send Bc committed amount of data or Bc committed amount of data PLUS Be excess amount of data. Generally, Tc=Bc/CIR. Bc and CIR are usually set to the same value, obtaining a Tc of one second.

  – CIR - Committed Information Rate

  The Committed Information Rate (CIR) is the subscriber data throughput that the network commits to supporting under normal network conditions.

  CIR specifies the bits per second that the network should be able to deliver on this circuit without dropping data.

  The sum of the CIR values on all circuits on a link should not exceed the bandwidth of the link. Default is 55M bits (55000000).

  Note: Oversubscription can occur when the total of CIR values exceeds the available bandwidth. Frames may be dropped when CIR exceeds the bandwidth of the physical link.

  – Bc - Committed Burst Size

  The Committed Burst Size (Bc) is the maximum amount of subscriber data the network agrees to transfer, under normal conditions, during time interval Tc.

  Bc specifies the amount of data (in bits) that the network should be able to deliver on this circuit without dropping packets during a fixed period of time: Tc. Tc = Bc/CIR.

  Typically, Bc is set to be the same as CIR, for a Tc of 1 second. Default is 55M bits (55000000).

  – Be - Excess Burst Size

  The Excess Burst Size (Be) is the maximum amount of uncommitted data in excess of Bc that the network will attempt to deliver during time interval Tc.

  Be specifies the amount of data (in bits) above Bc that the network will attempt to deliver on this circuit during the time period Tc.

  This data is eligible for discard (DE) if the network becomes congested. Default is 0.

Port 0, interface 0 requirement for HSSI NNI

Interface 0 must be configured on port 0 for HSSI Frame Relay NNI interfaces configured on port 1 to operate.

The entry can be either an active or dummy interface. For interface `gs0380` to work, you must define the following interfaces in `/etc/grifconfig.conf`:

```
# name address  netmask broad_dest argument
gs0380 192.168.0.1 255.255.255.0
gs030 - - - up
```

If the media card has already booted, then interface 0 must be added to the `/etc/grifconfig.conf` file and the **fred** daemon must be restarted. This requirement applies only to HSSI Frame Relay switching.

Installing PVCs with grfr command

In earlier versions of Frame Relay, you installed a PVC by resetting the media card with the **grreset** command.

Now, after you configure the logical interface in `/etc/grifconfig.conf` and configure the desired PVC parameters in `/etc/grfr.conf`, you use a **grfr** command to install the configured PVC. There is no need to reset the media card.

Create and install a PVC

Each PVC must be configured "on" a logical interface. Multiple PVCs can be configured on the same logical interface. The logical interface must be configured and assigned an IP address in `/etc/grifconfig.conf`.

Execute a **grfr - c ccp -s *slot* -l *port* -i *DLCI*** command to install each Frame Relay PVC you create. The **grfr -c ccp** command installs all types of Frame Relay circuits: PVCs, PVCRs, PVCSs, and so on.

```
# grfr -c ccp -s 5 -l 0 -i 100
PVC Defined: 5, link 0, dlci 100
# grfr -c ccp -s 5 -l 0 -i 150
PVC Defined: 5, link 0, dlci 150
```

The response tells you whether or not the PVC is successfully created.

To verify the PVC is there, use the **grfr -c dpc** display PVC configuration command:

```
#grfr -c dpc

C O N F I G U R E D   P V C s   :
=================================
(A* = Autoadded, D* = Deleted)
Name        Slot  Port  DLCI   Type    CIR   Bc    Be    State     EPs/ISIS
----        ---   ----  ----   ----    ----  ---   ---   -----     --------
13:0:0      13    0     100    Switch  56K   56K   26K   Active    13:0:0
M1-Group    13    0     150    Mcast-R 56K   56K   26K   Active    13:0:312
```

Disable a PVC

To disable a configured circuit, execute a **grfr - c cdp -s *slot* -l *port* -i *DLCI*** command:

```
# grfr -c cdp -s 5 -l 0 -i 100
PVC disabled: 5, link 0,  dlci 100
```

Enable a PVC

To enable a configured PVC, execute a **grfr - c cep -s *slot* -l *port* -i *DLCI*** command:

```
# grfr -c cep -s 5 -l 0 -i 100
PVC Enabled: 5, link 0,  dlci 100
```

Use HSSI **maint 8** and SONET **maint 88** commands to verify the status of PVCs on the target media card.

Note: PVC 0 (LMI DLCI) is automatically created by **fred** when a link is defined. Users have no control over the DLCI.

GRF Frame Relay network example

This example shows a Frame Relay network with GRFs configured as Frame Relay switches and also as edge routers. A combination of route circuits and switch circuits need to be configured. The next several pages describe the Frame Relay configuration tasks.

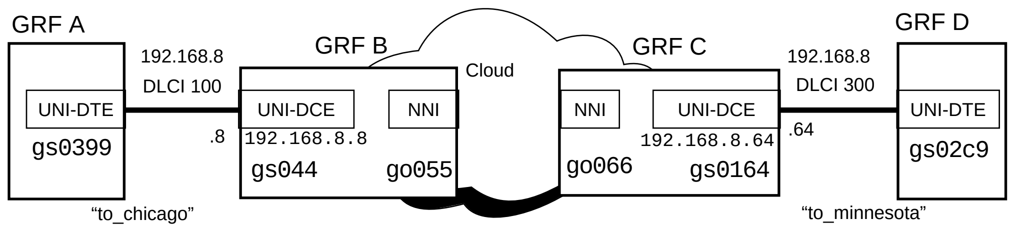


Figure 15-9. GRF Frame Relay network example

Configure the edge routers

For the edge routers, links and route circuits need to be configured. One set goes via Frame Relay HSSI from GRF A to GRF B, and the second goes via HSSI from GRF D to GRF C:

The GRF A and GRF D configuration examples show:

- Frame Relay link (physical port) parameters
- PVC/PVCR section entries in `/etc/grfr.conf`
- IP interface assignments in `/etc/grifconfig.conf` files.

GRF A

A routed circuit is required. Port 1 in the HSSI card in slot 3 has a routed circuit via logical interface `gs0399` to a HSSI card in GRF B.

1 `/etc/grfr.conf` Link entry:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 3 1 LMIType=AnnexA linktype=UNI-DTE
```

2 `/etc/grfr.conf` PVCR entry:

```
#lif DLCI Peer IP Address Optional Parameters
#=== ==== ============== ==================
pvcr gs0399 100 192.168.8.8 Name="to_chicago" isis=Y
```

3 `/etc/grifconfig.conf` Interface entry:

```
# name address  netmask broad_dest argument
gs0399 192.168.2.3 255.255.255.0
```

4 Install PVCR with **grfr -c ccp -s** ***slot*** **-l** ***port*** **-i** ***dlci*** commands:

```
# grfr -c ccp -s 3 -l 0 -i 100
PVC Defined: 3, link 0, dlci 100
```

5 Install link with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 3 -l 1
LINK Defined: Slot 3, link 1
```

GRF D

A link and a routed circuit is required. Port 1 in the HSSI card in slot 2 has a routed circuit via logical interface `gs02c9` to a HSSI card in GRF C.

1 `/etc/grfr.conf` Link entry:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 2 1 LMIType=AnnexA linktype=UNI-DTE
```

2 `/etc/grfr.conf` PVCR entry:

```
#lif DLCI Peer IP Address Optional Parameters
#=== ==== ============== ==================
pvcr gs02c9 300 192.168.8.64 Name="to_minnesota" isis=Y
```

3 `/etc/grifconfig.conf` Interface entry:

```
# name address  netmask broad_dest argument
gs02c9 192.168.8.8 255.255.255.0
```

4 Install PVCR with **grfr -c ccp -s *slot* -l *port* -i *dlci*** commands:

```
# grfr -c ccp -s 2 -l 1 -i 300
PVC Defined: 2, link 1, dlci 300
```

5 Install link with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 2 -l 1
LINK Defined: Slot 2, link 1
```

Configure the Frame Relay switches

For the switches, links and switch circuits need to be configured.

You configure switch circuit parameters in the PVCS section of `/etc/grfr.conf`. These are the configuration

- Segments (endpoints) - *required*

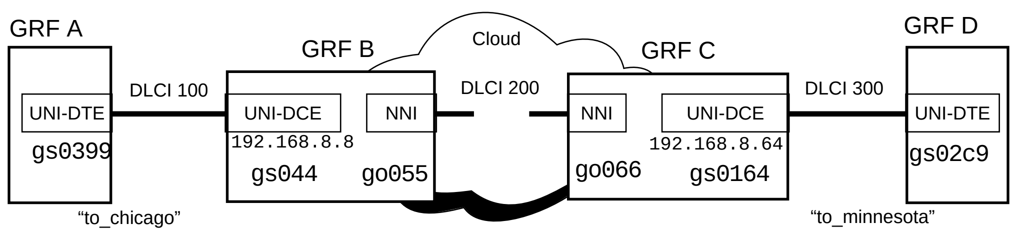
  Specify the chassis slot, card port, and DLCI of each of the two segments of the circuit that meet here at the switch.
- Name - *optional*

  Each switch circuit can be named for convenience.
- Enabled Y|N - *optional*

  Enable circuit, default is Y.
- Bandwidth enforcement parameters - *optional*

  Same as for route circuit parameters described earlier in the PVC section.

In the Frame Relay network example, the switches connect across a Frame Relay cloud:



If the two switches, GRF B and GRF C, were directly connected, they would form a "cloud of two." The Frame Relay configuration is the same for a cloud of thousands as it is for a cloud of two.

A switch circuit is composed of two segments, each on a different link. In the example:

- one circuit is the segment pair, GRF B–GRF A and GRF B–GRF C.
- the other circuit is the segment pair GRF C–GRF B and GRF C–GRF D:

Links and switch circuits need to be configured for GRF B and GRF C.

GRF B

1 `/etc/grfr.conf` Link entries:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 4 0 LMIType=AnnexA linktype=UNI-DCE
link 5 0 linktype=NNI
```

2 `/etc/grfr.conf` PVCS entry:

```
#     EPA      EPB      Optional Parameters
#     ===      ====     ==================
pvcs  4:0:100  5:0:200  Name="minn_chi" CIR=1500000 Bc=1500000 Be=75000
```

3 Install links with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 4 -l 0
LINK Defined: Slot 4, link 0
# grfr -c ccl -s 5 -l 0
LINK Defined: Slot 5, link 0
```

4 Install circuit with **grfr -c ccp -s *slot* -l *port* -i *dlci*** commands.
Use one entry point's slot, port, and DCLI values, **grfr** automatically creates the other.

```
# grfr -c ccp -s 4 -l 0 -i 100
PVC Defined: 4, link 0, dlci 100
```

GRF C

1 `/etc/grfr.conf` Link entries:

```
#Slot Port Optional Parameters
#==== ==== ==================
link 6 0 linktype=NNI
link 1 0 LMIType=AnnexA linktype=UNI-DCE
```

2 `/etc/grfr.conf` PVCS entry:

```
#     EPA      EPB      Optional Parameters
#     ===      ====     ===================
pvcs  6:0:200  1:0:300  Name="chi_minn" CIR=1500000 Bc=1500000 Be=75000
```

3 Install links with **grfr -c ccl -s *slot* -l *port*** commands:

```
# grfr -c ccl -s 6 -l 0
LINK Defined: Slot 6, link 0

# grfr -c ccl -s 1 -l 0
LINK Defined: Slot 1, link 0
```

4 Install circuit with **grfr -c ccp -s *slot* -l *port* -i *dlci*** commands.
Use one entry point's slot, port, and DCLI values, **grfr** automatically creates the other.

```
# grfr -c ccp -s 6 -l 0 -i 200
PVC Defined: 6, link 0, dlci 200
```

Port 0, interface 0 requirement for HSSI NNI

Interface 0 must be configured on port 0 for HSSI Frame Relay NNI interfaces configured on port 1 to operate.

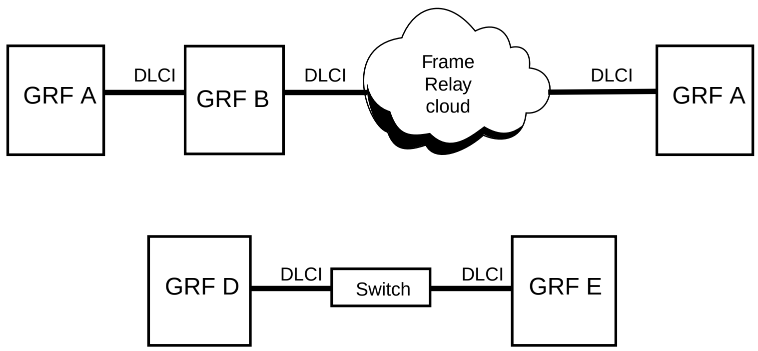
The entry can be either an active or dummy interface. For interface `gs0380` to work, you must define the following interfaces in `/etc/grifconfig.conf`:

```
# name address  netmask broad_dest argument
gs0380 192.168.0.1 255.255.255.0
gs030 - - - up
```

If the media card has already booted, then interface 0 must be added to the `/etc/grifconfig.conf` file and the **fred** daemon must be restarted. This requirement applies only to HSSI Frame Relay switching.

Matching DLCI and IP subnets

This brief summary describes the assignment of matching DLCIs and subnets to Frame Relay circuits.



Connection endpoints	**Requirements**
A to B (direct)	DLCI must match for local connection, must be on same IP subnet.
A to C (across cloud)	DLCI does not need to match for remote connection, must be on same IP sublet.
B to C (across cloud)	same as A–C
D to E (across switch)	same as A–C

Configuring Frame Relay multicast

The next sections describe configuration of Frame Relay multicast and asymmetrical traffic shapes.

One-way multicast - PVCM1 section

In one-way multicast, the root node can send to all leaf nodes. A leaf node can respond to only the root, and only on its unicast circuit.

Configure these one-way multicast parameters in the PVCM1 section of `grfr.conf`:

- First entry (EPR) must be root circuit endpoint - *required*
  Specify the chassis slot, card port, and DLCI of root endpoint.
- Next *n* entries are the endpoints of each member of the group - *required*
  Specify the chassis slot, card port, and DLCI of *n* leaf endpoints.

- Name - *optional*
  Each multicast group can be named for convenience.
- Enabled Y|N - *optional*
  Enable multicast group, default is Y.
- Bandwidth enforcement parameters - *optional*
  Same as for route circuit parameters described earlier in the PVC section.

Example

In this example, a node is the root station for a one-way multicast group consisting of the two leaf nodes. GRF 2 is a Frame Relay switch, and forms a one-node network.

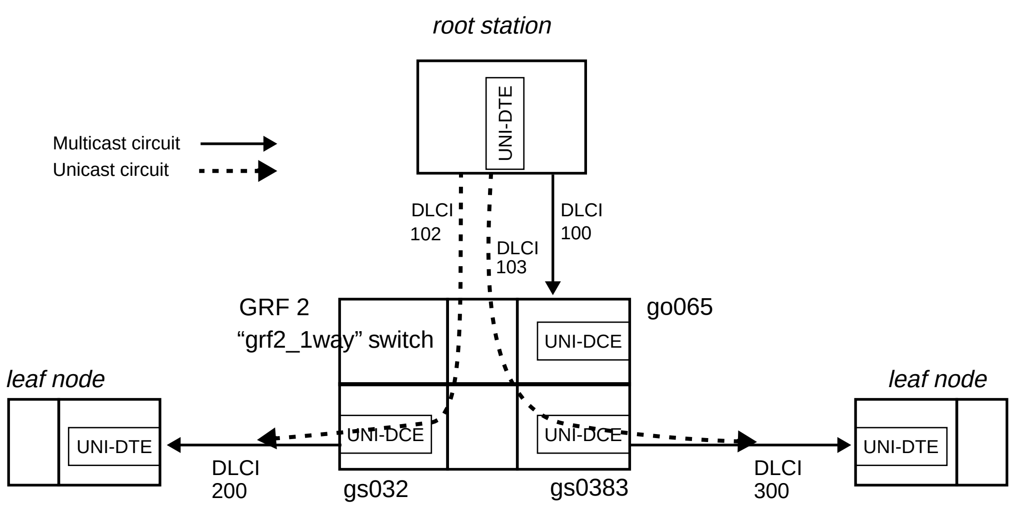


Figure 15-10. One-way multicast example

These are the configuration steps:

1 Configure the unicast circuits in the PVCS section of `/etc/grfr.conf`.

```
      EPA     EPB      Optional Parameters
      ====    ===      ===================
pvcs 6:0:102 3:0:200
pvcs 6:0:103 3:1:300
```

2 Configure the PVCM1 entry in `/etc/grfr.conf` to create the multicast group.

```
      EPR      EP1     EP2       Optional Parameters
      ===      ====    ===       ===================
pvcm1 6:0:100  3:0:200  3:1:300  Name="grf2_1way" CIR=64000 Bc=64000 Be=0
```

3 Configure the logical interface entries in `/etc/grifconfig.conf`.

```
# name address  netmask broad_dest argument
go060  192.168.0.1 255.255.255.0
gs030  192.168.0.2 255.255.255.0  #interface 0 can be active
gs0380  192.168.0.3 255.255.255.0
```

4 Install circuits with **grfr -c ccp -s *slot* -l *port* -i *dlci*** commands.
Use one entry point's slot, port, and DCLI values, ignore the other.

```
# grfr -c ccp -s 6 -l 0 -i 102
PVC Defined: 6, link 0, dlci 102
# grfr -c ccp -s 6 -l 0 -i 103
PVC Defined: 6, link 0, dlci 103
# grfr -c ccp -s 6 -l 1 -i 100
PVC Defined: 6, link 1, dlci 100
```

Although it is not shown in the scope of this example, two switch PVCs (routed PVCs from the root station viewpoint) are also to be configured.

Two-way multicast - PVCM2 section

Configure these two-way multicast parameters in the PVCM2 section of `grfr.conf`:

- First entry (EPR) must be root circuit endpoint - *required*
  Specify the chassis slot, card port, and DLCI of root endpoint.
- Next *n* entries are the endpoints of each member of the group - *required*
  Specify the chassis slot, card port, and DLCI of *n* leaf endpoints.

- Name - *optional*
  Each multicast group can be named for convenience.
- Enabled Y|N - *optional*
  Enable multicast group, default is Y.
- Bandwidth enforcement parameters - *optional*
  Same as for route circuit parameters described earlier in the PVC section.

Example

In this example, a node is the root station for a two-way multicast group consisting of two leaf nodes. GRF 2 is a Frame Relay switch, and functions as a one-node network.

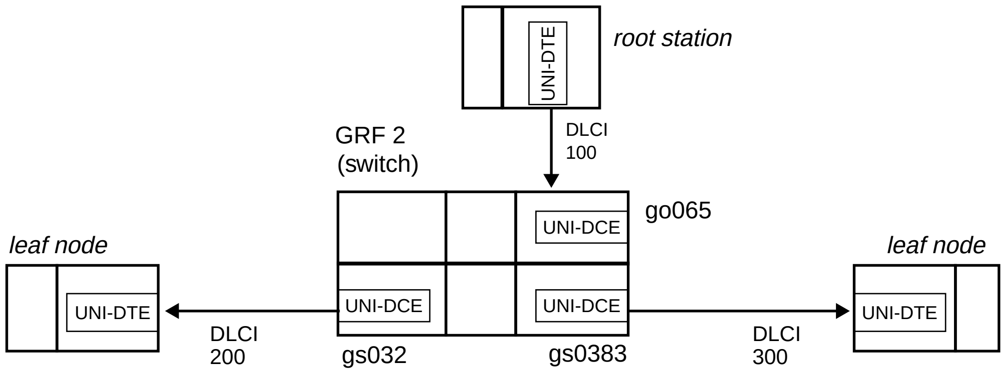


Figure 15-11. Two-way multicast example

1 Configure the PVCM2 entry in `/etc/grfr.conf` to create the group:

```
      EPR      EP1      EP2     Optional Parameters
      ===      ====     ===     ==================
pvcm2 6:0:100  3:0:200  3:1:300  Name="grf2_1way" CIR=64000 Bc=64000 Be=0
```

2 Configure the logical interface entries in `/etc/grifconfig.conf`:

```
# name address  netmask broad_dest argument
gs032 192.168.0.5 255.255.255.0
gs0383 192.168.0.6 255.255.255.0
go065 192.168.0.7 255.255.255.0
```

3 Install the PVCM2 circuit with a **grfr -c ccp -s *slot* -l *port* -i *dlci*** command:

Use one entry point's slot, port, and DCLI values, ignore the others.

```
# grfr -c ccp -s 6 -l 0 -i 100
PVC Defined: 6, link 0, dlci 100
```

N-way multicast - PVCMN section

In N-way multicast there are no leaves, no root, all are equivalent multicast nodes.

Configure these N-way multicast parameters in the PVCMN section of `grfr.conf`:

- Enter the endpoints of *n* members of the group - *required*
  Specify the chassis slot, card port, and DLCI of *n* member endpoints.
- Name - *optional*
  Each multicast group can be named for convenience.
- Enabled Y|N - *optional*
  Enable multicast group, default is Y.
- Bandwidth enforcement parameters - *optional*
  Same as for route circuit parameters described earlier in the PVC section.

Example

In this example, three nodes are in an N-way multicast group. GRF 2 is a Frame Relay switch.

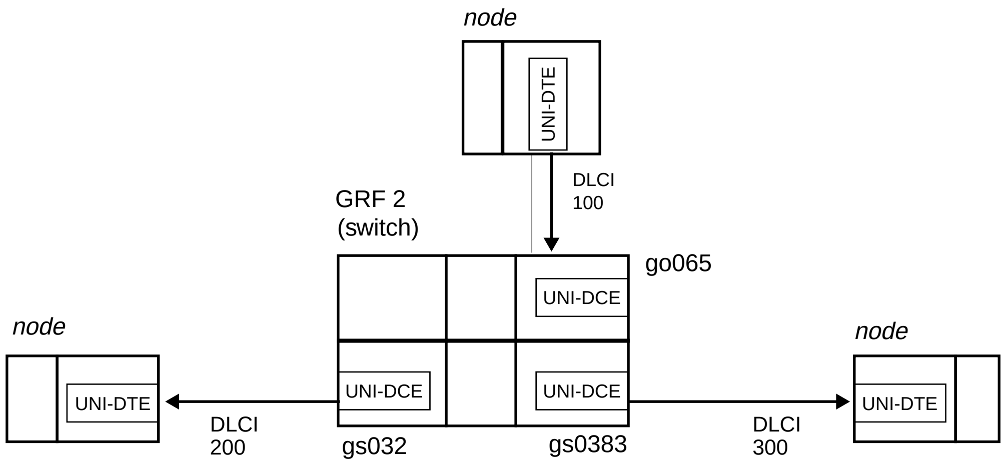


Figure 15-12. N-way multicast example

1 Configure the PVCMN entry in `/etc/grfr.conf` to create the group:

```
      EPR       EP1     EP2     Optional Parameters
      ===       ====    ===     ===================
pvcmn 6:0:100  3:0:200  3:1:300  Name="grf2_nway" CIR=64000 Bc=64000 Be=0
```

2 Configure the logical interface entries in `/etc/grifconfig.conf`:

```
# name address  netmask broad_dest argument
gs032 192.168.0.5 255.255.255.0
gs0383 192.168.0.6 255.255.255.0
go065 192.168.0.7 255.255.255.0
```

3 Install the PVCMN circuit with a **grfr -c ccp -s *slot* -l *port* -i *dlci*** command:
Use one entry point's slot, port, and DCLI values, ignore the others.

```
# grfr -c ccp -s 6 -l 0 -i 100
PVC Defined: 6, link 0, dlci 100
```

Asymmetrical traffic shapes

Configure different bandwidth enforcement parameters for each individual endpoint on a link (an asymmetric circuit) in the PVCEP section of `grfr.conf`:

- Define target endpoint - *required*
  Specify the chassis slot, card port, and DLCI of endpoint.
- Bandwidth enforcement parameters - *required*
  Same as for route circuit parameters described earlier in the PVC section.

Example

This example gives the circuit going to GRF 4 about 2Mb of bandwidth, the traffic coming back to GRF 3 gets 64 Kb.

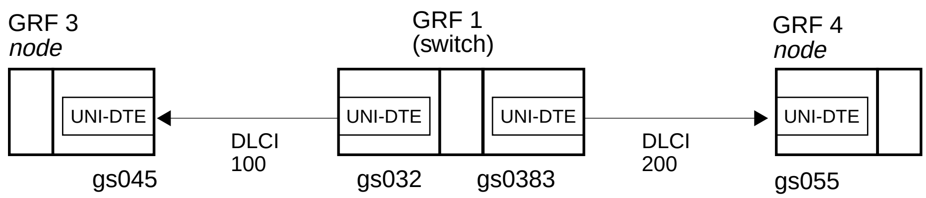


Figure 15-13. Asymmetrical traffic shape example

1 Configure the switch circuit that connects GRF 3 and GRF 4 in the PVCS section of `/etc/grifconfig.conf`:

```
#     EPA      EPB      Optional Parameters
#     ===     ====     ===================
pvcs 4:0:100  5:0:200  Name="grf3-grf4" CIR=64000 Bc=64000 Be=2400
```

2 Configure the endpoint circuit that sends to GRF 3 in the PVCEP section of `/etc/grifconfig.conf`:

```
#     EP       CIR         Bc          Be
#     ===      ====        ===         ===
pvcep 4:0:100  CIR=2000000 Bc=2000000 Be=9600
```

3 Configure the logical interface entries in `/etc/grifconfig.conf`:

```
# name address  netmask broad_dest argument
gs045 192.168.0.5 255.255.255.0      # grf 3
gs032 192.168.0.5 255.255.255.0      # grf 1
gs0383 192.168.0.6 255.255.255.0     # grf 1
gs055 192.168.0.7 255.255.255.0      # grf 4
```

4 Install the circuits with **grfr -c ccp -s *slot* -l *port* -i *dlci*** commands:

```
# grfr -c ccp -s 5 -l 0 -i 200
PVC Defined: 5, link 0, dlci 200
# grfr -c ccp -s 4 -l 0 -i 100
PVC Defined: 4, link 0, dlci 100
```

On-the-fly configuration using grfr

Adding links

You can add or delete Frame Relay links, or modify a link parameter, without resetting the media card.

Note: Once the `/etc/grfr.conf` and `/etc/grifconfig.conf` configuration files are created, if you reboot the system or restart the Frame Relay daemon, **fred**, you do not need to invoke the **grfr** commands we have shown in this section. Either of those actions will cause all links and PVCs specified in the configuration files to be automatically created.

This example adds a UNI-DTE link on interface `gs07f0`, DLCI 122.

Here are the steps:

1 In the appropriate Card profile, specify framing protocol, CRC, and clock as required.

2 Configure the logical interface in `/etc/grifconfig.conf`.

Start the UNIX shell and edit `/etc/grifconfig.conf`:

```
super> sh
# vi /etc/grifconfig.conf
# name address  netmask broad_dest argument
#
gs07f0 192.168.3.3 0.0.0.0
```

Save the file and exit **vi**.

3 Edit the `/etc/grfr.conf` file to add the link in the Link section.

```
# vi /etc/grfr.conf
#Slot Port Optional Parameters
#==== ==== ==================
link 7 1 name="new_link" LMIType=AnnexD
```

Add the logical interface 0 as an active PVC:

```
#   lif    DLCI Peer IP Address Optional Parameters
#   ===    ==== =============== ===================
pvc gs07f0  122  0.0.0.0  Name="new_link"
```

Save the file and exit **vi**.

4 Use a **grfr -c ccl** command to create the link on slot 7, port 1.

```
# grfr -c ccl -s 7 -l 1
LINK Defined: Slot 7, link 1
```

Use a **grfr** display command to check the status of the new link:

```
# grfr -c dlc
```

```
C O N F I G U R E D   L I N K S :
=================================
Name:    S/P:  LMI:     Link:    Autogrif: N391: N392: N393: T391: T392: S:
----     ---   ---      ----     --------  ----  ----  ----  ----  ----  --
new_lnk  7/1  ANNEX-D  UNI-DTE  None       6     3     4     10     15   Ae

Total: 1 links configured
```

Configuring a PVC on-the-fly

You can add or delete PVCs without resetting the media card by editing the `/etc/grfr.conf` file and then using **grfr -c ccp** to create and install the PVC or **grfr -c crp** to remove the PVC.

To add a PVC to the HSSI card in slot 13, start the UNIX shell and edit `/etc/grfr.conf`.

```
super> sh
# vi /etc/grfr.conf
```

Then make the PVC entry as usual:

```
#   lif    DLCI Peer IP Address Optional Parameters
#   ===    ==== ============== ===================
pvc gs0d0  606  0.0.0.0   Name="test606"
```

Save the file and exit **vi**.

To add any additional interfaces, you must edit `/etc/grifconfig.conf`.

Use the **grfr -c ccp** command to create a new PVC. The slot, link and DLCI must be specified:

```
# grfr -c ccp -s 13 -l 0 -i 606
```

Here is the response:

```
PVC Defined: 13, link 0, dlci 606
```

To delete a PVC, you do not need to edit the `/etc/grfr.conf` file, the **grfr -c crp** command is sufficient. Specify the target slot, link and DLCI number to be deleted:

```
# grfr -c crp -s 13 -l 0 -i 600
```

Here is the response:

```
PVC Deleted: 13, link 0, dlci 600
```

Assigning multiple route PVCs to an interface

DLCIs map point-to-point. One DLCI maps a unique circuit between two endpoints, and so only one destination can be assigned on a given DLCI.

The 0.0.0.0 notation is treated specially in that it says instead of hard-coding the ARP entry for the other end of the circuit, obtain it by sending an inverse ARP to the other end and see what comes back.

If the peer IP addresses are in the same subnet, you can assign multiple DLCIs to the interface. This is an example from the PVC section of `/etc/grfr.conf`:

```
#    lif    DLCI  Peer IP address
PVC  gs047e  405  222.222.10.5
PVC  gs047e  406  222.222.10.6
PVC  gs047e  407  222.222.10.7
PVC  gs047e  408  222.222.10.8
```

If the peer IP addresses are in different subnets, you need multiple interfaces:

```
#    lif   DLCI  Peer IP address
PVC  gs040  405  222.222.10.5
PVC  gs041  406  222.222.11.5
PVC  gs042  407  222.222.12.5
PVC  gs043  408  222.222.13.5
```

Verifying and monitoring Frame Relay

The **grfr** display commands return system and interface levels of information that can help you review Frame Relay configurations and monitor the circuit statistics.

Frame Relay system statistics

The **grfr -c dsc** command displays the system-wide Frame Relay configuration.

```
S Y S T E M   P A R A M E T E R S:
===================================

Name:......................... X
Time and Date compiled ........ Fri May  8 02:38:22 CDT 1998
Compiled from source in ....... /nit/A1_4_8R_1/BSDI/usr.sbin/fred
Start Time .................... Fri May  8 10:40:47 CDT 1998
Up-time ....................... 0 days, 4 hours, 3 mins, 40 secs
Configuration File ............ /etc/grfr.conf
grif Configuration File ....... /etc/grifconfig.conf
Debug Level.................... 1
Statistics Interval............ 10
Portcard Heartbeat Interval.... 10
Media Types Supported.......... HSSI, SONET-OC3,
Boards configured ............. 4
Links  configured ............. 4
PVCs  configured .............. 8
   Routed PVCs  configured .... 4
   Switched PVCs  configured .. 4
   Mcasted  PVCs  configured .. 0
   ATMP PVCs configured ....... 0
Active Links .................. XX
Active PVCs ................... XX
```

Active link and PVC data are not available using this command option.

PVC list

The **grfr -c -dpc** command displays a list of configured PVCs and their parameters.

```
C O N F I G U R E D   P V C s   :
=================================
(A* = Autoadded, D* = Deleted)
Name     Slot  Port DLCI  Type   CIR   Bc    Be    State    EPs/ISIS
----     ---   ---- ----  ----   ----  ---   ---   -----    --------
13:0:0    13   0    0     Switch  56K   56K   56K   Active   13:0:0
M1-Group  13   0    311   Mcast-R 56K   56K   56K   Active   13:0:312
                                                             13:0:313
                                                             13:0:314
M1-Group  13   0    312   Mcast-L 56K   56K   56K   Active   13:0:311
```

```
M1-Group  13   0   313    Mcast-L 56K   56K   56K   Active   13:0:311
M1-Group  13   0   314    Mcast-L 56K   56K   56K   Active   13:0:311
Circ-1    13   0   600    Route   56K   56K   56K   Active   ISIS
Circ-2    13   0   601    Route   56K   56K   56K   Active   ISIS
Circ-3    13   0   602    Route   56K   56K   56K   Active   NO-ISIS
Circ-4    13   0   603    Route   56K   56K   56K   Active   NO-ISIS
Circ-5    13   0   604    Route   56K   56K   56K   Inact    NO-ISIS
atmp-1    13   0   609    ATMP    56K   56K   56K   Active

Total  10 PVCs configured
       5 Routed PVCs
       1 Switched PVCs
       4 Multicast PVCs
       1 ATMP PVCs
```

PVC statistics

The **grfr -c dps** command displays PVC statistics. The Transmit Packet and Transmit Octet Dropped columns are not shown in the example below:

```
C O N F I G U R E D   P V C s   S T A T s:
=========================================
(S=Slot, P=Port, R=receive, T=Transmit)
(TP=Transmitted Packets, TO=Transmitted Octets)

Name     S/P/DLCI   Type   R-Packets  R-Octets  T-Packets  T-Octets
----     --------   ----   ---------  --------   ---------  --------
0:0:0     00:0:0     Switch  0        0         0           0
0:1:0     00:1:0     Switch  0        0         0           0
1:0:0     01:0:0     Switch  63925    1001710   63926       948134
south-0   01:0:100   Route   44       3872      41         3608
lunar     01:0:101   Route   0        0         0           0
1:1:0     01:1:0     Switch  0        0         0           0
south-1   01:1:16    Route   0        0         0           0
regulu    01:1:101   Route   0        0         0           0
south-2   01:1:102   Route   0        0         0           0
```

Link configuration

The **grfr -c dlc** command displays link configuration. The Status column is partially shown.

```
C O N F I G U R E D   L I N K S :
=================================
Name: S/P:  LMI:   Link:   Autogrif: N391: N392: N393: T391: T392: Stat
----  ---   ---    ----   --------  ----  ----  ----  ----  ----  -----
jan0  1 /0  ANNEX-D  NNI      None    6     3     4     10    15  Active
acme  1 /1  ANNEX-D  UNI-DTE  None    6     3     4     10    15  Active
Jan   2 /0  ANNEX-D  UNI-DCE  None    6     3     4     10    15  Inacti
mike  2 /1  ANNEX-A  UNI-DCE  None    6     3     4     10    15  Inacti

Total: 4 links configured
```

Collect data via grdinfo

Refer to the “Management Commands and Tools” chapter for information about using the **grdinfo** tool to collect Frame Relay statistics and configuration data.

With a single command, **grdinfo** collects the output from system configuration and status, board status, link configuration and status, PVC statistics, PVC configuration and status, and interface configuration and status, and compresses it in a log file.

grfr command set

Display commands

The **grfr** command has display options that return useful information about Frame Relay links. Display commands are prefaced with the **-c** flag and begin with the letter **d**:

-c dsc, display system configuration and status

-c dlc, display link configuration and status

-c dpc, display PVC configuration and status

-c dic, display interface configuration and status

-c dss, display system status

-c dls, display link status

-c dps, display PVC statistics

-c dbs, display board status

Link configuration commands

Link configuration commands are prefaced with the **-c** flag and begin with the letter **c**:

-c cel, enable a link, must specify slot and port.
Example: enable a link on slot 3, port 1 `# grfr -c cel -s 3 -l 1`

-**c cdl,** disable a link, must specify slot and port.
Example: disable a link on slot 3, port 0 `# grfr -c cdl -s 3 -l 0`

-**c ccl,** configure a new link, must specify slot and port.
If the specified link is already configured, the link will be configured as specified in the `/etc/grfr.conf` file.
Example: configure a link on slot 5, port 0 `# grfr -c ccl -s 5 -l 0`

-**c crl,** remove a link and all underlying PVCs, must specify slot and port.
Example: remove the link on slot 5, port 0 `# grfr -c crl -s 5 -l 0`

PVC configuration commands

PVC configuration commands are prefaced with the **-c** flag and begin with the letter **c**:

-c cep, enable a PVC, must specify slot, port, and DLCI (-s, -l, -i).
Example: enable a PVC in slot 3, port 0 `# grfr -c cep -s 3 -l 0 -i 888`

-c cdp, disable a PVC, must specify slot, port, and DLCI (-s, -l, -i).
Example: disable a PVC in slot 3, port 0 `# grfr -c cdp -s 3 -l 0 -i 888`

-c ccp, configure a new PVC, must specify slot, port, and DLCI (-s, -l, -i).

Example: create a new PVC in slot 3, port 0 `# grfr -c ccp -s 3 -l 0 -i 888`

-c crp, remove a PVC, must specify slot, port, and DLCI (-s, -l, -i).

Example: remove a PVC in slot 3, port 0 `# grfr -c crp -s 3 -l 0 -i 888`

Debug commands

-c ddl, display debug level.

Example: `# grfr -c ddl`

-c csd, set debug level, requires **-d** option to specify level 0–4.

Example: `# grfr -c csd -d 3`

0 reports the least amount of debug information, 4 the highest.

Refer to the *GRF Reference Guide* for more information about the **grfr** command.

States of configured PVCs

Some **grfr** commands return state information, these are the current state options:

Active

An active PVC is correctly configured on both endpoints and the circuit is up.

Inactive

An inactive PVC is correctly configured on both endpoints, but the circuit is not up. If all the PVCs on a port show inactive, the cable could be the problem. If only one is reported inactive, it is likely that the endpoint PVC is down.

Deleted

This state is assigned if the configuration exists on the GRF endpoint but is not configured from the remote endpoint.

Disabled

This is a user-initiated state (via **grfr**) that keeps the configuration information in place but does not let the circuit activate. May also be used when setting an on-the-fly configuration via **grfr**.

Enabled

This is a user-initiated state (via **grfr**) that activates a pre-configured circuit. May also be used when doing an on-the-fly configuration via **grfr**.

tcpdump

You can use the **tcpdump** utility when analyzing routed Frame Relay circuits.

Integrated Services: Controlled-Load 16

Chapter 16 describes the initial GRF implementation of Integrated Services, the provision of Controlled-Load services on ATM OC-3c, Ethernet, FDDI, SONET, and HSSI media cards.

Chapter 16 contains these topics:

Overview

Integrated Services is the IETF name for features that allow network resources to be reserved for specific applications.

Resource reservations can give applications guaranteed bandwidth and delay bounds from the network. The IETF Integrated Services Working Group has defined several kinds of service that can be requested from the network, for example, Controlled-Load Service and Guaranteed Service. This is not a complete implementation of Integrated Services, only Controlled-Load service is implemented. Other service types, including Guaranteed, are not implemented.

Controlled-Load is defined by the IETF Integrated Services working group (`draft-ietf-intserv-ctrl-load-svc-04.txt`). In the draft, its end-to-end behavior is characterized by:

- A very high percentage of transmitted packets will be successfully delivered by the network to the receiving end-nodes.

  (The percentage of packets not successfully delivered must closely approximate the basic packet error rate of the transmission medium).
- The transit delay experienced by a very high percentage of the delivered packets will not greatly exceed the minimum transmit delay experienced by any successfully delivered packet.

  (This minimum transit delay includes speed-of-light delay plus the fixed processing time in routers and other communications devices along the path.)

Controlled-Load implementation

Controlled-Load is implemented on GRF media cards that support Selective Packet Discard; ATM OC-3c, Ethernet, FDDI, SONET, and HSSI.

Controlled-Load does not affect queuing, only discarding. The identification of which packets to select as high precedence is based on a filter bound to a logical interface.

Controlled-Load allows the identification of certain packets as high precedence. This identification is done through filters. Filters provide flexibility for targeting source, destination, protocol, port, and combinations of these criteria. The difference with class filtering is that instead of filtering out matches to the criteria, the filter marks that packet as high precedence in the GRIEF header.

Any criteria you can define a filter to detect can be assigned high precedence in the GRIEF header before the packet is sent across the switch to the transmitting media card.

The GRF delivers Controlled-Load service to a specific flow by marking its packets to prevent Selective Packet Discard (SPD). The marking mechanism uses filters to identify the packets belonging to the applications for which resources are reserved. Filters can be manually configured by adding them to `/etc/filterd.conf`.

The GRF implementation does not place Controlled-Load traffic in separate output queues from other traffic; all traffic is FIFO-queued. As a result, while Controlled-Load traffic will see minimal packet loss, it may see more than minimal delay.

Controlled-Load protects packets that match the filter from being lost. Packets that match the filter are marked so they will not be dropped by SPD. SPD drops packets that are not marked when the number of free buffers gets too low. Dynamic routing packets are marked. The class filter is another way of setting the same bit in the packet header.

Filters

The filter syntax is the same as for any filter. The `class` *`class_value`* option is added to the available filtering actions to support Controlled-Load. Use any integer for *`class_value`*, the value is actually ignored, the `action class` specification marks a filter-matching interface to receive Controlled-Load service.

The filter and action syntax are like this:

```
media <media type> <slot>   {
        bind <filter name>  {
                vlif <physical port>;
                direction in;
                action class <class_value>;
        }
}
```

The action class means that packets which match the applied filter will receive Controlled-Load service.

Filters can be applied to applications such as GateD, or to all packets coming from a given source or source network, or all packets to a given destination. The filter gives marked packets resources that could otherwise be unavailable or limited. In terms of GRF architecture, these resources are data buffers.

Filter examples

This Controlled-Load filter is applied to the flow of packets coming in to the `gs021` interface on the HSSI card in GRF slot 2:

```
media HSSI 2   {
        bind controlled_load  {
                vlif 1;
                direction in;
                action class 22;
```

Introduction to Subnetting

A

The GRF supports variable-length subnet masking. Appendix A describes these masks, and how they are used to improve the efficiency of routing.

Appendix A contains these topics:

What is subnetting?

This appendix provide a brief overview of class-based addresses and the evolution of netmasking from implicit (fixed) to explicit (variable).

Note that not all routing protocols support subnetting and supernetting. RIPv1 does not support subnetting and supernetting. Also, GateD v2.0 does not support RIPv2 or OSPF.

Routing protocols RIPv2, OSPF, and BGP4, and explicit static routes, do support netmasks and classless addressing.

Early implementation of classes and implicit masks

Delivering an IP datagram to a network station requires that the originator and each intervening "hop" (a hop is a host or router the datagram passes through enroute to its destination) have routing information to determine where best to direct the packet.

In most cases, this information is the network number of the destination and a 32-bit mask called the netmask that is used to determine whether the destination address is a part of that network.

At its inception, the 32-bit Internet Protocol (version 4) address space was segmented into separate classes that designers assumed would make full use of all available address space and meet the demand for addresses.

Classes were specified in 32-bit words as shown in Figure A-1:

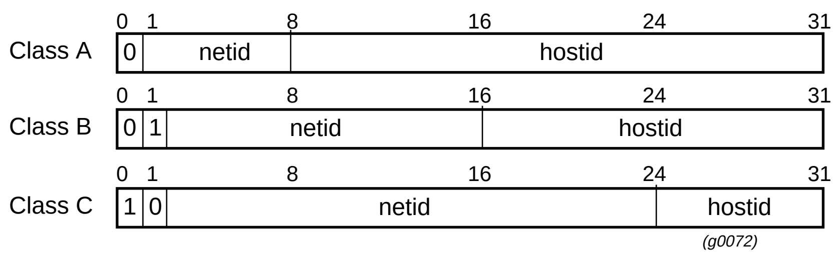


Figure A-1. Specification of classes in IP addresses

As such, "class-based" addresses were in the range:

- Class A: `1-127.0.0.0`
- Class B: `128-191.0.0.0`
- Class C: `192-254.0.0.0`

Netmask information was implicitly determined based upon the class of the address:

Class:	Mask:
`A`	`255.0.0.0`
`B`	`255.255.0.0`
`C`	`255.255.255.0`

To a router, the netmask was implicit in that all addresses in class A had a netmask of `255.0.0.0`, all in class B had a netmask of `255.255.0.0`, and so on. Netmasks were not part of a route table.

When administrators of the Internet address space assigned new participants unique addresses, the recipient could only manipulate the part of the address that, when logically-ANDed with the mask, resulted in 0.

In reality, most organizations fell into the category of needing a class B address. Later, it became clear that the class B address space was beginning to be exhausted (which would effectively end the life of the current address scheme), while A and C class addresses were comparatively untouched.

Classless inter-domain routing (CIDR)

The main difficulty in class-based addresses is in the rigid structure of netmasks. A strategy lending itself to solving the problems both of address space and global route table size was to eliminate the implicit nature of netmasks. In effect, this change also eliminated class-based routing.

One part of the strategy is to no longer key the netmask from the address class, but rather to explicitly provide a mask for each assigned network.

Using the explicit mask shown below assigns 2^16th bits of address space to the end user and creates only *one* network route at the backbone level. This is commonly known as "Classless Inter-Domain Routing" (CIDR).

Address: `198.224.0.0` *Class-based mask:* `255.255.255.0`

Classless mask: `255.255.0.0`

In the next logical extension of the strategy, netmasks are not required to end on an 8-bit boundary within an address. To provide the user with 2^18th bits of address space, the following mask is assigned:

`198.224.64.0 / 255.255.192.0`

This is commonly known as "variable-length subnet masking".

Supernetting: benefits for routing

Strategies for masking network addresses provide interesting possibilities for implementing efficient routing. If addresses are assigned properly, the number of routes required to reach any part of any large network from any other point in the large network can be made extremely small.

One such strategy is called “supernetting”. In supernetting, networks with addresses of identical prefix can be stored using only one route-table entry at the upstream router.

Figure A-2 illustrates an example in which supernetting is applied:

A single GRF handles routing between two buildings:

This address / mask allocates the address space that will
be used to address machines in the two buildings: `198.224.0.0 / 255.255.0.0`

Network addresses in building A are assigned in the range: `198.224.0-127.0`
and addresses in building B are assigned in the range: `198.224.128-254.0`

The GRF needs only TWO entries in its route table to correctly route packets:

Net address	Netmask	Destination
198.224.0.0	255.255.128.0	Building A
198.224.128.0	255.255.128.0	Building B

(g0074)

Figure A-2. Basic supernetting example

In class-based addressing, this internetwork would contain 256 Class C networks, and would require 254 separate route table entries. Maintaining large tables uses additional memory and is less efficient to query.

The supernet strategy can be recursively applied until the address space is exhausted, that is, until you subnet mask to 255.255.255.255.

For example, if building B had two floors and a second GRF routing between those two floors, the netmask could be extended to:

`255.255.224.0`

Addresses on floor 1 could be: `198.224.128.0/255.255.224.0`

while floor 2 could be: `198.224.192.0/255.255.224.0`

Supernetting allows the network administrator to apportion an assigned address block into smaller network (or host) segments in sizes based on powers of two (2, 4, 8, 16 networks, for example).

A router does not communicate these segments to peers that are higher in the routing topography tree. Upward peers need only a single route to the entire subordinate address block. When implemented properly, supernetting results in significantly smaller and more efficient route tables.

Support for explicit netmasks

Changes in how routing decisions are performed have only occurred conceptually.

In practice, the same algorithms are applied to the route table data and the destination IP address. The only change is that a netmask must be explicitly provided with routing information.

Some dynamic routing protocols (OSPF, BGP4, etc.) exchange netmask information with route table updates, other protocols do not. Check your dynamic routing agent documentation for more information.

Deriving a supernet address

A supernet address is derived by logically ANDing an IP address with the netmask assigned to the net.

When the router receives an IP address, the assigned netmask is ANDed to it and the supernet address is produced. The route table is searched for the resulting supernet address and the packet is then forwarded to the destination at that address.

Example 1

Destination host IP address:	`139.51.21.48`
Destination netmask:	`255.255.0.0`
Supernet address:	`139.51.0.0`

Example 2

Destination host IP address:	`137.66.12.48`
Destination netmask:	`255.255.255.0`
Supernet address:	`137.66.12.0`

Example 3

Destination host IP address:	`137.66.12.48`
Destination netmask:	`255.255.0.0`
Supernet address:	`137.66.0.0`

A supernet routing example

This section uses a typical routing situation as an example of how to use netmasks and set up supernets.

In the example, an Internet service provider is allocated a CIDR block of IP addresses: 256 “Class C” networks starting at `192.24.`.

The service provider intends to distribute smaller blocks of addresses to a set of customers named “A” through “F”. A GRF router is assigned to handle all routing.

The address assignments are as follows:

Customer	Subnet Numbers		Number of routes
A	`192.24.0`	through `7`	8
B	`192.24.8`	through `11`	4
C	`192.24.12`	through `15`	4
D	`192.24.16`	through `31`	16
E	`192.24.32`	through `33`	2
F	`192.24.34`	through `35`	2

The GRF needs to correctly direct packets destined for any host on any of these networks. There are two ways to list these routes in the route table. One method results in a route table with 36 entries, the other in a route table with six entries.

Example 1: Traditional route storage method

Figure A-3 shows the first method. A traditional route table stores one route for each subnet, requiring 36 entries in the route table:

– entries 0 –7 point to customer A

– entries 8 –1 point to customer B

– entries 12–15 point to customer C, and so on.

Entry number	Subnet address	Netmask	Customer destination
1	192.24.0.0	255.255.255.0	A
2	192.24.1.0	255.255.255.0	A
3	192.24.2.0	255.255.255.0	A
4	192.24.3.0	255.255.255.0	A
• • •	• • •	• • •	• • •
8	192.24.7.0	255.255.255.0	A
9	192.24.8.0	255.255.255.0	B
• • •	• • •	• • •	• • •
12	192.24.11.0	255.255.255.0	B
13	192.24.12.0	255.255.255.0	C
• • •	• • •	• • •	• • •
36	192.24.35.0	255.255.255.0	F

(g0075)

Figure A-3. Example 1, a traditional route table with one entry per subnet

This method works because the GRF is given the routing information it requires. The drawback is that this method sets up a large route table that has to contain each individually-assigned network.

The search resources required for large route tables negatively affect routing performance and efficiency.

Example 2: Subnet mask storage method

The second method stores one route for each *block* of customer addresses. A variable netmask defines the *range* of the destination addresses for each customer. In our example, the set of customer nets can be defined as:

Customer	Range of nets (blocks)	Netmask (variable)	(0x 0x 0b 0x)
A	192.24.0-7	255.255.248.0	(ff.ff.11111000.00)
B	192.24.8-11	255.255.252.0	(ff.ff.11111100.00)
C	192.24.12-15	255.255.252.0	(ff.ff.11111100.00)
D	192.24.16-31	255.255.240.0	(ff.ff.11110000.00)
E	192.24.32-33	255.255.254.0	(ff.ff.11111110.00)
F	192.24.34-35	255.255.254.0	(ff.ff.11111110.00)

In the case of customer A, the netmask `255.255.248` specifies that the first 21 bits of the address is the net address. Because of the way the blocks are allocated, that supernet address is unique, and can be used as a routing key for each subnet in that block.

The same is true for each customer B through F.   For each customer A–F, the address block can again be subdivided using another set of masks.

Forming a supernet address

To derive a supernet address, the IP address is ANDed with the netmask.

Here are examples from the address block allocations given above:

Supernet derivation 1:

```
192.24.9.3     =  11000000  .00011000  .00001001  .00000011
mask           =  11111111  .11111111  .11111100  .00000000
supernet       =  11000000  .00011000  .00001000  .00000000
               =  192        24         8          0
```

Supernet derivation 2:

```
192.24.13.131  =  11000000  .00011000  .00001101  .10000011
mask           =  11111111  .11111111  .11111100  .00000000
supernet       =  11000000  .00011000  .00001100  .00000000
               =  192        24         12         0
```

Supernet derivation 3:

```
192.24.27.131  =  11000000  .00011000  .00011011  .10000011
mask           =  11111111  .11111111  .11110000  .00000000
supernet       =  11000000  .00011000  .00010000  .00000000
               =  192        24         16         0
```

When the masks are applied to the remaining customer addresses, a list of customer supernet addresses is obtained:

Customer	Supernet Address
A	192.24.0
B	192.24.8
C	192.24.12
D	192.24.16
E	192.24.32
F	192.24.34

Entry number	Net address (supernetted)	Netmask	Customer destination
1	192.24.0	255.255.248.0	A
2	192.24.8	255.255.252.0	B
3	192.24.12	255.255.252.0	C
4	192.24.16	255.255.240.0	D
5	192.24.32	255.255.254.0	E
6	192.24.34	255.255.254.0	F

(g0076)

Figure A-4. Example 2: a route table with supernetting applied

Only one supernet address for each block of addresses needs to be in the route table. In our example, Figure A-4, supernets reduce the size of the route table from 36 to six entries. Router storage space and search times are minimized.

How the GRF uses a mask

This is how the GRF processes IP addresses using subnet masks:

1 The system route table is created by any or all of:

- a network administrator
- a dynamic routing agent
- by activating an interface

2 When it receives an IP packet, the GRF examines the IP header for correctness and extracts the destination IP address.

3 The GRF checks its route table to see if a route to that destination is present by comparing the received destination address against the table entries.

The comparison involves:

- walking down the route table tree (a tree data structure is used to store entries) using the destination address as a key
- as potential matches are encountered, the GRF first does an address-to-mask bitwise comparison, obtains a result, and then does a result-to-address bitwise comparison.

  In the first step, the GRF logically ANDs the destination address to the mask accompanying the entry.

  In the second step, the result from the first step is compared bit-for-bit to the supernet address at the entry.
- if a single match is made, the packet is forwarded to the found address

  For details about matching and longest match, see the *Rules for matching* and *Longest match example* sections in this chapter.

Routing look-up example

This example is discussed in the next several sections.

A packet must be routed to: `192.24.14.30`, the GRF route table looks like this:

Net address (supernetted)	Netmask (binary)	Destination address	
`192.24.0`	`ff.ff.11111000.00)`	`102.24.aaa.aaa`	`(A)`
`192.24.8`	`ff.ff.11111100.00)`	`102.24.bbb.bbb`	`(B)`
`192.24.12`	`ff.ff.11111100.00)`	`102.24.ccc.ccc`	`(C)`
`192.24.16`	`ff.ff.11110000.00)`	`102.24.ddd.ddd`	`(D)`
`192.24.32`	`ff.ff.11111110.00)`	`102.24.eee.eee`	`(E)`

Address-to-mask logical ANDing

The first two octets of the net (supernet) addresses (`192.24`) and the netmasks (`ff.ff`) are identical, and are ignored to simplify the routing look-up example.

Beginning with the 3rd octet, the binary representation of each supernet address is:

```
0   =  0000 0000
8   =  0000 1000
12  =  0000 1100
16  =  0001 0000
32  =  0010 0000
```

The 3rd octet in the address the GRF is trying to route is 14:

```
14  =  0000 1110
```

As shown in Figure A-5, the router performs a left-to-right bitwise comparison of bits the length of the netmask between the netmask (top line) and the corresponding bits in the destination address.

```
Supernet 192.24.0          0    =   1111 1000   <-- Netmask
                     AND  14    =   0000 1110   <-- Destination address
                                    ---------
                                    0000 1000   <-- Result

Supernet 192.24.8          8    =   1111 1100
                     AND  14    =   0000 1110
                                    ---------
                                    0000 1100

Supernet 192.24.12        12    =   1111 1100
                     AND  14    =   0000 1110
                                    ---------
                                    0000 1100

Supernet 192.24.16        16    =   1111 0000
                     AND  14    =   0000 1110
                                    ---------
                                    0000 0000

                          32    =   1111 1110
Supernet 192.24.32   AND  14    =   0000 1110
                                    ---------
                                    0000 1110
                                                          (g0077)
```

Figure A-5. Routing logic: ANDing destination address to the subnet mask

Result-to-address comparison

Next, the router performs a left-to-right bitwise comparison of each entry's supernet address (top line) against the corresponding bits in the results from the logical AND.

```
Supernet address -->      0   =   0000 0000  } Bits fail to match at position 5  (from left)
Result from logical AND -->   =   0000 1000  } NOT a candidate

                          8   =   0000 1000  } Bits fail to match at position 6  (from left)
                              =   0000 1100  } NOT a candidate

                         12   =   0000 1100  } Bits match in all masked positions !
                              =   0000 1100  } CANDIDATE

                         16   =   0001 0000  } Bits fail to match at position 4  (from left)
                              =   0000 0000  } NOT a candidate

                         32   =   0010 0000  } Bits fail to match at position 3  (from left)
                              =   0000 1110  } NOT a candidate
                                                          (g0078)
```

Figure A-6. Bit-by-bit comparison to the supernet address

The router determines the destination supernet address to be `192.24.12`. A route table lookup is made, the destination is found to be C (`192.24.ccc.ccc`), and the packet is handed off.

Rules for matching

1 A match is attempted using the result of the routing logic (logical ANDs) and the supernet address.

2 In order to match, all bits the length of the mask (beginning with the first octet) must match.

Bits beyond the length of the netmask are not used for comparison.

In this case, the match fails at the final 1 in the address:

subnet mask: `0000 0000.0000 0000.0001 1000`
address ANDed: `0000 0000.0000 0000.0001 1100`

3 The "longest match" is taken.

"Longest match" means more bits match. In this case, more implies a "length" of adjacent bits ranged in an octet. An example follows.

Longest match example

There are two entries in the route table in this order:

```
198.174.128.0   / 255.255.255.0     -->  target 1
198.174.128.42  / 255.255.255.255   -->  target 2
```

A packet must be routed to: `198.174.128.42`

When the router logically ANDs the target 1 netmask with the destination address, the result is a match:

```
   198.174.128.42
   255.255.255.0
=  198.174.128.0
```
(17 contiguous bits match)

The router performs the same routing logic to target 2. It *also* matches, but this match is:

```
   198.174.128.42
   255.255.255.255
=  198.174.128.42
```
(28 contiguous bits match)

which is the *longer* match since bits in the 4th octet also match.

Warranty

B

Appendix B contains warranty information for the following Ascend products:

- GRF 400 (GRF-4-AC)
- GRF 400 (GRF-4-DC)
- GRF 1600 (GRF-16-AC)
- GRF 1600 (GRF-16-DC)

Product warranty

1. Ascend Communications, Inc. warrants that the GRF 400, GRF 1600, and GR-II will be free from defects in material and workmanship for a period of twelve (12) months from date of shipment.
2. Ascend Communications, Inc. shall incur no liability under this warranty if
   - the allegedly defective goods are not returned prepaid to Ascend Communications, Inc. within thirty (30) days of the discovery of the alleged defect and in accordance with Ascend Communications, Inc.'s repair procedures; or
   - Ascend Communications, Inc.'s tests disclose that the alleged defect is not due to defects in material or workmanship.
3. Ascend Communications, Inc.'s liability shall be limited to either repair or replacement of the defective goods, at Ascend Communications, Inc.'s option.
4. Ascend Communications, Inc. MAKES NO EXPRESS OR IMPLIED WARRANTIES REGARDING THE QUALITY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE BEYOND THOSE THAT APPEAR IN THE APPLICABLE Ascend Communications, Inc. USER'S DOCUMENTATION. Ascend Communications, Inc. SHALL NOT BE RESPONSIBLE FOR CONSEQUENTIAL, INCIDENTAL, OR PUNITIVE DAMAGE, INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR DAMAGES TO BUSINESS OR BUSINESS RELATIONS. THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES.

Warranty repair

1. During the first three (3) months of ownership, Ascend Communications, Inc. will repair or replace a defective product covered under warranty within twenty-four (24) hours of receipt of the product. During the fourth (4th) through twelfth (12th) months of ownership, Ascend Communications, Inc. will repair or replace a defective product covered under warranty within ten (10) days of receipt of the product. The warranty period for the replaced product shall be ninety (90) days or the remainder of the warranty period of the

original unit, whichever is greater. Ascend Communications, Inc. will ship surface freight. Expedited freight is at customer's expense.

2 The customer must return the defective product to Ascend Communications, Inc. within fourteen (14) days after the request for replacement. If the defective product is not returned within this time period, Ascend Communications, Inc. will bill the customer for the product at list price.

Out-of warranty repair

Ascend Communications, Inc. will either repair or, at its option, replace a defective product not covered under warranty within ten (10) working days of its receipt. Repair charges are available from the Repair Facility upon request. The warranty on a serviced product is thirty (30) days measured from date of service. Out-of-warranty repair charges are based upon the prices in effect at the time of return.

Configuration tracking via CFMS

C

Appendix C describes a new GRF application, the Configuration File Management System (CFMS).

CFMS enables network administrators to keep local copies and histories of configuration files for multiple remote GRF routers. Using CFMS, administrators can review, edit, and track changes to those configuration files from their management workstation.

Appendix C includes these topics:

Note: When you download the CFMS binary, please be sure to download the accompanying README file. It contains additional information you will find helpful.

Introduction to CFMS

The Configuration File Management System (CFMS) enables network administrators to monitor and manage the configuration files for multiple remote GRF routers. CFMS builds a CVS tree structure that contains local copies of the files and change histories when files are changed.

Using CFMS commands, administrators can review, edit, and track changes to those configuration files from their management workstation. CFMS runs on a UNIX server attached to client routers via a secure administrative LAN.

Here is a diagram of a typical CFMS installation:

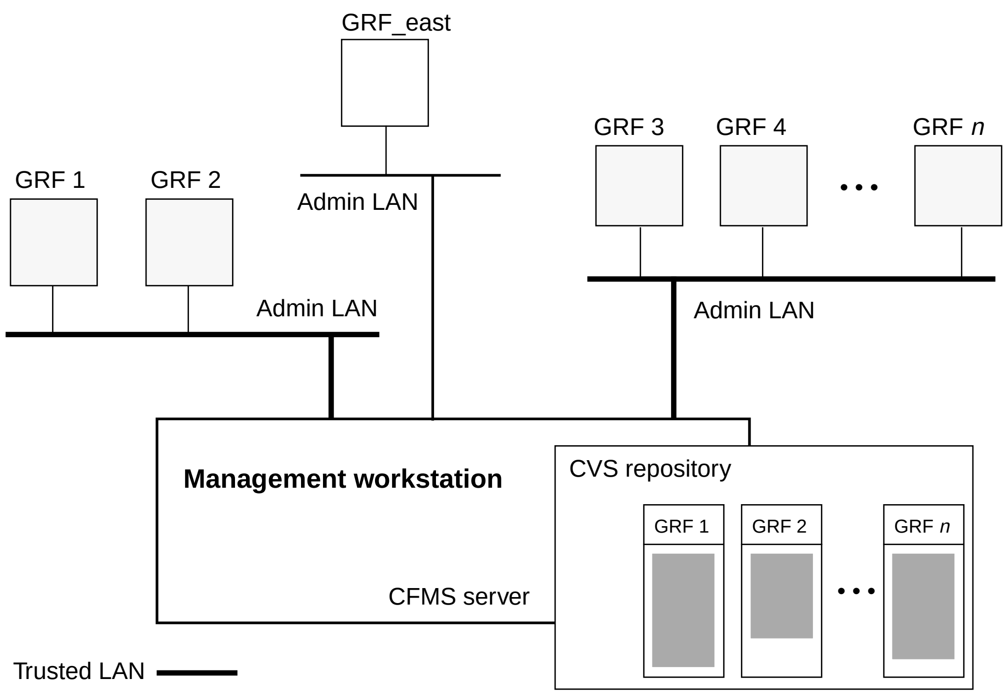


Figure C-1. Diagram of CFMS server and client GRF routers

The diagram shows a CFMS server supporting GRFs on multiple administrative LANs. CFMS is intended to be used from a server on the administrative LAN connected to one or more routers. To protect against unauthorized access to routers on the network, CFMS must be run on a trusted LAN. In the example above, the connecting LAN is not secure and GRF_east cannot come under CFMS management.

Space availability on the server hosting the CFMS application determines how many routers can be managed. CFMS does not otherwise limit the number of client routers.

CFMS uses **expect** and **telnet** to move configuration files between the server and managed routers, and maintains a CVS repository on the CFMS server of configuration files for all routers under CFMS management.

Capabilities

CFMS performs the following functions:

- generates access and “collection” paths to put remote routers under management
- removes routers from managed category
- collects the default set of configuration (`.conf`) files from the `/etc` directory of each managed router or a user-specified set of files (ascii) located anywhere in the router
- maintains the collected files in a CVS tree structure that enables revision and other file attributes to be tracked
- maintains a log of configuration file changes for archival and audit purposes
- enables the administrator to edit a copy of a specific configuration file, then places the new file version to the target router’ own `/etc` directory

Note: CFMS cannot remotely activate the new configuration file on the target GRF. The administrator must log in to that router to activate the configuration change.

Security

Router passwords are stored on the server on which CFMS is run. CFMS does not encrypt the passwords it uses, and transmits passwords in clear text when it accesses managed routers. To protect against unauthorized access to routers on the network, CFMS must only be run on a trusted LAN.

Server requirements

Disk space

The base CFMS application requires 50Kb. Each router requires approximately 200Kb for its CVS repository space.

The CVS repository is typically located in the same directory where CFMS is installed, but you can specify that it be located elsewhere.

Software

The CFMS server must be running or have resident the following software:

- **CVS** version 1.8.1 or later
- **perl** version 5.003 or later
- **ksh**
- **make**
- **expect** version 5.22.0 or later
- Solaris version 5.4 or later, BSDI/OS 2.1 or later
  This feature will work on these OS levels and may also work on Solaris 5.5, Solaris 5.6, and BSDI/OS 3.0.

GRF requirements

The GRF router must have the following software installed:

- GRF software release 1.4.8 or later
- `/usr/bin/uudecode`
- `/usr/bin/uuencode`
- `/bin/tar`
- `/usr/contrib/bin/gzip`

The `user` and `root` accounts should operate properly with the default shell. These accounts should have **tar** and **uuencode**/**uudecode** in the command search path.

CFMS has no storage requirements on the GRF.

Logging

Logging is done to the `cfms-trace.log` file. This file is created when CFMS starts and is located in the directory where CFMS is installed. It collects error and other messages that are generated from the operations of CFMS commands.

The log file can grow over time as CFMS is used. The system administrator of the CFMS server needs to see that it is periodically cleaned up.

Command set

CFMS has the following command set:

Command	**Function**
cfms add_host	Adds a GRF to CFMS management
cfms remove_host	Removes a GRF from CFMS management
cfms edit	Edits a configuration file that is under CFMS management.
cfms list	Lists the GRFs currently under CFMS management
cfms status	Lists the status of configuration files for GRFs under CFMS.

Sections that appear later in this chapter describe options for each command and provide examples of usage.

CVS repository and commands

A CVS repository is created at the location specified at install time. The files being managed are sorted in a tree with the repository root being what was specified at install time. For example, if you specify the CVS repository to be located at `/home/cfms`, then this forms the root of the CVS tree. The figure below illustrates the installation directory of such a CVS tree.

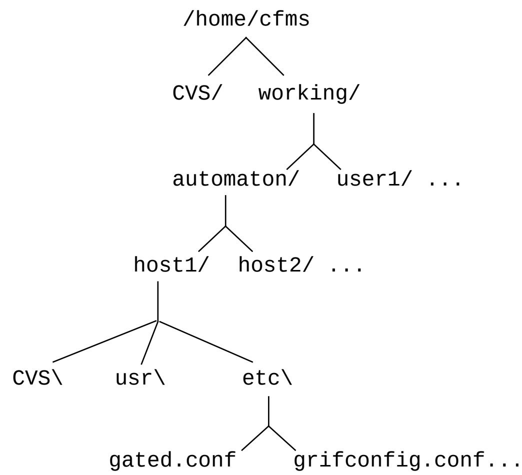


Figure C-2. CVS repository created by CFMS

Use CVS commands to access the information in the CVS repository. Refer to **cvs --help -*command*** for details. A few commonly-used commands are explained here, they are based on the example illustrated above.

Access log information

This command gives you log information about the edit history of the `gated.conf` files on the GRF with a hostname of host1:

```
# cvs log /home/cfms/working/automaton/host1/etc/gated.conf
```

Display file version

This command gives you the current version number of the `gated.conf` file on host1:

```
# cvs status /home/cfms/working/automaton/host1/etc/gated.conf
```

This is the information returned:

```
File: gated.conf        Status: Up-to-date
Working revision:    1.3  Wed Jun 24 18:09:04 1998
Repository revision: 1.3  /home/cfms/CVS/to host1.site.com/gated.conf
Sticky Tag:          (none)
Sticky Date:         (none)
Sticky Options:      (none)
```

Display line-by-line history

This command gives you the history, line-by-line, of each line in host1's `gated.conf` file:

```
# cvs annotate /home/cfms/working/automaton/host1/etc/gated.conf
```

Before you download

This section describes the preparations for CFMS installation.

Two important notes:

Note: CFMS cannot be installed or operated by root. Certain preparation tasks are performed by root:

- reserve disk space for CFMS
- create directory for CFMS installation
- create administrative CFMS user

Note: All CFMS operations and commands must be performed from the CFMS installation directory.

Server tasks

Before you download CFMS, complete these tasks on the CFMS server:

1 Make sure the following software is installed:
- **CVS** (version 1.8.1 or later)
- **perl** (version 5.003 or later)
- **ksh**
- **make**
- **expect** (version 5.22.0 or later)
- Solaris version 5.4 or later, BSDI/OS 2.1 or later
CFMS works on these OS levels and may also work on Solaris 5.5,
Solaris 5.6, and BSDI/OS 3.0.

2 On the intended server, create a user account for the purpose of running CFMS, that is, create a "CFMS user." Designate the CFMS user's home directory as the base for the CFMS installation.

3 Create the directory that you want as the base for the CFMS installation, giving the CFMS user write permission.

GRF tasks

Complete these tasks for each GRF to be managed:

1 Be sure the following software is installed:
- GRF software release 1.4.8 or later
- `/usr/bin/uudecode`
- `/usr/bin/uuencode`
- `/usr/contrib/bin/gzip`
- `/bin/tar`

2 Verify that the `user` and `root` accounts operate properly with the default shell, and that these accounts have **tar** and **uuencode**/**uudecode** in their command search path.

Downloading CFMS

This section describes how to retrieve a copy of CFMS from the Ascend ftp server,

CFMS is released as a compressed TAR file. Included in the TAR file is a README file that also describes how to install and run CFMS.

To retrieve CFMS, perform the following steps:

1 On the intended server, log in as the CFMS user and change to the installation directory. For example, if the directory is `/usr2/local/router`:

```
# cd /usr2/local/router
```

2 Log in as anonymous to this ftp server:

```
# ftp ftp.min.ascend.com
```

3 Then change directory:

```
ftp> cd releases
ftp> cd A1_4_8
```

4 Set the file format and download the CFMS file:

```
ftp> bin
ftp> get CFMS.TAR.gz
ftp> quit
```

5 Unpack the zipped file.

```
# gunzip CFMS.TAR.gz
# tar xvof CFMS.TAR
```

or

```
# gzcat CFMS.TAR.gz | tar xvof
```

The unpacking step creates the subdirectory /`cfms`:

```
# ls /usr2/local/router/cfms
```

The installation subdirectory /`cfms` will contain more than 40 files.

Installation procedure

Exit as root and log in as CFMS user.

1 Check that the **gzip** program is installed in `/usr/local/bin.`

2 If **tclsh** is not installed in a standard location (`/usr/local/bin`), then the TCL_LIBRARY environment variable needs to be set to point to the location where **Tcl** is installed.

If it is installed in `/usr/local/lib/tcl8.0.`, then you need to reset the variable as shown here:

for Korn/Bourne/Sh:

```
# export TCL_LIBRARY=/usr/local/lib/tcl8.0
```

for Csh:

```
% setenv TCL_LIBRARY /usr/local/lib/tcl8.0
% rehash
```

3 Before running any of the **expect** scripts, perform a call to **make**:

```
# make
```

The call builds a file/function index to assure that things work properly. You will only need to re-run **make** if you add or delete functions from a file or if you re-install CFMS.

4 Edit the CFMS `Makefile` and verify that the `TCLSH` and CVSINIT variables point to the correct locations of **tclsh** and **cvsinit** installed on the CFMS server.

5 Run the install script from within the installation directory. Change directory if you are not already there:

```
# cd /cfms
# ./install.pl
```

The script begins a series of prompts and messages:

```
sitebox-4 ./install.pl
What is pathname of perl5 on your system? [/usr/local/bin/perl5]
# Fixing up #! paths in perl scripts
Where is the CFMS root directory? [/usr/local/CFMS] /usr2/local/cfms
/usr2/local/cfms does not exist, so I will create it for you.
Unable to create the directory /usr2/local/cfms.  Please try another
location.
Where is the CFMS root directory? [/usr2/local/cfms] /usr2/local/holly
creating /usr2/local/admin/CVS directory!
creating /usr2/local/admin/working directory!
# Fixing up CFMSROOT and CVSROOT in the Makefile
     .
     .
     .
```

```
Enabling CVS history logging...
(Remove /usr2/local/admin/CVS/CVSROOT/history to disable.)
All done!  Running 'mkmodules' as my final step...
# Fixing up cvsinit path in the Makefile

Everything seems to be okay.  Please be sure to set the following
environment variables before you try to use CFMS.

For sh/ksh:

CFMSROOT=/usr2/local/admin
CVSROOT=/usr2/local/admin/CVS
export CFMSROOT CVSROOT

For csh/tcsh:

setenv CFMSROOT /usr2/local/admin
setenv CVSROOT /usr2/local/admin/CVS

And please be sure to add this install directory to your command
search path.
```

6. You are asked to enter the location on your system of **perl5**.if it is not located in the standard directory. The install script substitutes what you enter into all CFMS **perl** scripts. Press the Return key to select the standard default location.

7. You are asked to enter the location of your CFMS root directory. This is the directory where you want the CFMS working files and CVS repository to go. This directory must be writable by the CFMS user.

The install script now creates all the directories used by CFMS.

8. The script informs you how to set the CFMSROOT and CVSROOT environment variables:

```
% setenv CFMSROOT /usr2/local/admin
% setenv CVSROOT /usr2/local/admin/CVS
```

9. Set the PATH environment variable so that it contains the location where the CFMS is to be installed. For example, given that CFMS will be installed into directory `/opt/cfms`, the PATH variable should be set as follows:

   for Korn/Bourne/Sh:

   ```
   # PATH=$PATH:/opt/cfms
   # export PATH
   ```

   for Csh:

   ```
   % set path=($path /opt/cfms)
   % rehash
   ```

Problems ?

Commands not successful

If a command fails, an error message tells you. The router log in may have failed, the target router is busy, the network is slow, the password was wrong, this sort of condition.

Try the command once or twice again, this is usually effective.

```
# ./cfms edit fox etc/login.conf
Getting configs from fox.site.com.  Please wait...
ERROR: getFiles could not get files
Couldn't get configs from fox.site.com.
Please try again. Also, see CFMS log file for more information
```

As this message suggests, you may be able to track where the problem occurs because the trace log file presents an audit trail of CFMS operations. More information about the trace log file `cfms-trace.log`, is in the “Using CFMS commands” section that appears later in this chapter.

Make again

If the **cfms add_host** command cannot complete and successfully add routers to the managed list, CFMS may need to be cleaned up. Execute this command:

```
# make startover
```

The command erases the existing repository, removes all working files, and empties the `cfms-hosts` database.

The **makefile startover** command may not have access to the CVS initialization script at `/usr/contrib/bin/cvsinit`. If the **cvsinit.sh** script is located in another place, rerun **install.pl** to update the path.

Check tclsh

If **tclsh** is not installed in a standard location (`/usr/local/bin`), then the TCL_LIBRARY environment variable needs to be set to point to the location where **Tcl** is installed. If it is installed in `/usr/local/lib/tcl8.0`., then you need to reset the variable as shown here:

Examples:

for Korn/Bourne/Sh:

```
# export TCL_LIBRARY=/usr/local/lib/tcl8.0
```

for Csh:

```
% setenv TCL_LIBRARY /usr/local/lib/tcl8.0
% rehash
```

Changing the CFMS directory

If you need to run CFMS from a directory location other than where it was installed, add the new location to the search path of the shell CFMS is being run from.

If CFMS was installed into directory `/opt/cfms`, the PATH variable must be set as follows:

Korn/Bourne/Sh:

```
# export PATH=$PATH:/opt/cfms
```

or Csh:

```
% set path=($path /opt/cfms)
% rehash
```

Setting up CFMS management

To establish CFMS management for one or a group of routers, these are the tasks:

1. Review the default list of configuration files you want CFMS to monitor. The list is stored in the installation directory in `/grf-files`.
2. Collect the following information for each target router:
   – hostname
   – the username and password that CFMS needs to log in to the router
   – the root password for the router
3. Use the **cfms add_host** command to add routers to CFMS.

1. Define the configuration file list

The same set of files are monitored for each CFMS-managed router. Here is the default list of configuration files that CFMS will collect and monitor:

```
etc/
    aitmd.conf
    bridged.conf
    filterd.conf
    gated.conf
    grarp.conf
    grass.conf
    gratm.conf
    grclean.conf
    grclean.logs.conf
    grfr.conf
    grifconfig.conf
    grlamap.conf
    grpp.conf
    grroute.conf
    login.conf
    snmpd.conf
    syslog.conf
```

You can edit `grf-files` to remove any file you do not wish to include. The default editor is **vi** unless you specify a different editor when prompted by the install script. You can add files that are not included in the default list.

File names are specified one name per line, and you must include the file path from the root file system. However, do not include the leading slash (/) character. To add the file `/usr/sbin/grconslog`, make this entry in `grf-files`:

```
    usr/sbin/grconslog
```

2. Collect router information

For each router you want to add, you need to know:

- its hostname
- the user name and password that CFMS needs to log in to the router
- the `root` password for the router

3. Add routers to CFMS

To put a router under CFMS management, use the **cfms add_host** command, also abbreviated to **cfms add**.

You can run the command either in batch mode or interactively, in which case you are prompted for each parameter. All CFMS commands must be run from within the CFMS installation directory.

The **cfms add_host** command has two formats, the first is interactive, the second is batch:

cfms add_host *hostname*
cfms add_host *hostname username passwd rootpw*

where:
hostname is the name of the router to put under CFMS management
username is the administrative user name under which CFMS logs in to the router
passwd is the password for the ***username*** login
rootpw is the password for user `root` on the router

Interactive format

When you run **cfms add_host *hostname***, you are using the interactive form. You are prompted for the remaining parameters. All passwords are twice-prompted to ensure that they are entered properly, but passwords are not echoed on the terminal.

Batch format

When you use the batch form of **cfms add_host**, you must provide all the parameters as command line arguments. All parameters must be provided because CFMS does not prompt for missing parameters.

CFMS operation

For each **cfms add_host *hostname*** or **cfms add** command, the CFMS application:

- **telnets** to the remote router
- logs in with the specified user name and password
- changes to `root` with the specified `root` password
- checks for the existence of **uuencode** and **uudecode**

If CFMS cannot perform any one of the operations, an error is reported to the user and the router is not added. The error is logged to the `cfms-trace.log` file.

If CFMS successfully performs all the operations, the specified router is added to the CFMS database. CFMS then copies the router's configuration files listed in `grf-files` to the CFMS server, and places the files into the CVS repository.

You must be in the `/cfms` directory to use the CFMS commands. Add one user per command, you do not need to enter the full hostname.

Enter CFMS commands with `./` preceding the command.

This example adds a router with the domain name `fox.site.com`:

```
# cd /cfms        ##if needed
#
# ./cfms add_host fox

Enter : Adding fox.site.com
Please enter the userid for maintaining host fox.site.com: cfmsuser
Please enter the password for user cfmsuser:
Re-enter the password for user cfmsuser:
Please enter the root password for the host fox.site.com:
Re-enter the root password for the host fox.site.com:
Checking host fox.site.com.  Please wait...
Host fox.site.com checked successfully.
Making Repository for fox.site.com.
Getting configuration files from fox.site.com. This can be SLOW..
Please wait...
Got configuration files from fox.site.com.
Adding hostname configuration files to repository.
Host fox.site.com has been placed under cfms control.
#
```

Re-adding a router to CFMS management

When a router is re-added to CFMS management, CFMS will likely find that server and router configuration file versions are different. The **cfms add** script will detect the conflicting versions and will prompt you to either:

- import the router's local versions of the configuration files to the CFMS server (answer "no" to the prompt)
- or, to install the server's versions of configuration files on the re-added router. (answer "yes" to the prompt)

Here is an example of the **cfms add** script and prompt:

```
Checking host fox.site.com.  Please wait...
Host fox.site.com checked successfully.
Making Repository for fox.site.com.
Getting configuration files from fox.site.com. This can be SLOW..
Please wait...
Got configuration files from fox.site.com.
Adding hostname configuration files to repository.
```

```
We have a conflict during import in the following files:
   .
   .
   .
I can try to resolve it for you using this command:
  cvs checkout -jCFMS:yesterday -jCFMS fox.ascend.com
Shall I do this for you? [n]  y

Host fox.site.com has been placed under cfms control.
Use the “cfms list” command to see the list of hosts under cfms control
#
```

Using CFMS commands

After CFMS is installed and the routers are set up, you can use CFMS commands to:

- list those routers currently under CFMS management
- change router configuration files on the server and send those files to the target router
- get the revision and change status of CFMS-tracked configuration files

List managed routers - cfms list

To list the routers currently under CFMS management, use the **cfms list** command.

The **cfms list** command has the following format:

cfms list

Enter CFMS commands with `./` preceding the command.

The **cfms list** command prints out the host name of each router in the CFMS database, one router per line:

```
# cd installation_directory
# ./cfms list

fox.site.com
intrepid.site.com
   •
   •
   •
```

Change router configuration files - cfms edit

To change the configuration for a target router, edit the configuration files on the CFMS server by using the **cfms edit** command. You must run an edit command for each configuration file you want to change.

The **cfms edit** command has the following format:

cfms edit *hostname filename*

where:

hostname is the name of the target router

filename is the specific configuration file from the list kept in `/grf-files`.

Enter CFMS commands with `./` preceding the command.

For each **cfms edit** command, the CFMS application does the following:

- Gets the current version of the specified file from the target router and checks the file into the CFMS repository.
- Invokes the UNIX editor specified by the EDITOR environment variable. If no editor was specified, it invokes **vi** by default.

At this point, the administrator edits the file as needed and saves the new file.

– After the file is saved, CFMS opens a text file (window) in which the user is requested to describe the line change made and the reason for the change.

After the user enters and saves this text, the file closes. This data is placed in a CVS repository log and serves as documentation for the changes.

– Checks the modified file into the CFMS repository.

– Puts the file in the remote router's `/etc` directory, overwriting the previous version of the file.

The user must log into the target GRF router and activate the new configuration.

In this example, changes are made to the router's `grifconfig.conf` file:

```
# ./cfms edit fox.site.com etc/grifconfig.conf

Getting configs from fox.site.com.  Please wait... Done.
```

The specified configuration file is open for editing. After it is saved, the history file is available. You are asked for some history log information:

```
CVS  Enter reason for modifications to fox.site.com.
CVS  Lines beginning with CVS will be ignored.
```

You need to use a CVS command to access this history information. Refer to the "CVS repository" section that appears earlier in this chapter.

After the history file is saved, CFMS sends the new configuration file to the specified router. You see the following message if the transfer was successful:

```
Pushing configs to fox.site.com.  Please wait... cfms edit : Done.
```

Files edited on router

If a file is edited directly on the router, this causes a conflict with the file version in the CFMS server database.

The output below shows an example of how CFMS notifies you that the server and router files do not match and how it handles the conflict:

```
***  Conflicts found while committing changes.  ***
The router configuration file is different from the file on this server.
Your changes have been saved locally on this server. Please rerun the
edit command to resolve conflicts manually.

Conflicts are marked by <<<< and >>>> markings in the configuration file.
Conflicts can arise when a configuration file is edited directly on the
router, instead of using CFMS on the server.
Pushing configs to test.ascend.com. Please wait...cfms edit: Done.
```

Get the status of repository files - cfms status

To get the status of files in the CVS repository, use the **cfms status** command. The command returns the list of all repository files for the specified router, along with the CVS status and revision number of each file.

The **cfms status** command has the following format and an optional parameter:

cfms status [***hostname***]

where:

hostname is the name of the router for which you want status information.

If you do not specify a router, the status for all managed routers is returned.

Enter CFMS commands with `./` preceding the command.

```
# ./cfms status fox

fox.site.com:
  Name: etc/aitmd.conf        Status:            Revision: 1.1.1.1
  Name: etc/bridged.conf      Status:            Revision: 1.2
  Name: etc/filterd.conf      Status:            Revision: 1.2
  Name: etc/gated.conf        Status:            Revision: 1.6
  Name: etc/grarp.conf        Status:            Revision: 1.2
  Name: etc/grass.conf        Status:            Revision: 1.1.1.1
  Name: etc/gratm.conf        Status:            Revision: 1.3
  Name: etc/grclean.conf      Status:            Revision: 1.2
  Name: etc/grclean.logs.conf Status:            Revision: 1.2
  Name: etc/grfr.conf         Status:            Revision: 1.3
  Name: etc/grifconfig.conf   Status:            Revision: 1.5
  Name: etc/grlamap.conf      Status:            Revision: 1.1.1.1
  Name: etc/grppp.conf        Status:            Revision: 1.2
  Name: etc/grroute.conf      Status:            Revision: 1.1.1.1
  Name: etc/login.conf        Status:            Revision: 1.1.1.1
  Name: etc/snmpd.conf        Status:            Revision: 1.3
  Name: etc/syslog.conf       Status:            Revision: 1.1.1.1
  Name: usr/nbin/grconslog    Status:            Revision: 1.1.1.1
```

The Status field is currently not reported or may be reported as “Up to date.”

The Revision field indicates the number of changes made to the configuration file since the router was brought under CFMS management:

`Revision: 1.1.1.1`	– no change has been made since
`Revision: 1.2`	– two changes have been made
`Revision: 1.3`	– three changes have been made
`Revision: 1.4`	– four changes have been made

and so on.

Read the CFMS trace log file

The CFMS log file, `cfms-trace.log`, resides in the same directory where CFMS is installed.

The trace log file contains a description of each operation CFMS performs on the server as well as on the managed routers. This includes operations ranging from log ins to file retrievals,

Note: This log file can grow rapidly, especially if CFMS is heavily used. The file must be cleaned frequently to avoid wasting CFMS server resources.

Use a UNIX command to view the trace log file. This example shows entries made as CFMS starts up:

```
# more cfms-trace.log

Wed Jun 24 16:43:28 1998:
There are currently no hosts being managed by CFMS.

Wed Jun 24 16:45:43 1998:
Host fox.site.com has been added to HostDB.

Wed Jun 24 16:45:43 1998:
Checking host fox.site.com.  Please wait...

Wed Jun 24 16:45:43 1998:
CFMS::command : preprouter.exp fox.site.com / etc/aitmd.conf
etc/bridged.conf etc/filterd.conf etc/gated.conf etc/grarp.conf
etc/grass.conf etc/gratm.conf etc/grclean.conf etc/grclean.logs.conf
etc/grfr.conf etc/grifconfig.conf etc/gritd.conf etc/grlamap.conf
etc/grppp.conf etc/grroute.conf etc/inetd.conf etc/login.conf
etc/pccard.conf etc/snmpd.conf etc/syslog.conf usr/nbin/grconslog

preprouter.exp : telnet to fox.site.com: as cfmsuser
spawn telnet fox.site.com
Trying 192.116.22.10...
Connected to fox.site.com.
Escape character is '^]'.

Ascend Embedded/OS 1.4.8 (fox.site.com) (ttyp2)
User: cfmsuser
Password:?[K
erase ^H, kill ^U, intr ^C status ^T
$ su root
Password:
# /bin/sh

# preprouter.exp : Returned from prepFiles to fox.site.com
preprouter.exp : All done

Wed Jun 24 16:45:50 1998:
Host fox.site.com checked successfully.

Wed Jun 24 16:45:50 1998:
Making Repository for fox.site.com.

Wed Jun 24 16:45:50 1998:
```

```
Creating user directory /usr2/local/cfms/CFMS/working/automaton

Wed Jun 24 16:45:50 1998:
Creating subdirectory for host fox.site.com

Wed Jun 24 16:45:50 1998:
Creating etc subdirectory /usr2/local/cfms/CFMS/working/automa-
ton/fox.site.com
```

Remove CFMS routers - cfms remove_host

To delete a router from CFMS management, use the **cfms remove_host *hostname*** command.

The **cfms remove_host** command has the following formats:

cfms remove_host *hostname*

cfms remove *hostname*

where:

hostname is the name of the router to delete from CFMS management

The command removes the specified router from the CFMS database and cleans up other internal CMFS data related to that router.

Enter CFMS commands with `./` preceding the command.

```
# ./cfms remove fox.site.com
#
```

Use the **cfms list** command to verify that the specified router is deleted.

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W