

GRF Configuration Guide 1.4

Ascend Communications, Inc.

Part Number: 7820-2011-001

For software version 1.4

December, 1997

GRF is a trademark of Ascend Communications, Inc. Other trademarks and trade names mentioned in this publication belong to their respective owners.

Copyright © 1997, Ascend Communications, Inc. All Rights Reserved.

This document contains information that is the property of Ascend Communications, Inc. This document may not be copied, reproduced, reduced to any electronic medium or machine readable form, or otherwise duplicated, and the information herein may not be used, disseminated or otherwise disclosed, except with the prior written consent of Ascend Communications, Inc.

Ascend Customer Service

You can request assistance or additional information by telephone, email, fax, or modem, or over the Internet.

Obtaining Technical Assistance

If you need technical assistance, first gather the information that Ascend Customer Service will need for diagnosing your problem. Then select the most convenient method of contacting Ascend Customer Service.

Information you will need

Before contacting Ascend Customer Service, gather the following information:

- Product name and model
- Software and hardware options
- Software version
- Service Profile Identifiers (SPIDs) associated with your product
- Whether you are routing or bridging with your Ascend product
- Type of computer you are using
- Description of the problem

How to contact Ascend Customer Service

After you gather the necessary information, contact Ascend in one of the following ways:

Telephone in the United States	800-ASCEND-4 (800-272-3634)
Telephone outside the United States	510-769-8027 (800-697-4772)
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
Spain/Portugal	(+33) 492 96 5675
UK	(+33) 492 96 5671
Email	support@ascend.com
Email (outside US)	EMEAsupport@ascend.com
Facsimile (FAX)	510-814-2312
Customer Support BBS by modem	510-814-2302

You can also contact the Ascend main office by dialing 510-769-6001, or you can write to Ascend at the following address:

Ascend Communications
1701 Harbor Bay Parkway
Alameda, CA 94502

Need information about new features and products?

Ascend is committed to constant product improvement. You can find out about new features and other improvements as follows:

- For the latest information about the Ascend product line, visit our site on the World Wide Web:

`http://www.ascend.com`

- For software upgrades, release notes, and addenda to this manual, visit our FTP site:

`ftp.ascend.com`

Contents

Ascend Customer Service	iii
-------------------------------	-----

About This Guide xxvii

How to use this guide.....	xxvii
What you should know	xxviii
Manual set.....	xxviii
Related publications	xxviii
Documentation conventions.....	xxix

Chapter 1 Working with the CLI 1-1

Logging on to a GRF 400 or 1600	1-2
First-time power on configuration script	1-2
root log on	1-3
CLI commands.....	1-3
UNIX commands	1-3
Change password	1-4
Add users	1-4
Administrative LAN log on	1-4
Logging on to systems with RMS nodes	1-5
Initial configuration script.....	1-5
root log on	1-5
CLI commands.....	1-6
UNIX commands	1-6
Change password	1-6
Add users	1-7
Administrative LAN log on	1-7
Introduction to the command-line interface (CLI).....	1-8
CLI command table.....	1-8
On-line help options.....	1-12
CLI prompts.....	1-12
Line-editing commands.....	1-14
Using the command-history buffer	1-14
Command-line shortcuts	1-15
Use the period (.)	1-15
Abbreviate field names	1-15
All other control characters	1-16
Printable characters.....	1-16
Use of asterisks	1-16
Paged line output.....	1-16

Introduction to profiles.....	1-17
Card.....	1-17
Dump	1-17
Load	1-17
System.....	1-17
User.....	1-17
Profile fields	1-18
Complex structure.....	1-19
Card profile components	1-21
Card profile field descriptions.....	1-22
1st-level fields	1-22
2nd-level fields.....	1-22
3rd-level port fields.....	1-24
Dump profile components.....	1-27
Dump profile field descriptions	1-29
1st-level fields	1-29
3rd-level fields	1-30
Load profile components	1-31
Load profile field descriptions	1-32
1st-level fields	1-32
2nd-level fields.....	1-32
System profile components	1-33
System profile field descriptions.....	1-34
User profile components	1-35
User profile field descriptions.....	1-36
1st-level fields	1-36
2nd-level fields.....	1-36
Working with profiles	1-38
Profile management commands.....	1-38
Access the profile set	1-39
Read profile into local memory	1-39
Viewing the contents of a profile.....	1-40
Viewing to change the contents of a profile	1-40
Check another profile before you change a setting.....	1-41
Moving up and down in a profile.....	1-42
Getting field information	1-43
Checking where you are.....	1-44
Changing a profile.....	1-45
Multiple set commands	1-45
Writing changes	1-46
Deleting a profile	1-46
Saving and loading alternate profiles.....	1-47
Using the save command	1-47
Using the load command	1-49
Creating a new profile.....	1-50
Adding user profiles.....	1-50
Using the new command.....	1-51

Chapter 2	Configuring System Parameters	2-1
	A note about configuring the GRF.....	2-1
	Configuration overview	2-3
	Configuration files and their uses	2-4
	Use grwrite to save config changes before boot	2-5
	Set up system logging	2-6
	Update changes to grclean.logs.conf (<i>if needed</i>)	2-6
	Log to external flash (option).....	2-7
	Set up a syslog server (option).....	2-9
	syslog.conf file.....	2-10
	Use an NFS-mounted file system (option).....	2-12
	Set up NFS on the GRF	2-12
	Set up IP routing	2-13
	Assign IP addresses - grifconfig.conf	2-13
	Interface name.....	2-13
	Internet address	2-14
	Netmask (optional).....	2-14
	Broadcast / destination address (optional).....	2-14
	Argument field.....	2-14
	Maximum transmission unit (MTU).....	2-15
	Define an alias or secondary address	2-15
	Create a loopback alias	2-15
	Defining an ISO address (IS-IS).....	2-16
	Put grifconfig changes into effect	2-16
	Change GRF hostname	2-17
	Enable host telnet access - /etc/ttys	2-18
	Configure SNMP (option).....	2-19
	Configure SNMP subagents.....	2-19
	Configure community names.....	2-19
	Configure system contact information.....	2-20
	Configure system name information.....	2-20
	Configure system location information	2-20
	Configure trap management.....	2-20
	Put configuration changes into effect	2-21
	SNMP support.....	2-22
	TCP/IP Network Management Support (RFC 1213).....	2-22
	Enterprise MIB support	2-22
	Enterprise TRAP support.....	2-23
	MIB locations	2-23
	Authentication options	2-24
	TACACS+ (option).....	2-24
	GRF client-side implementation	2-24
	Configuration steps on the GRF client.....	2-24
	Set RADIUS authentication (option)	2-26
	How RADIUS works	2-26
	Configure the GRF RADIUS client.....	2-26
	Fields in User profile	2-27
	Set securID (option)	2-28
	How securID works	2-28
	Configure the GRF securID client	2-29
	securID fields in User profile.....	2-30

Static-only IP routing (option)	2-31
grroute.conf file.....	2-31
Default route	2-31
Putting changes into effect.....	2-31
Error checking	2-31
route command.....	2-32
Static route example.....	2-32
Using route add or grroute.conf	2-32
Using GateD.....	2-33
Displaying static route tables	2-33
Using grrt	2-33
Static routes using GateD (option).....	2-34
GateD static statement	2-34
Display GateD route tables	2-35
Enable GateD (option)	2-36
Create and edit gated.conf.....	2-36
Start up the dynamic routing daemon	2-36
Assign an alias as routerid	2-36
Enable IP loose source routing (option).....	2-38
Selective packet discard (option)	2-39
ARP on the GRF	2-40
Things to remember when updating software	2-41
Changes to /etc/services are overwritten.....	2-41
Test a new binary	2-42
Test a new configuration.....	2-43
Do a system backup	2-44
Duplicate configurations among GRF systems.....	2-45
Save an alternate Load configuration.....	2-47
Run the config_netstart script	2-49
Reset the system to install files	2-50
Save configuration files	2-50
grms command (non-privileged login)	2-51
shutdown command (root login).....	2-51
Simple connectivity tests	2-52
Test remotely from a workstation or host	2-52

Chapter 3 **ATM OC-3c Configuration 3-1**

ATM OC-3c functions and features.....	3-2
Large route table support	3-2
PVC on-the-fly reconfiguration	3-2
Selective packet discard.....	3-2
Precedence field.....	3-2
Controlled-load (class filtering).....	3-3
Special buffer management during congestion.....	3-3
IS-IS protocol support.....	3-3
Broadcasting	3-4
Encapsulated bridging.....	3-4
Signaling	3-4
Packet buffering	3-5
Ping times	3-5
SDH and SONET modes	3-5
Inverse ARP.....	3-6

Internal/external clock source	3-6
MTU.....	3-6
ATM adaption layer support	3-6
ATM statistics and configuration data	3-7
Physical and logical ATM interfaces	3-8
Physical interfaces	3-8
Logical interfaces	3-8
Virtual circuits	3-9
Virtual paths.....	3-9
VCIs	3-9
VPI/VCI	3-10
Permanent virtual circuits	3-10
Switched virtual circuits	3-10
Verifying an ATM configuration.....	3-11
Installing configurations or changes	3-11
Configuration files and profiles	3-12
Assign IP addresses - grifconfig.conf	3-13
Specify ATM card parameters – Card profile.....	3-14
Change run-time code (optional) – Load profile	3-17
Change dump default (optional) – Dump profile	3-19
Configuring PVCs.....	3-21
Overview.....	3-21
PVC example	3-22
ATM on-the-fly PVCs	3-24
Support for SVCs	3-25
ARP servers	3-25
Configuration overview	3-25
SVC example	3-27
Traffic shaping	3-28
Parameters.....	3-28
Peak cell rate	3-28
Sustainable cell rate	3-29
Maximum burst size.....	3-29
Burst rate credits	3-29
Rate queues and QOS	3-30
Priority	3-31
Rate queue example	3-31
Traffic shape names	3-33
Setting output rates.....	3-34
Sending at a controlled rate.....	3-34
Allowing an average or fluctuating rate.....	3-34
Changing a rate queue.....	3-34
SVC creation process	3-35
Assumptions:	3-35
Configuring selective packet discard	3-37
Checking results.....	3-37
Example	3-38
maint commands for ATM OC-3c media cards	3-39
Preparing to use maint	3-39
Transmit / receive side maint commands.....	3-39
Receive side list	3-39
Transmit side list.....	3-40

Sample ATM maint displays	3-41
Display active interfaces	3-41
Check IP addresses	3-41
Check virtual circuits	3-41
Display ATM media statistics	3-42
Display switch statistics	3-44
VPI/VCI configuration	3-45
VPI/VCI traffic statistics	3-45
Select SUNI framing mode	3-46
Display ARP servers	3-46
Display rate queues	3-46
Display ATMP home network table	3-46
Display ATMP tunnel information	3-47
Display UME	3-47
Use grt to display the route table	3-48
Using grstat	3-48

Chapter 4 **FDDI Configuration 4-1**

FDDI functions	4-2
Large route table support	4-2
Selective packet discard	4-2
IS-IS protocol support	4-2
Transparent bridging	4-3
Proxy ARP	4-3
Controlled-load (class filtering)	4-3
How FDDI interfaces are named	4-4
Physical interface numbers	4-5
GRF interface name gf0yz	4-5
MTU	4-5
Configuration files and profiles	4-6
Installing configurations or changes	4-6
Set up FDDI media card – Card profile	4-7
Set up FDDI media card – Load profile	4-11
Set up FDDI media card – Dump profile	4-13
Installing configurations or changes	4-15
Single attach (SAS)	4-16
Dual attach (DAS)	4-16
Configuring SAS vs DAS	4-17
Installing FDDI connector keys	4-18
Basic functionality	4-18
Optical bypass switch interface	4-19
Manually activating optical bypass	4-19
Support for dual homing	4-21
Selective packet discard	4-21
Checking results	4-22
Example	4-22
Monitoring FDDI media cards	4-23
Canonical output	4-23
Using maint commands	4-23
Preparing to use maint	4-23
FDDI maint commands	4-24
Display port card S/W version	4-25

Verify FDDI configuration	4-26
List statistics per FDDI interface	4-26
List switch interface statistics	4-27
List communications bus interface statistics and status.....	4-27
Clear all statistics	4-28
Display current ARP table contents.....	4-28
Set history trace.....	4-28
Display SMT MIB variables.....	4-28
Reset individual FDDI interface	4-29
Display CAM addresses.....	4-29
Read MAC_CNTRL_A IF.....	4-30
Toggle promiscuous mode	4-30
Display CPU1 memory	4-31
Print FSI indirect registers	4-31
Print PHY registers/counters.....	4-31
Print MAC registers/counters	4-32

Chapter 5 **HIPPI Configuration** **5-1**

Introduction to HIPPI.....	5-2
Connection processing	5-2
Starting a HIPPI connection.....	5-3
How the I-field is used	5-3
Camp-on bit	5-3
Path selection bits	5-4
Direction bit	5-7
L, VU, and W bits	5-7
Taking stock.....	5-8
Beyond HIPPI	5-8
IP routing	5-8
What is an IP datagram ?	5-9
IP routing and the I-field.....	5-9
Using the IP address.....	5-9
HIPPI in a bridging environment.....	5-10
MTU.....	5-10
ARP	5-10
HIPPI standards and RFCs via ftp	5-10
HIPPI configuration options	5-11
Example 1: Source routing – host selects the path.....	5-12
Collect host information	5-12
Set up host I-field table	5-13
Example 2: Using logical addresses.....	5-14
Logical addressing configuration example	5-15
Set up host I-field logical addresses.....	5-16
Edit the logical address file – /etc/grlamap.conf.....	5-16
Downloading new mappings.....	5-17
Execute grlamap.....	5-17
Example 3: IP routing – HIPPI-to-HIPPI across a switch	5-18
HIPPI-to-HIPPI IP routing process.....	5-18
Configuration steps	5-18
Set up host I-field table to establish IP routing	5-19
Set site-specified address for IP routing	5-19
Downloading new mappings.....	5-20

Execute grlmap command.....	5-20
Map output IP address to output I-field – grarp command.....	5-20
Link destination IP address to output media card.....	5-20
Link destination IP address to forwarding I-field.....	5-21
Execute grarp command(s)	5-21
Example 4: IP routing – HIPPI-to-IP media	5-22
Host A-to-B IP transfers	5-22
Set up host I-field table	5-23
Execute grlmap command.....	5-24
Configure WAN media card	5-24
Special case 1: HIPPI IPI-3 routing	5-25
Special case 2: IBM H0 HIPPI option	5-26
Media card functions.....	5-26
Enabling H0 mode	5-26
Configuration files and profiles	5-28
Installing configurations or changes	5-29
Set up HIPPI media card – Card profile	5-30
Set up HIPPI media card – Load profile	5-33
Set up HIPPI media card – Dump profile	5-34
Installing configurations or changes	5-35
Monitoring HIPPI media cards	5-36
Preparing to use maint	5-36
List of HIPPI maint commands.....	5-36
Print IP statistics	5-37
Print IEEE address	5-38
Dump trace buffers	5-38
Print IP routing statistics.....	5-38
Dump trace buffers symbolically.....	5-38
Print switch error counts	5-39
HIPPI tunnel statistics.....	5-39
Show ARP table entries	5-40

Chapter 6 **HSSI Configuration 6-1**

HSSI (High Speed Serial Interface) implementation.....	6-2
Framing protocol options	6-2
Frame Relay.....	6-2
HDLC.....	6-2
Point-to-Point Protocol (PPP)	6-3
Protocol per interface support.....	6-3
IS-IS protocol support.....	6-3
Large route table support	6-4
Selective packet discard.....	6-4
Controlled-load (class filtering).....	6-4
Physical and logical interfaces	6-5
Physical interfaces	6-5
Logical interfaces.....	6-5
Interface name gs0yz	6-5
Identify logical interfaces in grifconfig.conf	6-6
Card connection options.....	6-7
Link type	6-7
Null modem cabling (HSSI crossover cables)	6-7
Cyclic redundancy checking (CRC)	6-9

Configuration file and profile overview.....	6-11
Installing configurations or changes	6-12
Set up HSSI media card – Card profile	6-13
Set up HSSI media card – Load profile	6-18
Set up HSSI media card – Dump profile.....	6-19
Selective packet discard	6-21
Checking GateD results	6-21
Example	6-22
SPD statistics	6-22
Configuring the HDLC protocol	6-23
Configuring the Frame Relay protocol	6-26
Configure link parameters.....	6-27
Required link parameter descriptions	6-28
Optional link descriptors.....	6-28
Link descriptor examples.....	6-28
Configure PVCs	6-28
Number of configurable PVCs	6-29
DLCI numbering.....	6-29
Required PVC parameters	6-29
Optional PVC parameters	6-29
PVC examples	6-29
Adding a logical interface on-the-fly.....	6-29
Adding a PVC on-the-fly	6-30
Assigning multiple DLCIs	6-30
grfr command set	6-31
Configuring Point-to-Point Protocol	6-32
Required interface parameter.....	6-33
Optional negotiation automata parameters	6-33
Link Control Protocol (LCP) parameters.....	6-33
Link Quality Reporting (LQR) parameters.....	6-34
IPCP parameter	6-34
PPP status reporting	6-34
grppp status commands.....	6-34
Looking at a PPP configuration	6-35
grppp.conf file	6-37
Monitoring HSSI media cards.....	6-38
Preparing to use maint	6-38
Display maint commands.....	6-38
Read S/W and H/W revisions	6-39
Configuration and status	6-39
Display media statistics.....	6-40
Display switch statistics:.....	6-40
Clear status info	6-41
Display PVC status	6-41
Display IPC statistics	6-42
Display next hop data.....	6-42
List of filters.....	6-43
Display filtering statistics	6-43
Display ATMP home network table	6-43

Chapter 7 Ethernet Configuration..... 7-1

Ethernet implementation	7-2
Selective packet discard	7-2
IS-IS protocol support	7-2
Large route table support	7-3
LLC/SNAP support	7-3
ARP support	7-3
Proxy ARP	7-3
MTU	7-3
CSMA/CD (flow control)	7-3
Transparent bridging	7-4
Transfer rates	7-4
Cables	7-4
Controlled-load (class filtering)	7-4
Physical and logical interfaces	7-5
Physical interfaces	7-5
Logical interfaces	7-5
Interface name	7-5
Configuration file and profile overview	7-6
Installing configurations or changes	7-7
Identify logical interfaces	7-7
Set up Ethernet media card – Card profile	7-8
Set up Ethernet media card – Load profile	7-13
Set up Ethernet media card – Dump profile	7-14
Selective packet discard	7-16
Checking GateD results	7-16
Example	7-17
Monitoring Ethernet media cards	7-18
Preparing to use maint	7-18
Display maint commands	7-18
Display operating status	7-20
Media statistics	7-21
Display switch statistics	7-24
Display Combus statistics	7-24
Clear status info	7-25
Display ARP tables	7-25
List of filters	7-26
Display filtering statistics	7-26

Chapter 8 SONET OC-3c Configuration 8-1

SONET implementation	8-2
Frame Relay	8-2
HDLC	8-2
Point-to-Point Protocol (PPP)	8-3
IS-IS protocol support	8-3
Large route table support	8-4
Selective packet discard	8-4
Controlled-load (class filtering)	8-4

Logical and physical interfaces	8-5
Physical interfaces	8-5
Logical interfaces.....	8-5
Interface name.....	8-5
Identify logical interfaces in grifconfig.conf	8-6
Card connection options.....	8-7
Link type	8-7
Null modem cabling (SONET crossover cables).....	8-7
Configuration file and profile overview.....	8-9
Installing configurations or changes	8-11
Configure logical interfaces	8-12
Set up SONET media card – Card profile.....	8-12
Installing configurations or changes	8-16
Set up SONET media card – Load profile	8-17
Set up SONET media card – Dump profile	8-18
Selective packet discard	8-20
Checking GateD results	8-20
Example	8-21
SPD statistics	8-21
Configuring the HDLC protocol.....	8-22
Configuring the Frame Relay protocol	8-24
Configure link parameters.....	8-25
Required link parameter descriptions	8-25
Optional link descriptors.....	8-25
Link descriptor examples.....	8-26
Configure PVCs	8-27
Number of configurable PVCs	8-27
DLCI numbering.....	8-27
Required PVC parameters	8-27
Optional PVC parameters	8-27
PVC examples	8-27
Adding a logical interface on-the-fly.....	8-28
Adding a PVC on-the-fly	8-28
Assigning multiple DLCIs	8-29
grfr command set	8-29
Configuring Point-to-Point Protocol	8-30
Required interface parameter.....	8-31
Optional negotiation automata parameters	8-31
Link Control Protocol (LCP) parameters.....	8-31
Link Quality Reporting (LQR) parameters.....	8-31
IPCP parameter	8-31
PPP status reporting	8-32
grppp status commands.....	8-32
Looking at a PPP configuration	8-32
grppp.conf file	8-33
Monitoring SONET OC-3c media cards.....	8-34
Preparing to use maint	8-34
Display maint commands.....	8-34
Display software and hardware versions	8-35
Display card configuration and status.....	8-36
Display media statistics.....	8-36
Display switch statistics.....	8-36

Display RX combus statistics	8-37
Clear status info	8-37
Display ARP table.....	8-38
Display IPC statistics	8-38
List of filters.....	8-38
Display filtering statistics	8-38
Display signal detect	8-39
Display TX combus statistics.....	8-39
List next hop data.....	8-39

Chapter 9 **Ascend Tunnel Management Protocol 9-1**

Introduction to ATMP	9-2
How ATMP connections work	9-2
GRF ATMP implementation.....	9-3
Starting aitmd	9-4
Support for virtual private networks	9-5
GRF in gateway mode	9-5
Scalability	9-5
Memory usage	9-6
Interoperability	9-6
IP packets and GRE	9-7
ATMP tunnel components	9-8
Tunnel ID	9-8
Tunnel negotiation	9-10
Life-cycle of a tunnel	9-11
Mobile node IP addresses	9-12
RADIUS profile.....	9-12
Home agent atmp0 IP addresses	9-13
ATMP gateway circuit (pvcatmp)	9-15
Number of home networks.....	9-15
Connection from home network	9-15
OSPF advertises home network addresses.....	9-15
Configuring the GRF as a home agent.....	9-17
A. Connection to a home network	9-17
HSSI Frame Relay configuration.....	9-17
Optional link parameters.....	9-18
Optional PVCATMP parameters	9-19
Large packets through tunnel.....	9-19
1. ATM OC-3c configuration	9-20
2. Configure GRF ATMP parameters in aitmd.conf.....	9-22
3. Configuration links to the TNT foreign agent	9-24
Mobile node RADIUS profile.....	9-24
Ascend-Primary-Home-Agent	9-25
Ascend-Home-Agent-Password	9-26
Ascend-Home-Agent-UDP-Port = 5150.....	9-26
Ascend-Home-Agent-Name	9-26
GRF ATMP tunnel gateway - example 1	9-27
GRF ATMP tunnel gateway - example 2	9-28
Extended ATMP configuration example	9-29
Configure mobile node and office server	9-29
Configure mobile host and RADIUS client.....	9-30
Configure the home agent GRF.....	9-31

Configure the home network router GRF	9-32
Monitoring ATMP activity on the GRF	9-33
maint commands	9-33
List home networks configured per HSSI or ATM card	9-33
maint 70 examples	9-34
Display tunnel information	9-35
netstat -i command	9-36
Using grstat commands	9-37
Using grfr commands	9-39
Display PVC statistics	9-39
Display media card interface status	9-39
Display link configuration and status	9-40
Display configured PVCs	9-40
Display system configuration and status	9-41
Display configured interfaces	9-41
Adding PVCs on-the-fly	9-42

Chapter 10 ATM OC-12c Configuration 10-1

ATM OC-12c functions	10-2
Large route table support	10-2
Selective packet discard	10-2
IS-IS protocol support	10-2
Packet buffering	10-2
MTU	10-2
LLC/SNAP encapsulation	10-3
NULL encapsulation	10-3
Raw ATM mode support	10-3
SDH and SONET modes	10-3
Inverse ARP	10-3
Setting SUNI receive clock	10-4
Broadcasting	10-4
Selective packet discard	10-4
Checking results	10-5
Example	10-5
Physical and logical ATM OC-12c interfaces	10-6
Physical interfaces	10-6
Logical interfaces	10-6
Virtual circuits	10-7
Virtual paths	10-7
VCIs	10-7
VPI/VCI	10-8
Permanent virtual circuits	10-8
Switched virtual circuits	10-8
Verifying an ATM configuration	10-8
Installing configurations or changes	10-8
Configuration file and profile overview	10-9
Assign IP addresses - grifconfig.conf	10-10
Specify ATM card parameters – Card profile	10-11
Change run-time code (optional) – Load profile	10-14
Change dump default (optional) – Dump profile	10-16

Configuring PVCs	10-18
Overview	10-18
OC-12c PVC parameters.....	10-18
PVC example	10-19
Traffic shaping	10-20
Profile parameters	10-20
Peak cell rate	10-20
Rate queues and QOS	10-21
Sustained cell rate	10-22
Maximum burst size.....	10-22
Assigning traffic shaping profiles.....	10-22
Traffic shape names	10-24
ATM OC-12c parameters	10-24
Queueing	10-24
Setting output rates.....	10-25
Sending at a controlled rate.....	10-25
Allowing an average or fluctuating rate.....	10-25
Changing a rate queue.....	10-25
maint commands for ATM OC-12c media cards.....	10-26
Preparing to use maint	10-26
Lists of maint commands	10-26
Display card version numbers.....	10-27
Display the interface configuration.....	10-27
Display virtual circuit statistics.....	10-27
Display ARP information	10-28
Display memory usage.....	10-28
Display packet traffic counts	10-29
Display VPCI configuration	10-29
Display broadcast groups.....	10-30
Setting parameters.....	10-30

Chapter 11 **IP Packet Filtering** **11-1**

GRF filtering implementation	11-2
Configuration daemon	11-2
Configuration file.....	11-2
CLI access to filterd.conf.....	11-2
maint commands	11-2
Filters	11-3
Rules.....	11-3
Applying a mask	11-4
Applying a filter.....	11-4
Filters for service ports	11-5
Specifying port numbers.....	11-5
Bindings	11-6
Logical interface number (vlif).....	11-6
Media type	11-7
Direction	11-7
Actions	11-8
Filtering states.....	11-8
Packet header logging	11-9
Logging loops	11-10
Loops caused by ICMP messages.....	11-11

Controlling access to the internal system.....	11-12
Filtering configuration process	11-14
Changing filters.....	11-14
Filtering configuration file – filterd.conf	11-15
Filter grammar reference.....	11-18
Monitoring filtering operations – maint commands	11-21
Preparing to use maint	11-21
Filtering command set.....	11-21
Translating filterIDs to actual names.....	11-23
Display list of actions.....	11-25
Display filtering statistics	11-25
Clear statistics	11-25
Show protocol statistics	11-26
 Chapter 12	
Introduction to GRF BGP	12-1
Introduction to BGP	12-2
Exterior protocols.....	12-2
EGP – BGP transition	12-2
BGP protocol features	12-3
Autonomous systems	12-3
Sessions.....	12-4
Route advertisements	12-4
Enforcing policy.....	12-5
Path selection	12-5
Additions to BGP	12-6
Configurable Export-Best-BGP (CEBB).....	12-6
Asynchronous Multi-Level Next Hop Resolution (AMLNHR)	12-7
CLI access to gated.conf	12-7
BGP confederations	12-7
GSM - monitoring tool for GateD	12-8
MEDs	12-9
Communities	12-10
Destination preference attribute (DPA)	12-10
Route reflection.....	12-10
Routing arbiter interaction	12-10
Route preference biasing.....	12-11
Interface statement	12-11
BGP Group statement	12-11
BGP statement.....	12-12
Configuring multi-exit discriminators (MEDs)	12-14
Originating a MED	12-14
BGP group statement	12-14
BGP peer statement.....	12-14
BGP export statement	12-15
Configuring DPA attributes	12-16
Using import policy	12-16
Export policy	12-16
Configuring route reflection.....	12-17
Route reflection example	12-19
Weighted route dampening	12-23
Weighted route dampening statement.....	12-23

Chapter 13	Introduction to IS-IS.....	13-1
	IS-IS Intra-Domain Protocol	13-2
	IS-IS Statement	13-2
	Assign configuration values	13-3
	Assign a systemid	13-3
	Assign an area address	13-3
	Derive an ISO address	13-3
	Settings in the IS-IS statement	13-4
	Enable IS-IS	13-4
	Configure area	13-4
	Configure systemid	13-4
	Configure interface	13-5
	Enable IS-IS on HSSI interfaces	13-6
	Frame Relay	13-6
	PPP	13-6
	HDLC	13-6
	Enable IS-IS on ATM/Q interfaces	13-6
	Assign ISO address in grifconfig.conf	13-7
	Configuration example 1	13-8
	Router 1 configuration	13-8
	Line to add in grifconfig.conf:	13-8
	Entries in gated.conf:	13-8
	Router 2 configuration	13-9
	Lines to add in grifconfig.conf:	13-9
	Line to add in grppp.conf :	13-9
	Entries in gated.conf:	13-9
	Router 3 configuration	13-9
	Lines to add in grifconfig.conf:	13-9
	Line to add in grppp.conf :	13-9
	Lines to add in gratm.conf:	13-10
	Entries in gated.conf:	13-10
	Configuration example 2	13-11
	Router A1 configuration	13-11
	Lines to add in grifconfig.conf:	13-11
	Entries in gated.conf:	13-11
	Router A2 configuration	13-12
	Line to add in grifconfig.conf:	13-12
	Entries in gated.conf:	13-12
	Router B1 configuration	13-12
	Lines to add in grifconfig.conf:	13-12
	Line to add in grfr.conf:	13-12
	Entries in gated.conf:	13-13
	Router B2 configuration	13-13
	Line to add in grifconfig.conf:	13-13
	Line to add in grfr.conf:	13-13
	Entries in gated.conf:	13-13
	GSM support for IS-IS	13-15

Chapter 14	Integrated Services: Controlled-Load.....	14-1
	Overview	14-2
	Controlled-Load implementation	14-2
	Filters	14-3
	Filter examples	14-3
Chapter 15	Transparent Bridging.....	15-1
	GRF bridging implementation	15-2
	Specifications	15-2
	Simultaneous routing and bridging	15-3
	Configuration options	15-3
	Interoperability	15-3
	Spanning tree	15-4
	Bridge filtering table	15-4
	Fragmentation	15-4
	Spamming	15-4
	Bridging components	15-5
	Bridging daemon – bridged	15-5
	Configuration file – bridged.conf	15-5
	Editing utility – bredit	15-5
	Management tools	15-6
	brstat.....	15-6
	brinfo.....	15-6
	Configuration file and profile overview.....	15-7
	1. Starting bridged.....	15-8
	Bridging example	15-9
	2. Creating bridge groups in bridged.conf	15-10
	3. Assign IP addresses to bridge groups	15-11
	4. Create an ATM PVC for an encapsulated bridge	15-12
	Configuration in gratm.conf.....	15-12
	PVC configuration examples	15-14
	Configuration for ARP service - grarp.conf	15-15
	Installing configuration changes	15-15
	Sources of bridging data	15-16
	Bridging trace log	15-16
	Bridge group information	15-17
	Low-level state information.....	15-17
	Route trees and filtering table	15-18
	Examining and debugging bridge configurations	15-19
	Introduction.....	15-19
	Information needed by Ascend support	15-19
	Enabling traces via bridged command.....	15-19
	Displaying useful information	15-20
	Using brinfo	15-21
	State information - brstat	15-22
	Bridge IDs via netstat -ni	15-23
	Restarting bridged during debug.....	15-24

Appendix A **Subnetting**

What is subnetting?	A-2
Early implementation of classes and implicit masks	A-2
Classless inter-domain routing (CIDR)	A-3
Supernetting: benefits for routing	A-4
Support for explicit netmasks	A-5
Deriving a supernet address	A-5
A supernet routing example	A-6
Example 1: Traditional route storage method	A-7
Example 2: Subnet mask storage method	A-8
Forming a supernet address	A-8
Supernet derivation 1:	A-8
Supernet derivation 2:	A-8
Supernet derivation 3:	A-8
How the GRF uses a mask	A-9
Routing look-up example	A-10
Address-to-mask logical ANDing	A-10
Result-to-address comparison	A-11
Rules for matching	A-12
Longest match example	A-12

Appendix B **Warranty**

Product warranty	B-1
Warranty repair	B-1
Out-of warranty repair	B-2

Index.....	Index-1
-------------------	----------------

Figures

Figure 1-1	Diagram of Card profile levels	1-21
Figure 1-2	Dump profile: hw-table fields.....	1-27
Figure 1-3	Dump profile: dump vector tables	1-28
Figure 1-4	Diagram of Load profile levels.....	1-31
Figure 1-5	Diagram of GRF 400 System profile level (single level)	1-33
Figure 1-6	Diagram of GRF 1600 System profile level (single level)	1-33
Figure 1-7	Diagram of User profile levels	1-35
Figure 2-1	Components in the GRF interface name.....	2-13
Figure 2-2	Illustration for static routing configuration.....	2-32
Figure 2-3	Support for external flash devices in PCMCIA slot A	2-45
Figure 3-1	ATM physical and logical interfaces	3-8
Figure 3-2	Interface name structure for an ATM logical interface	3-8
Figure 3-3	Components that form a virtual path	3-9
Figure 3-4	GRF role in ATM-ATM connection.....	3-35
Figure 4-1	Assigning numbers to FDDI interfaces	4-4
Figure 4-2	Physical interface numbering on FDDI media card	4-5
Figure 4-3	GRF interface name for FDDI interfaces	4-5
Figure 4-4	Master/slave connector keys for single-attach interfaces	4-16
Figure 4-5	A/B connector keys for dual-attach interfaces.....	4-16
Figure 4-6	DAS and SAS configuration options	4-17
Figure 4-7	Types of FDDI connector keys.....	4-18
Figure 4-8	Optical bypass switch attachments	4-19
Figure 4-9	Dual homing options.....	4-21
Figure 5-1	HIPPI I-field components	5-3
Figure 5-2	I-field for source-directed routing	5-4
Figure 5-3	Return path created in source routing	5-5
Figure 5-4	I-field for logical addressing (PS is set to 01)	5-5
Figure 5-5	I-field for logical addressing (PS is set to 11)	5-6
Figure 5-6	Source routing and logical addressing with D = 0.....	5-7
Figure 5-7	Source routing and logical addressing with D = 1	5-7
Figure 5-8	I-field 0xfc0 entry that triggers IP routing.....	5-9
Figure 5-9	Planning diagram for source routing example.....	5-12
Figure 5-10	I-field list for source routing example	5-13
Figure 5-11	Planning diagram for logical addressing	5-15
Figure 5-12	Sample host I-field table for logical addressing	5-16
Figure 5-13	Planning diagram for HIPPI-HIPPI IP routing with switch	5-18
Figure 5-14	I-field for HIPPI-HIPPI IP routing example.....	5-19
Figure 5-15	Mapping an IP address to a destination I-field	5-20
Figure 5-16	Planning diagram for HIPPI IP routing	5-22

Figure 5-17	I-field for IP routing example	5-23
Figure 5-18	Planning diagram for HIPPI IPI-3 configuration.....	5-25
Figure 6-1	Logical interfaces supported per HSSI physical interface.....	6-5
Figure 6-2	HSSI interface name	6-5
Figure 6-1	Template for grppp.conf file	6-37
Figure 7-1	Components in Ethernet interface name	7-5
Figure 8-1	SONET support for redundant links	8-5
Figure 8-2	Components in SONET OC-3c interface name.....	8-5
Figure 8-3	Template for grppp.conf file.....	8-33
Figure 9-1	Components in a basic ATMP tunnel configuration	9-2
Figure 9-2	Movement of encapsulated packets through the tunnel.....	9-7
Figure 9-3	ATMP tunnel negotiation	9-10
Figure 9-4	How the atmp0 IP address is used	9-13
Figure 9-5	GRF configuration for an ATMP tunnel	9-27
Figure 9-6	GRF ATMP configuration supporting multiple home networks.....	9-28
Figure 9-7	Diagram of extended.....	9-29
Figure 10-1	ATM OC-12c physical and logical interfaces	10-6
Figure 10-2	Interface name structure for an ATM logical interface	10-6
Figure 10-3	Components that form a virtual path	10-7
Figure 11-1	Placing filter with the direction option	11-7
Figure 11-2	Receive-side logging filters do not loop.....	11-10
Figure 11-3	How logging loops can occur	11-10
Figure 11-4	Template for unedited filterd.conf file (part 1 of 3)	11-15
Figure 12-1	Full mesh and reflected route topologies.....	12-17
Figure 12-2	Client and non-client reflected route topologies.....	12-18
Figure 12-3	Route reflection configuration example	12-19
Figure 13-1	IS-IS configuration example - 1	13-8
Figure 13-2	IS-IS configuration example - 2	13-11
Figure 15-1	Bridging example diagram	15-9
Figure 15-2	Interface name for FDDI, Ethernet, and ATM OC-3c interfaces	15-10
Figure 15-3	Output from bridging trace file.....	15-16
Figure A-1	Specification of classes in IP addresses.....	A-2
Figure A-2	Basic supernetting example	A-4
Figure A-3	Example 1, a traditional route table with one entry per subnet	A-7
Figure A-4	Example 2: a route table with supernetting applied.....	A-9
Figure A-5	Routing logic: ANDing destination address to the subnet mask	A-11
Figure A-6	Bit-by-bit comparison to the supernet address	A-11

Tables

Table 1-1	CLI command list and descriptions	1-8
Table 1-2	CLI line-editing commands	1-14

About This Guide

The *GRF Configuration Guide* 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.

How to use this guide

The *GRF Configuration Guide* contains these chapters:

- Chapter 1, “Working with the CLI,” describes initial and administrative logins, and introduces the CLI.
- Chapter 2, “Configuring System Parameters,” details configuration of basic system-level operating parameters and options.
- Chapter 3, “ATM OC-3c Configuration,” describes configuration options and monitoring/debug commands available on ATM/Q media cards.
- Chapter 4, “FDDI Configuration,” describes configuration options and monitoring/debug commands available on FDDI/Q media cards.
- Chapter 5, “HIPPI Configuration,” describes configuration options and monitoring/debug commands available on HIPPI media cards.
- Chapter 6, “HSSI Configuration,” describes configuration options and monitoring/debug commands available on HSSI media cards.
- Chapter 7, “10/100Base-T Configuration,” describes configuration options and monitoring/debug commands available on fast Ethernet media cards.
- Chapter 8, “SONET Configuration,” describes configuration options and monitoring/debug commands available on SONET OC-3c media cards.
- Chapter 9, “Ascend Tunnel Management Protocol,” explains the Ascend Tunnel Management Protocol implementation on GRF HSSI media and provides configuration examples.
- Chapter 10, “ATM OC-12c Configuration,” describes configuration options and monitoring/debug commands available on ATM)C-12c media cards.
- Chapter 11, “IP Packet Filtering,” explains the application of packet filtering options.
- Chapter 12, “Configuring BGP,” describes additions to BGP that support requirements of high-traffic dynamic routing networks. The full text of GateD configuration Statements are included in the *GRF Reference Guide*.
- Chapter 13, “Introduction to IS-IS,” briefly explains the IS-IS implementation and provides configuration examples.

- Chapter 14, “Integrated Services,” describes the initial GRF implementation of Integrated Services, the provision of Controlled-Load services on Ethernet, FDDI, SONET, and HSSI media cards.
- Chapter 15, “Transparent Bridging,” describes the features of GRF bridging and provides media card configuration information and examples.
- 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.

This guide also includes an index.

What you should 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 documentation set consists of the following manuals:

- *GRF 400/1600 Getting Started 1.4*
- *GRF Configuration Guide 1.4* (this manual)
- *GRF Reference Guide 1.4*



Related publications

Here are some related publications that you may find useful:

- *TCP/IP Network Administration*, Craig Hunt (O’Reilly & Associates, Inc.)
- *Essential System Administration*, Aileen Frisch (O’Reilly & Associates, Inc.)
- *Internetworking with TCP/IP, Volume 1*, Douglas E. Comer, David L. Stevens (Prentice-Hall)
- *TCP/IP Illustrated, Volume 1*, W. Richard Stevens (Addison-Wesley)

Documentation conventions

This section explains all the special characters and typographical conventions in this manual.

Convention	Meaning
<code>Monospace text</code>	Represents text that appears on your computer's screen, or that could appear on your computer's screen.
<i>Italics</i>	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.
Note:	Introduces important additional information.
	Warns that a failure to follow the recommended procedure could result in loss of data or damage to equipment.
Caution:	
	Warns that a failure to take appropriate safety precautions could result in physical injury.
Warning:	

Working with the CLI

1

This chapter introduces the user command-line interface and ways to work with the management and configuration profiles. It also briefly explains the initial configuration script and log ons done during installations of the GRF 400, GRF 1600, and RMS node systems.

Chapter 1 includes these topics:

Logging on to a GRF 400 or 1600	1-2
Logging on to systems with RMS nodes	1-5
Introduction to the command-line interface (CLI)	1-8
CLI command table	1-8
Line-editing commands	1-14
Introduction to profiles	1-17
Profile fields	1-18
Card profile components	1-21
Card profile field descriptions	1-22
Dump profile components	1-27
Dump profile field descriptions	1-29
Load profile components	1-31
Load profile field descriptions	1-32
System profile components	1-33
System profile field descriptions	1-34
User profile components	1-35
User profile field descriptions	1-36
Working with profiles	1-38
Creating a new profile	1-50

Logging on to a GRF 400 or 1600

This section describes logging on to a GRF 400 or GRF 1600.

First-time power on configuration script

When you plug in the GRF it powers on and begins to boot. The first-time system configuration script runs automatically. Here are the questions asked in the script that you need to be ready to answer:

```
Host name for this machine ?
Do you wish to configure the maintenance Ethernet interface ?
Type of interface: TP BNC AUI  ?
IP address for this interface ?
Netmask for this network ?
Which region is this machine located in?
Which time zone ?
```

Next, you are prompted to change the local password for root:

```
New password (8 significant characters):
```

You are also prompted to set up a syslog server on which to save and access the GRF system logs. A target host must be specified to receive log entries or they will not be saved.

```
Enter the remote logging host name:
Enter the logging host IP address:

On <remote.host.name> at <ip_address>
add the following lines to /etc/syslog.conf:
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
Use "touch" to create the above files in <remote.host.name>: /etc
Kill and restart syslogd on <remote.host.name>.
Press <Enter> to continue.
Initial configuration is complete.
```

This is the end of the configuration script.

The configuration script automatically saves any configuration information you have entered. If you later add or change configuration information, save those changes by executing a **grwrite -v** command.

If later you need to change these parameters, run the **config_netstart** script described in the next chapter.

If you did not configure network logging during the configuration script, the *GRF 400/1600 Getting Started* manual describes how to establish a remote **syslog** server.

root log on

The GRF continues to boot. When boot finishes and you see the message, press the Enter key:

Press <Enter> to continue.

At the User prompt, type: root

User: root

Password:

At the Password: prompt, type the new password you entered in the configuration script.

If you did not change the password earlier, use the preset password, type: Ascend

Password:

When a password is entered, it is not echoed (displayed) on the screen.

The super> prompt appears:

super>

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

If 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.

CLI commands

Many system management and configuration commands are now available in the CLI. Enter a “?” to retrieve a list of CLI commands. CLI commands are described in the *GRF Reference Guide*.

To configure and manage the GRF, you will use both the CLI command set and the UNIX shell.

UNIX commands

To edit all .conf configuration files, you must be in the UNIX shell. The **sh** command opens the UNIX shell you use to edit configuration files.

At the super> prompt, enter the **sh** command:

super> sh

#

Type **exit** to leave the shell.

When you exit the UNIX shell, you can execute CLI-only commands. GRF-related UNIX commands are described in the *GRF Reference Guide*.

Change password

GRF systems are shipped with `Ascend` (*capital A*) preset as the password. If you have not already done so, Ascend recommends that you change this preset password now, before you begin to configure.

At the `#` prompt, type: `passwd`

```
# passwd
Old password:
```

Enter the old password:

```
Old password:
New password:
```

Enter the new password: (use 8 alphanumeric characters)

```
New password: . . . . .
Retype new password:
```

Enter the new password once more:

```
Retype new password: . . . . .
```

Add users

From the CLI, create User profiles for site users. Creating new user profiles is described later in this chapter.

Administrative LAN log on

After you attach the Ethernet connection from the control board to an administrative LAN, telnet from your administrative station to the GRF using its IP address.

At the `User:` prompt, enter: `netstar` (*all lowercase*)

At the `Password:` prompt, enter: `NetStar` (*capital N and S*)

```
User: netstar
Password: NetStar
#
```

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 `root` password you have 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).

Logging on to systems with RMS nodes

Initial configuration script

When you switch on the RMS node and GR-II, it powers on and begins to boot. The initial configuration script runs automatically. Here are the questions asked in the script that you need to be ready to answer:

```
Welcome to Ascend Embedded/OS system configuration...

Host name for this machine ?
Do you wish to configure the maintenance Ethernet interface ?
Which interface type ? (TP BNC AUI )
IP address of this machine ?
Netmask for this network ?
IP address of router ('none' for no default route)?
Do you wish to go through the questions again ?
Which region is this machine located in?
Which time zone ?
```

Next, you are prompted to change the local password for root:

```
New password (8 significant characters):
Retype new password:
passwd: updating passwd database
passwd: done

Initial configuration is complete...
Login as root and add users with the 'adduser' command...
```

This is the end of the script. If later you need to change these parameters, run the **config_netstart** script described at the end of this chapter.

The configuration script automatically saves any configuration information you have entered. If you later add or change configuration information, save those changes by executing a **grwrite -v** command.

root log on

The system continues to boot. When boot completes and you see this message, press the Enter key:

```
Press <Enter> to continue.

At the User prompt, type: root

User login: root

Password:
```

At the Password: prompt, type the new password you entered in the configuration script. If you did not change the password earlier, use the preset password, type: Ascend

```
Password: . . . . .
```

When a password is entered, it is not echoed (displayed) on the screen.

The `super>` prompt appears:

```
super>
```

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

CLI commands

Many system management and configuration commands are now available in the CLI. Enter a “?” to retrieve a list of CLI commands. CLI commands are described in the *GRF Reference Guide*.

To configure and manage the GRF or GR-II, you will use both the CLI command set and the UNIX shell.

UNIX commands

To edit the `.conf` configuration files, you must be in the UNIX shell. The **sh** command opens the UNIX shell you use to configure configuration files.

At the `super>` prompt, enter the **sh** command:

```
super> sh
#
```

Type **exit** to leave the shell.

When you exit the UNIX shell, you can execute CLI-only commands. GRF-related UNIX commands are described in the *GRF Reference Guide*.

Change password

Systems are shipped with *Ascend* (*capital A*) preset as the password. If you have not already done so, Ascend recommends that you change this preset password now, before you begin to configure.

At the `#` prompt, type: `passwd`

```
# passwd
Old password:
```

Enter the old password:

```
Old password:
New password:
```

Enter the new password: (use 8 alphanumeric characters)

```
New password: . . . . .
Retype new password:
```

Enter the new password once more:

```
Retype new password: . . . . .
```

Add users

From the CLI, create User profiles for site users. Creating new profiles is described later in this chapter.

Administrative LAN log on

After you attach the Ethernet connection from the RMS node to an administrative LAN, telnet from your administrative station to the GRF 400 or GR-II using its IP address.

At the User: prompt, enter: `netstar` (*all lowercase*)

At the Password: prompt, enter: `NetStar` (*capital N and S*)

```
User: netstar
Password: NetStar
#
```

At the User: prompt, enter: `netstar` (*all lowercase*)

At the Password: prompt, enter: `NetStar` (*capital N and S*)

```
User: netstar
Password: NetStar
#
```

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 `root` password you have 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).

Introduction to the command-line interface (CLI)

GRF profiles replace the `/etc/grinchd.conf` file. Variables from `grinchd.conf` are now fields in the 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.

CLI command table

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 won't 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**:

Table 1-1. CLI command list and descriptions

Command name	Permission level	Description
?	user	Help, or its alias, <code>?</code> , 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.
cd	user	Lists the fields of the current profile. (same as <code>list</code> 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.

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

Command name	Permission level	Description
exit	user	Exits a user from the command-line interface (CLI).
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, <code>gr.console</code> .
grdebug	system	Provides options to monitor system functionality.
grdump	system	Captures memory dump images.
gredit	system	Opens <code>filterd.conf</code> , <code>gated.conf</code> 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.

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

Command name	Permission level	Description
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.
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.
grmb	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.

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

Command name	Permission level	Description
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.
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.

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

Command name	Permission level	Description
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.

For details on each command, see the *GRF Reference Guide*.

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  09/28/97 13:54:18  admin
   92  09/28/97 13:54:18  default
  106  09/28/97 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
Ctrl-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).
Ctrl-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.
Ctrl-U	Erase entire line, but not the prompt. The cursor is placed immediately to the right of the prompt.
Ctrl-K	Erase the line to the right of the cursor position.
Ctrl-C	Use Ctrl-C only to terminate paged output. Other uses can have unexpected results.
Ctrl-M or Ctrl-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.
Ctrl-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.
Ctrl-N or ↓	Replace the current line with the next line from the line history. This sequence is valid only if Ctrl-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.
Ctrl-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).
Ctrl-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 Ctrl-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 Ctrl-N. This sequence is valid only if Ctrl-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 (Ctrl-D, Ctrl-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 Ctrl- 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 contains 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 /mae/jan.dumps/grdump 0}{fddi 20 /mae/jan.dumps/g+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi core +
config-spontaneous = off
keep-count = 2
super> list hw
hippi = { hipp 20 /var/portcards/grdump 0 }
fddi = { fddi 20 /var/portcards/grdump 2 }
rmb = { rmb 20 /var/portcards/grdump 3 }
atm-oc3-v1 = { atm-oc3-v1 20 /var/portcards/grdump 4 }
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.

Card 1

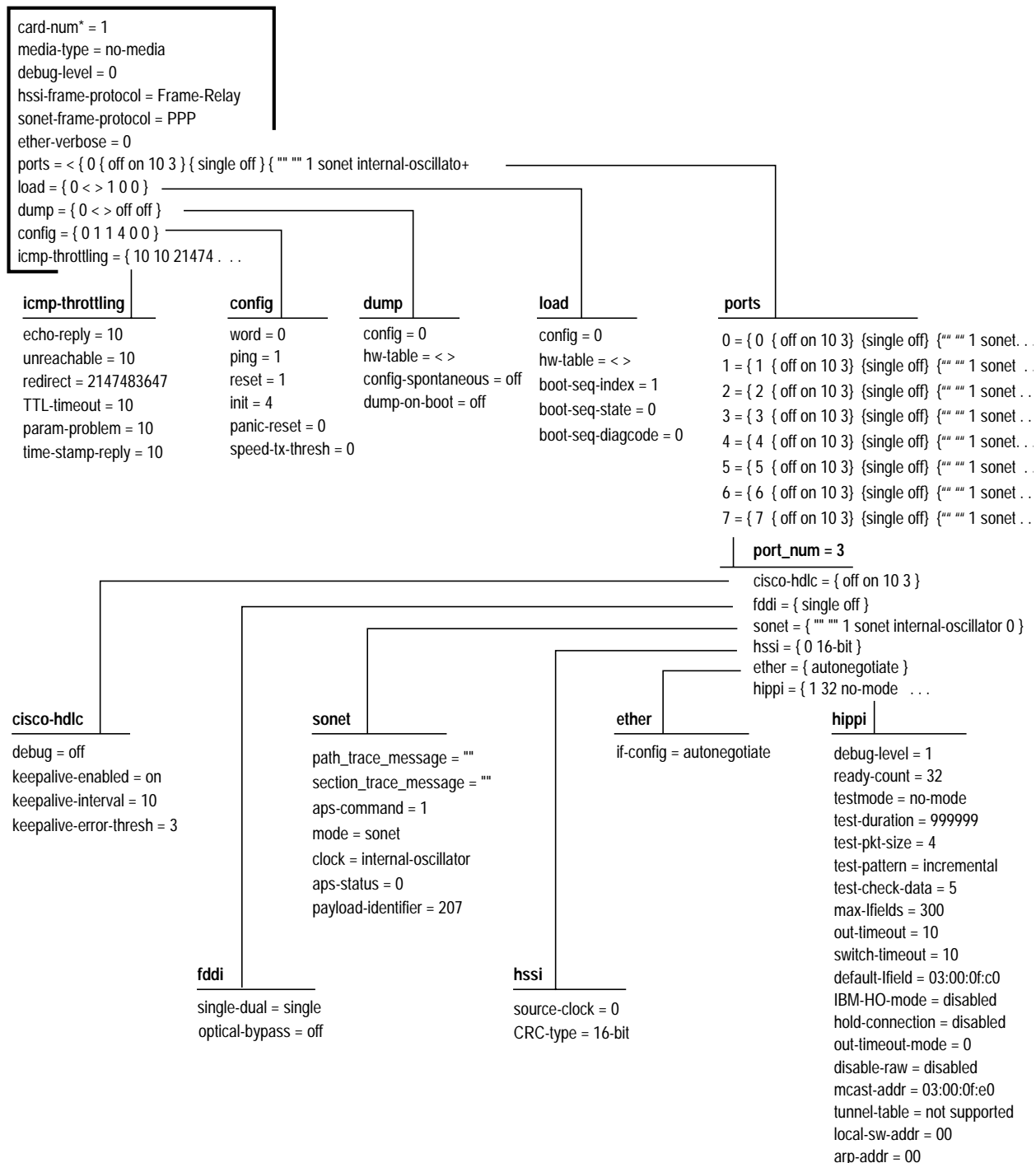


Figure 1-1. Diagram of Card profile levels

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
    -
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.

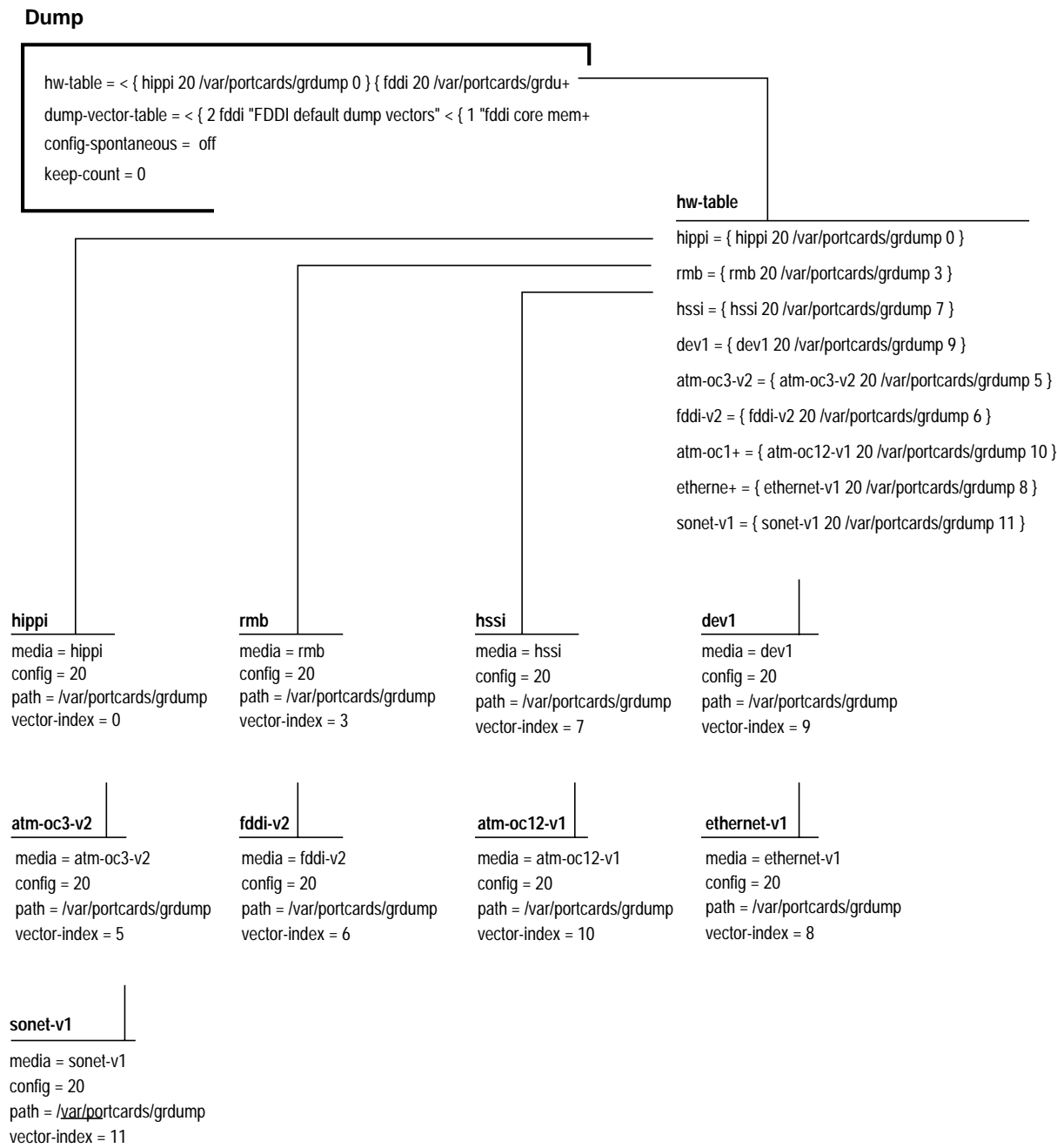


Figure 1-2. Dump profile: hw-table fields

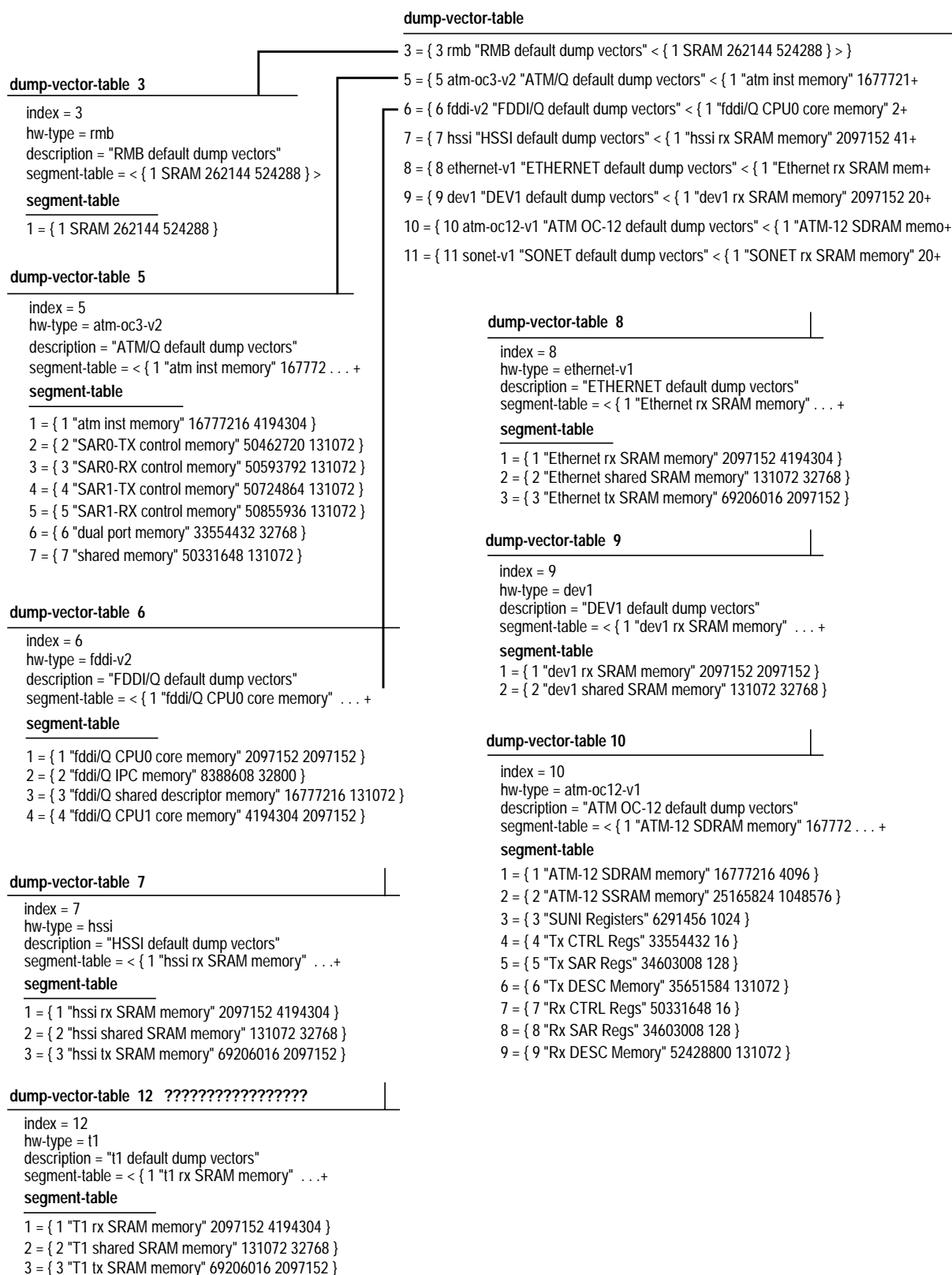


Figure 1-3. Dump profile: dump vector tables

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 = 0- sets number of dumps to keep plus the current and the first
                of the day, set to 0 or non-zero, default 0 saves 2 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 and, for normal operations, are not changed. Except as noted, fields are common across cards, FDDI/Q defaults are shown.

```
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.

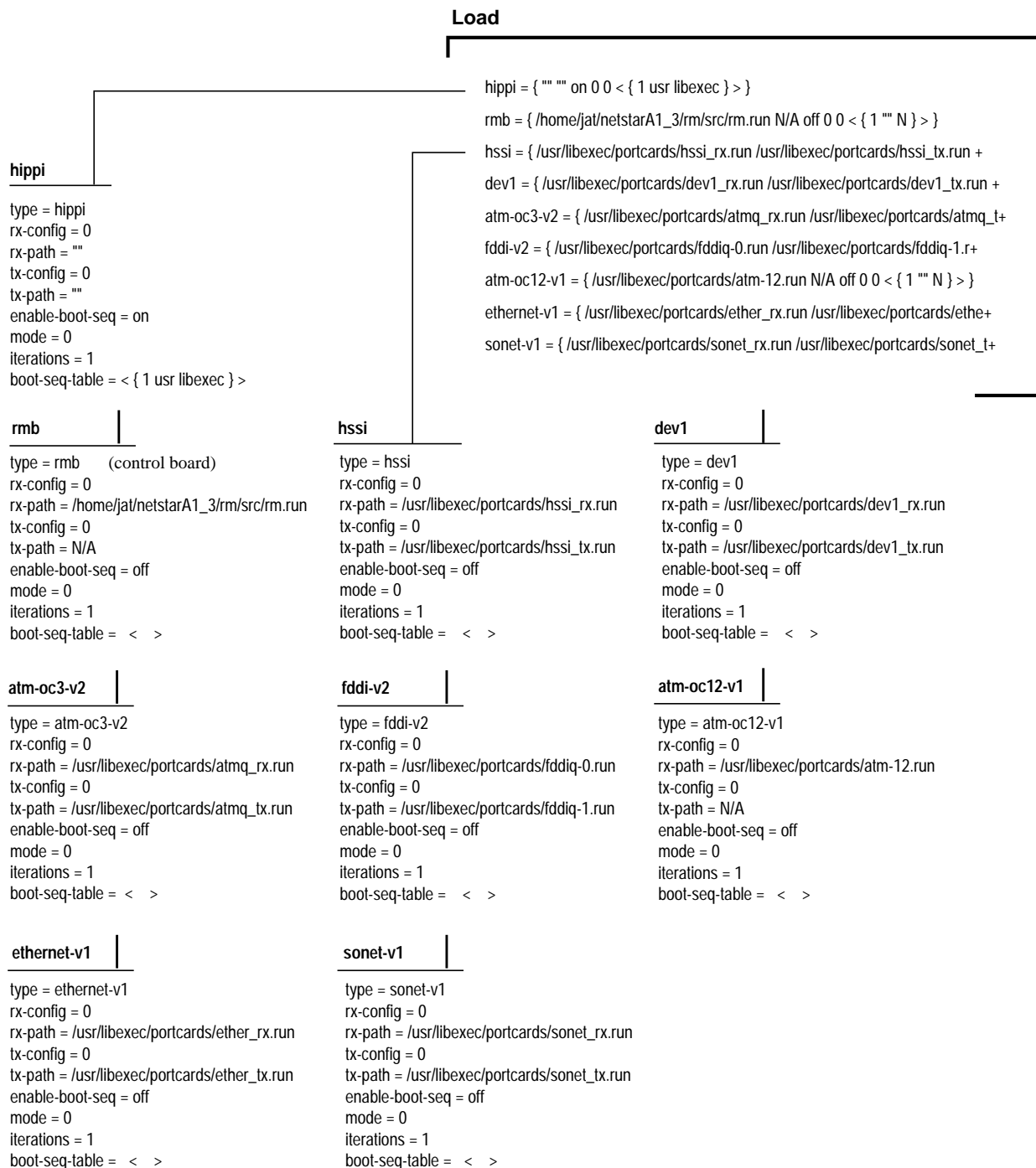


Figure 1-4. Diagram of Load profile levels

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 = < >-` empty field, use **new** command 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 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	<pre>os-level = 1.4.4 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</pre>
---------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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	<pre>os-level = 1.4.4 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</pre>
---------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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:

<code>os-level = 1.4.4</code>	- read-only, Ascend Embedded/OS release level
<code>hostname = grf.site.com</code>	- read-only, host name of this GRF
<code>ip-address = x.x.x.x</code>	- read-only, host IP address
<code>netmask = 0.0.0.0</code>	- read-only, system netmask field
<code>default-route = 0.0.0.0</code>	- read-only, system IP address or netmask field
<code>hippi-ifield-shift = 5</code>	- number of bit positions to shift an I-field, can be 4 or 5
<code>enable-congest = disabled</code>	- congestion management enabled or disabled
<code>num-slots = 4</code>	- read-only, number of slots in the chassis
<code>rmb-load-path = /usr/libexec/portcards/rm.run</code>	- control board (RMB) load path
<code>rmb-dump-config = 4</code>	
Default setting dumps control board (RMB) when it panics, other setting options are:	
<code>0x0001</code> - dump always (override other bits)	
<code>0x0002</code> - dump just the next time it reboots	
<code>0x0004</code> - dump on panic	
<code>0x0008</code> - dump whenever reset	
<code>0x0010</code> - dump whenever hung	
<code>0x0020</code> - dump on power up	
<code>physical-memory = 256</code>	- read-only, system memory in MB
<code>hardware-revision = "Not Available"</code>	- not currently used on GRF 400 or 1600
<code>chassis-revision = 1</code>	- read-only, GRF 1600 chassis revision level, not currently used on GRF 400
<code>xilinx-revision = 8</code>	- read-only, revision of hardware XILINX
<code>num-fans = 3</code>	- read-only, number of cooling fans, GRF 400 has 3, GRF 1600 has 2
<code>num-pwr-supply = 1</code>	- read-only, number of power supplies, 2 = redundant unit

User profile components

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

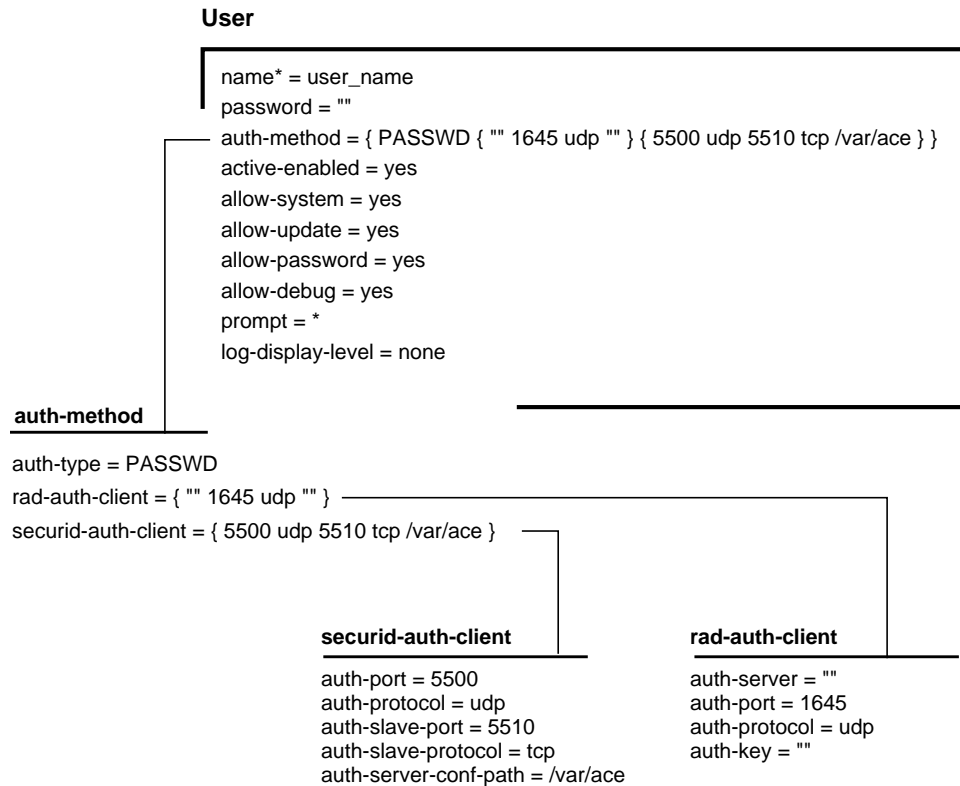


Figure 1-7. Diagram of User profile levels

User 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
- RADIUS
- 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	9/19/97	11:12:31	default

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

```
super> dir system
27  9/01/97 13:11:51  .
```

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

```
super> dir user
92  9/19/97 11:12:31  default
103 8/8/97  09:09:31  admin
106 6/16/97 11:19:31  super
88  11/22/97 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>
```

Check another profile before you change a setting

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 on 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
```

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  01/31/97 10:16:08  default
103 01/31/97 10:16:09  admin
106 01/31/97 10:16:10  super
99  02/03/97 15:27:00  frank

super> read user frank
USER/frank read

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

super> dir user
92  01/31/97 10:16:08  default
103 01/31/97 10:16:09  admin
106 01/31/97 10:16:10  super
99  02/03/97 15:27:00  frank
100 02/03/97 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 a profile.

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.

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 system parameters and other system options.

Unless otherwise noted, the information in this manual applies to GRF 400, GRF 1600, and 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.

Software release 1.4 does not support the first versions of the ATM OC-3c or FDDI media cards. These cards are informally called ATM Classic or FDDI Classic. Only the enhanced versions of these cards are supported, ATM/Q and FDDI/Q.

A note about configuring the GRF...

The GRF user interface has two aspects. One is an Ascend CLI interface in which system parameters are configured in Profiles.

The second is a UNIX interface in which system parameters are configured by editing configuration files. This shell is started with the CLI **sh** command, and supports standard and GRF versions of UNIX networking and configuration commands.

This chapter includes the following topics:

Configuration overview	2-3
Set up system logging	2-6
Log to external flash	2-7
Set up a syslog server	2-9
Use an NFS-mounted file system	2-12
Set up IP routing	2-13
Assign IP addresses - grifconfig.conf	2-13
Change GRF hostname	2-17
Enable telnet access - /etc/ttys	2-18
Configure SNMP (option)	2-19
Authentication options	2-24
TACACS+ (option)	2-24

Set RADIUS authentication (option)	2-26
Set securID	2-28
Static-only IP routing	2-31
Static routes using GateD	2-34
Enable GateD	2-36
Enable IP loose source routing	2-38
Selective packet discard (option)	2-39
ARP on the GRF	2-40
Things to remember when updating software	2-41
Save changes to release files	2-42
Test a new configuration	2-43
Do a system backup	2-44
Duplicate configurations among GRF systems	2-45
Save an alternate Load configuration	2-47
Run the config_netstart script	2-49
Reset the system to install files	2-50
Simple connectivity tests	2-52

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.

Configuration overview

The configuration procedures described in this chapter are performed after the automatic configuration scripts have run and you have logged in. Configuration tasks in which you edit a Card or other profile are done in the CLI, the command-line interface. Other tasks such as editing the `.conf` configuration files are done in a UNIX shell started from the CLI with the `sh` command.

Type `sh` at the prompt to start the UNIX shell and return the UNIX prompt:

```
super> sh
#
```

1. Configure IP addresses and operating options *Chapter 2*

- Set up system logging
- Set up IP routing
- Change GRF hostname
- Configuring SNMP (optional)
- Authentication options: TACACS+, RADIUS, securID
- static routes using GateD
- Enable GateD
- Enable IP loose source routing
- Enable host telnet access - `/etc/ttys`
- Selective packet discard on the GRF
- ARP on the GRF
- Things to remember when updating software
- Management tasks:
 - Testing a new binary*
 - Testing a new configuration*
 - Doing a system backup*
 - Duplicating configurations among GRF systems*
 - How to save an alternate Load configuration*
 - Resetting the system to install files*

2. Configure media cards and protocols *Chapters 3–10*

After installing the system-oriented information, configure cards and protocols.

- ATM OC-3c
- FDDI/Q
- HIPPI
- HSSI
- Ethernet (10/100Base-T)

- SONET OC-3c
- ATM OC-12c

3. Configure network services *Chapters 11-15*

After configuring the media cards and verifying the interface connections, usually with pings, configure network services as needed.

- Configure the Ascend Tunnel Management Protocol (ATMP)
- Set up filtering services
- Configure BGP (BGP4) in the `gated.conf` file
- Configure IS-IS (on FDDI/Q, ATM/Q, Ethernet, HSSI cards)
- Set up a filter to enable Controlled-Load for specific media card
- Set up transparent bridging

4. Save configuration and test

Save/archive a copy of your configuration files to another flash device or floppy, then:

- **grwrite -v** command
- Reset the media cards
- Test GRF operation and configuration settings

At this point, the GRF will be able to switch and route data.

Configuration files and their uses

This is an alphabetized list of `/etc` configuration files and their application:

<code>aitmd.conf</code>	- defines parameters for ATMP (VPN tunneling protocol)
<code>bridged.conf</code>	- defines system bridging services
<code>filterd.conf</code>	- defines system filtering services
<code>gated.conf</code>	- enables dynamic routing functions
<code>grarp.conf</code>	- maps IP addresses to physical hardware addresses
<code>grass.conf</code>	- manages IP services
<code>gratm.conf</code>	- configures ATM media cards, including PVCs, SVCs, peak and sustainable cell rates, UNI signalling, SONET and SDH modes
<code>grfr.conf</code>	- configures Frame Relay on HSSI and SONET cards
<code>grifconfig.conf</code>	- identifies each logical interface on a media card
<code>grlamap.conf</code>	- maps HIPPI logical addresses to media cards
<code>grppp.conf</code>	- configures the Point to Point Protocol on HSSI and SONET cards
<code>grroute.conf</code>	- sets static routes
<code>snmpd.conf</code>	- enables SNMP capabilities
<code>syslog.conf</code>	- configures remote logging of log files via syslogd

Refer to the *GRF Reference Guide* for copies of all configuration files.

Use grwrite to save config changes before boot

In the GRF, before you reboot, you must first use the **grwrite** command to save the changed configuration files to a flash device. This is the only way to make changes permanent and able to survive a reboot on the GRF system.

When using **grfins** to install a new version, issue a **grwrite** before running the **grfins** command to make sure any modified files are saved. When **grfins** completes, reboot as soon as possible with a **reboot -i** command.

In RMS node systems, changes to configuration files are made permanent by saving the files and rebooting.

Set up system logging

System memory restrictions on the GRF control board require that logged information be sent to external storage. If external storage is not configured to receive log entries, log entries are not saved. When the GRF is first installed and started up, logging does not automatically start, one of the following options must be configured:

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

Procedures to set up each option begin on the next page.

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

(External logging procedures do not apply to RMS node systems. These system log locally to their internal hard drive.)

Panic dumps to external flash

When a media card panics and a formatted flash device is plugged into PCMCIA slot A, a copy of the dump is automatically saved to the external flash in a directory called `/portcards`.

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 `grclean.logs.conf` file is renamed to `grclean.logs.conf.old`, and a new copy of `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 `grclean.logs.conf`, the upgrade has no effect. However, if you did change `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.

Log to external flash (option)

Use this procedure to configure system logging and saving media card dumps to an external flash device inserted in PCMCIA slot A. The procedure need only be done once, rebooting or updating to new releases does not affect the configuration.

The **iflash** command determines the geometry of any PCMCIA Type-II or Type-III ATA disk device, and formats the disk for use in a GRF

Note: Location of log and media card dump paths may change in future releases.

The procedure formats and initializes an external flash (`/dev/wd1a`), temporarily mounts it on `/mnt`, creates subdirectories and symbolic links, and creates a permanent site file for storing the symbolic links.

- 1 Insert the PCMCIA disk into a slot on the control board.
- 2 Log in as root, start the UNIX shell, and execute these commands from the shell:

```
prompt> sh
#
# cd /
# iflash -f -e
# mountf -e -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 -e
```

- 3 Edit the file `/etc/fstab` and add this line as shown in the excerpt below:

```
/dev/wd1a      /var/log      ufs      rw      0 2

# 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/wd1a      /var/log      ufs      rw      0 2
```

- 4 Edit the file `/etc/syslog.conf` to specify the location where the logs will be kept. Uncomment the local log configuration lines in the "Log messages to Network"

section by removing `#net#` from each line and specify `/var/log` as the directory for each log:

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
```

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

- 5 Modify `/etc/grclean.conf` and `/etc/grclean.logs.conf` to reflect the new log location. The entries should look like the following:

```
*****
* Log files that used to be archived by the /etc/{daily|weekly|monthly}
* scripts.
*****
size=10000
logfile=/var/account/acct
size=10000
#logfile=var/log
logfile=var/log
```

- 6 Save all changes and reboot:

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

- 7 Verify that the PCMCIA interface and device are up:

```
# csconfig -a
```

The `/var/portcards` directory only contains media card dump files. These include the dumps from media card panics and dumps created when automatic dumping is selected via the **grreset -D** command (media card dumps when it comes back up). The `/var/crash` directory contains dumps from BSD kernel crashes.

Set up a syslog server (option)

One way to log over the network is to use **syslog** and set up a remote **syslogd** host. You can configure remote logging during initial installation via the configuration script or you can change to remote logging at any time.

You begin to enable network logging in the first-time power on configuration script as described in the *GRF 400/1600 Getting Started* manual, or, if you are not powering on the GRF for the first time, you can directly edit the `/etc/syslog.conf` configuration file and set up a remote **syslog** server. The remote **syslog** or logging server is another node on your LAN that runs the **syslog** daemon, **syslogd**.

The GRF `syslog.conf` file configures the GRF to send log messages to a **syslog** server on your local network. These logs are the most useful to users:

- `gr.console`, the most helpful log because it contains status and event-reporting for the operating system and media cards.
- `gr.boot`, contains boot and load messages from media cards' boot software.
- `grinchd.log`, keeps a record of configuration information sent to and from cards.
- `mib2d.log`, collects the SNMP subagent log messages.

Follow these steps to configure the GRF and the **syslog** server for network logging:

- 1 On the GRF, open the `/etc/syslog.conf` file and uncomment these lines in the “Log messages to Network” section by removing `#net#` from each line:

```
#net#*.err;kern.debug;auth.notice;mail.crit
@server.domain.com
#net#*.notice;kern.debug;lpr,auth.info;mail.crit
@server.domain.com
#net#cron.info                                @server.domain.com
#net#local0.info                             @server.domain.com
#net#local1.info                             @server.domain.com
#net#local2.*                                @server.domain.com
#net#local3.*                                @server.domain.com
#net#local4.*                                @server.domain.com
#net#local5.*                                @server.domain.com
```

Still in `/etc/syslog.conf`, replace “`server.domain.com`” with the IP address or domain name of the target host (logging server) that will receive the log messages.

- 2 On the GRF, add the IP address and host name of the logging server to `/etc/hosts`.
- 3 On the GRF, run the **grwrite** command to save the changes to the `/etc` configuration files (`/etc/syslog.conf` and `/etc/hosts`).
- 4 On the **syslog** server, add these file names to the `/etc/syslog.conf` file:

```
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
```

- 5 On the **syslog** server, use the **touch** command to create the log files in the server's `/var/log` directory or wherever log files normally reside on the server:

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

- 6 On the **syslog** server, kill and restart **syslogd**.

syslogd must be running with the `-i` option to enable "internet listening." If **syslogd** is not running with `-i`, kill it and restart it. The **syslogd** pid can be found in the file `/var/run/syslog.pid`. You will lose any messages sent to **syslog** during the kill/restart processes.

Note: `/etc/rc` can be edited to get `syslogd -i` upon boot.

- 7 On the GRF, restart **syslogd**.

Because **syslogd** is usually run before the domain name server (**named**) at boot time, you need to add an entry to `/etc/hosts` that describes the host you want to send to. Here is a sample entry in `/etc/hosts`:

```
222.222.11.93          server.domain.com
```

Now kill **syslogd** and restart it with the `-i` option. The `-i` can be added to `/etc/rc` to turn this on at boot time.

Run **grwrite** to save changes made to the `/etc` files. Note that the **grconslog** command does not differentiate among multiple log files sent to a single remote **syslog** server from multiple GRF systems.

This completes the network logging configuration. An excerpt from the `/etc/syslog.conf` file follows:

syslog.conf file

This excerpt from the `/etc/syslog.conf` file applies to GRF 400 and GRF 1600 systems:

```
# syslog.conf,v 1.3.28.2 1997/01/06 20:01:00 wdp Exp $
#
*.err;kern.debug;auth.notice;mail.crit      /dev/console
*.notice;auth.debug                          root
*.emerg                                      *
#
# To enable the logging of system messages on GRF systems, edit the
# entries in the "Log messages to Network" section below.
#
#   - uncomment the lines in the "Log messages to Network" section
#     of this config file (just below these instructions)
#
#   - change "server.domain.com" to the name of a syslog server on
#     your local network to which this router may send log
messages
#
```

```
# - add the IP address of log server and its host name to
# /etc/hosts on this GRF system.
#
# - run the "grwrite" command to save your changes to
# /etc/syslog.conf and /etc/hosts
#
# - add the following lines to the /etc/syslog.conf on your log
# server: (do not include the # from the first column of this
# file,
# ie., add "local5.* /var/log/mib2d.log" not "#local5.*
# /var/...")
#
#          #
#          # Syslog configuration for syslog server systems
#          # GRF-specific log files (from GRF systems over the
network)
#          #
#          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
#
# - kill and restart syslogd on your syslog server machine
after
#          making sure that the log files added to the config file
exist
#          on the log server machine exist (use the "touch" command
#          to create them if they do not exist, then restart
syslogd
#          on the syslog server machine).
#
# - kill and restart syslogd on the GRF router (or reboot the
GRF).
#
# To uncomment, remove "#net#" from each line in this section
#
# Log messages to Network
#
#net#*.err;kern.debug;auth.notice;mail.crit
@server.domain.com
#net#*.notice;kern.debug;lpr,auth.info;mail.crit
@server.domain.com
#net#cron.info                  @server.domain.com
#net#local0.info                @server.domain.com
#net#local1.info                @server.domain.com
#net#local2.*                   @server.domain.com
#net#local3.*                   @server.domain.com
#net#local4.*                   @server.domain.com
#net#local5.*                   @server.domain.com
```

Use an NFS-mounted file system (option)

On the GRF 400 and GRF 1600, the maintenance Ethernet interface (usually `de0`) must be used for NFS-mounted file systems.

Note: NFS mounting CANNOT be done through a media card interface (for example, `gf0yz`). The maintenance interface is specified in the `/etc/netstart` file as the value assigned to the variable "`iface`". This variable defines the external connection to the GRF and is set when the GRF is initially installed.

Note: Programs writing to an NFS-mounted file system will block while the NFS connection to the remote file system is dropped.

Set up NFS on the GRF

You must add the appropriate host and IP information for any NFS server you are using into the `/etc/host` file if either of the following conditions apply:

- The name server(s) in `/etc/resolv.conf` point out one of the media card interfaces and are thus unavailable when the mount occurs.
- The name server(s) in `/etc/resolv.conf` point to a network that is reachable only through a route off the administrative Ethernet and only when GateD is being run. That is, if GateD is not active and a route entry exists in `grroute.conf`, the `/etc/host` entries are not needed.

Set up IP routing

The GRF is a crosspoint switch that performs IP routing. This section describes the configuration steps for IP routing functions. Required steps are presented first, options are described next.

Assign IP addresses - grifconfig.conf

Edit the `grifconfig.conf` file to assign an IP address to each logical GRF interface. You also can provide other information about the logical IP network to which that interface is physically attached. The **grifconfig** script reads the `grifconfig.conf` file.

Each logical interface is identified in `grifconfig.conf` as to its:

- interface name (*a GRF convention, defined below*)
- Internet address
- netmask
- broadcast/destination address
- arguments field (optional)

The GRF 400 has four media card slots, the GRF 1600 and GR-II have 16. Because of this, some examples in this manual use slots 0–3, others use 0–15. Particular slot numbers are not significant in examples.

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

name	address	netmask	broad_dest	arguments

Interface name

The GRF interface name has five components that describe an individual logical interface in terms of its physical slot location in the chassis, and its specific and virtual locations on a media card.

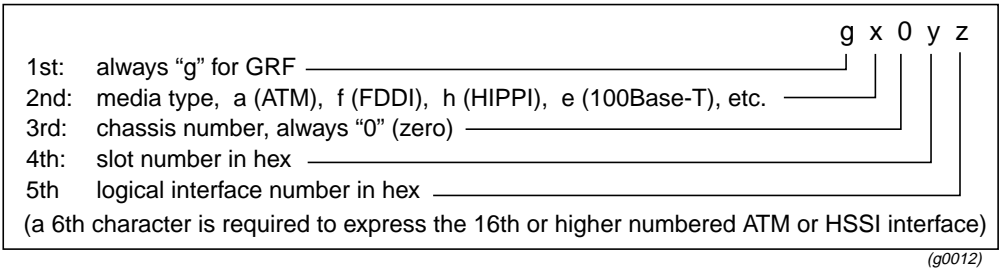


Figure 2-1. Components in the GRF interface name

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

ATM interface names look like:	ga037f, ga0281, ga01ff
FDDI interface names look like:	gf030, gf021, gf012, gf003
HIPPI interface names look like:	gh030 (only the slot # changes)
HSSI interface names look like:	gs030, gs021, gs0180
SONET OC-3c interface names look like:	go030 (only the slot # changes)
10/100Base-T interface names look like:	ge030, ge026, ge017
T1 interfaces names look like:	gm070-gm079 (the T1 card in slot 7)

Internet address

An Internet 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)

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, dash (-) is entered in the netmask column as a placeholder.

Broadcast / destination address (optional)

The broad_dest address is the 32-bit address for this network. 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. If you want to use the arguments field but do not need a broadcast or destination address, enter a dash as the address placeholder.

When you configure a logical interface on a point-to-point media such as HIPPI or ATM, the destination IP address is entered in the broad_dest address field. An entry in the broad_dest address field for a HIPPI or ATM interface 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

This optional field is currently used to specify MTU values on a logical interface basis, MTU 1046, for example. This field specifies ISO when an ISO address is being added to an interface's IP address. Refer to the **ifconfig** man page for a description of argument options.

Maximum transmission unit (MTU)

Maximum Transmission Unit (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.

Specify an MTU value as `mtu xyz`.

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, SONET, and T1 cards are:

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

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:

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).

Create a loopback alias

To create an alias IP address for the loopback (100) interface, specify the IP addresses for the 100 interface in `/etc/grifconfig.conf` as shown in the following example:

100	127.0.0.1	255.0.0.0
100	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 100. (All entries must be lower case.)

Defining an ISO address (IS-IS)

IS-IS is supported on ATM OC-3c, Ethernet, FDDI, HSSI, and SONET media cards. One step in configuring the IS-IS protocol is to 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
```

Configuration information and IS-IS examples are in the “IS-IS Configuration” chapter in this manual. Also, the IS-IS statement is described in the GateD chapter of the *GRF Reference Manual*.

Put grifconfig changes into effect

To install a new *grifconfig.conf* file, you either re-run the **grifconfig** program or reset the media card:

```
# grifconfig gx0yz
```

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

```
# grreset slot_number
```

Refer to the *GRF Reference Guide* for more information about using **grreset**.

Change GRF hostname

The factory-preset GRF hostname should be changed on each system.

You can use the **config_netstart** script described in this chapter, or you can change the preset GRF hostname in the `/etc/netstart` and `/etc/hosts` files. To edit these files, start the UNIX shell and use an editor to open each file.

To change `grf.ascend.com` to `new.host.com`:

In `/etc/netstart`, edit the line:

```
hostname=grf.ascend.com
```

to read the new host name:

```
hostname=new.host.com
```

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

In `/etc/hosts`, edit the line:

```
###.###.###.### grf.ascend.com
```

to read the new host name:

```
###.###.###.### 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.

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
```

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:

- 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 RFC 1213 groups:

- address translation (deprecated in MIB-II)
- egp
- oim (no support planned)

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 `hippisswShiftCount`, `hippisswPortNumber`, `hippisswPortTable`, `hippisswLTable`.

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 |
| - rfc1512.smi | FDDI MIB |
| - rfc1573.smi | Extended Interface MIB |
| - rfc1227.smi | SMUX MIB |
| - netstar.smi | Enterprise MIB |
| - hippimib.smi | HIPPI End-point MIB |
| - hswmib.smi | HIPPI Switch MIB |
| - rfc1471.smi | PPP MIB |
| - rfc1473.smi | PPP MIB |

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 **vi** 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:\n\n    :auth=tacacsplus:\n    :tacacs-server=x.x.x.x:\n    :tc=default:
```

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

```
# TACACS+ server\n\ntacacs          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 Ascend A1.2 releases must edit the `/etc/login.conf` file. First-time installations of an A1.2 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
```


Static-only IP routing (option)

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

Column one is the IP address of the target destination host or network

Column two holds the netmask for the network route destination
(destination netmask for host routes must be 255.255.255.255)

Column three 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.

Putting changes into effect

Changes to `/etc/grroute.conf` after the first-time installation will take effect only after the file is reloaded and the media card(s) reset. Use the **greset** command.

Error checking

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

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 such a 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:

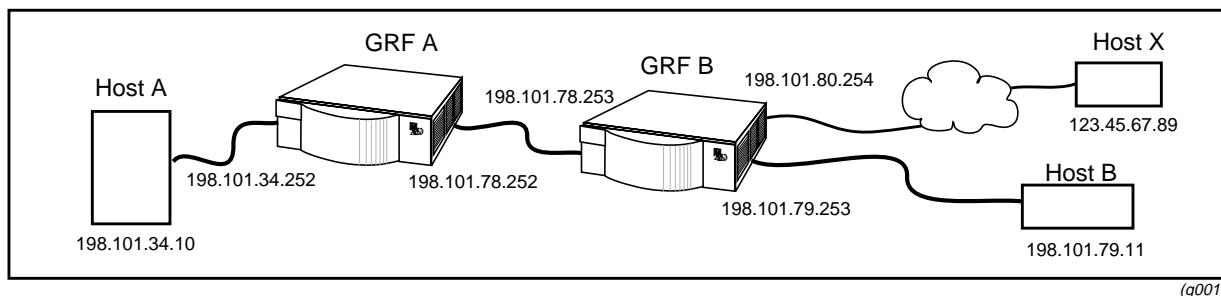


Figure 2-2. Illustration for static routing configuration

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

Using route add or grroute.conf

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
```

Using GateD

If GateD is running on both GRFs, the information about Host A, B and X is shared dynamically. They can also be specified as static routes in the `gated.conf` file. 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 this GateD Static statement:

```
static {
    198.101.79.0 mask 255.255.255.0 gateway 198.101.78.253 ;
    host 123.45.67.89 gateway 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 this GateD Static statement:

```
static {
    198.101.34.0 mask 255.255.255.0 gateway 198.101.78.253 ;
};
```

Displaying static route tables

The recommended way to view static route tables per media card is to use the **grrt** command.

Using grrt

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

Static routes using GateD (option)

When a network is using dynamic routing via GateD, and also needs static (permanent) routes, configure static routes using GateD. Replace any previously hand-entered static routes with a GateD Static statement. GateD removes manually-added static routes (those using **routeadd** or added to `grroute.conf`) from the master route table as it maintains the table.

Refer to the *GRF Reference Guide* for GateD configuration documentation.

GateD static statement

(Information in this section is from the Static statement in the *GRF Reference Guide*.)

Static statements define the static routes used by GateD. A single static statement can specify any number of routes. The static statements occur after protocol statements and before control statements in the `gated.conf` file. Any number of static statements can be specified, each containing any number of static route definitions. These routes can be overridden by routes with better preference values.

```
static {  
    ( host host ) | default |  
    ( network [ ( mask mask ) | ( ( masklen | / ) number ) ] )  
        gateway gateway_list  
        [ interface interface_list ]  
        [ preference preference ]  
        [ retain ]  
        [ reject ]  
        [ blackhole ]  
        [ noinstall ] ;  
    ( network [ ( mask mask ) | ( ( masklen | / ) number ) ] )  
        interface interface  
        [ preference preference ]  
        [ retain ]  
        [ reject ]  
        [ blackhole ]  
        [ noinstall ] ;  
};
```

where:

`interface interface_list`

When this parameter is specified, gateways are only considered valid when they are on one of these interfaces. See the section on interface list specification for a description of `interface_list`.

`preference preference`

This option selects the preference of this static route. The preference controls how this route competes with routes from other protocols. The default preference is 60.

`retain`

Normally **gated** removes all routes except interface routes from the kernel forwarding table during a graceful shutdown. The `retain` option can be used to prevent specific static

routes from being removed. This is useful to insure that some routing is available when **gated** is not running.

reject

Instead of forwarding a packet like a normal route, **reject** routes cause packets to be dropped and unreachable messages to be sent to the packet originators. Specifying this option causes this route to be installed as a reject route. Not all kernel forwarding engines support reject routes.

blackhole

A blackhole route is the same as a reject route except that unreachable messages are not supported.

noinstall

Normally the route with the lowest preference is installed in the kernel forwarding table and is the route exported to other protocols. When **noinstall** is specified on a route, it will not be installed in the kernel forwarding table when it is active, but it will still be eligible to be exported to other protocols.

Display GateD route tables

The recommended way to view GateD route tables is to use the GateD State Monitor (GSM).

When GateD is running, you can use the GateD State Monitor **gsm** command to examine the system route table.

The GateD State Monitor is an interactive program. To access GSM, users open a telnet connection to the machine running GateD on TCP port 616. To telnet to the GSM interface, one must use the password from the administrative 'netstar' account. The default password = "NetStar". If you have changed the password, use the new one.

If GateD is running on 192.22.22.22, here is the telnet command and login sequence:

```
# telnet -a 192.22.22.22 616
Trying 192.22.22.22...
Connected to 192.22.22.22.
Escape character is '^]'.
Password? ----- ( admin password, for example, NetStar )
lGRF Gated State Monitor. Version GateD R3_5Beta_3;
Path:
GateD-station.site.com>
```

Note that the GSM prompt includes the machine's domain name. Use the **show** command to list available command options. To obtain protocol-specific information, use **show** with the protocol, as in **show ip**.

```
GateD-station.site.com> show ip all
```

Here are commands for other protocols, and one to display the routes from all protocols:

```
GateD-station.site.com> show all all
GateD-station.site.com> show rip all
GateD-station.site.com> show static all
GateD-station.site.com> show ospf all
GateD-station.site.com> show bgp all
GateD-station.site.com> show isis all
```

Enable GateD (option)

The supported version of GateD is 3.5b3, a version with additions and enhancements to BGP developed by Ascend. See the chapter, “Introduction to GRF BGP,” in this manual for descriptions of 3.5b3 additions and features.

The *GRF Reference Guide* includes the 3.5b3 version of the *GateD Configuration Guide*, the standard source of GateD information. The *Guide* is updated to reflect the new features introduced by Ascend. The majority of changes have been made to the BGP protocol support.

The `/etc/gated.conf` file configures the GRF to exchange dynamic routing updates via RIP and OSPF with other routing agents. GateD builds and maintains the system’s master route table.

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 sections of the GateD documentation in the *GRF Reference Guide* into 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 up 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 Reference Guide*:

```
# gdc reconfig
```

If GateD has not been started, you will get a message. Use this command to start **gated**:

```
# gdc start
```

Assign an alias as routerid

Ascend strongly recommends that you configure an alias address on the loopback interface of your router and use this address as the routerid in `gated.conf`. Also, the alias should be used as the local address (**lcladdr**) in peer statements for BGP. This will enable proper route calculations across all interfaces.

To configure an alias of the loopback interface, add an interface definition to `grifconfig.conf` as shown in the example here. The example sets the IP address of the loopback interface to 127.0.0.1 with an alias of 210.210.210.1.

It is important to maintain the 127.0.0.1 address on the loopback interface. Enter the 127.0.0.1 line so it appears before the alias in `grifconfig.conf`.

name	address	netmask	broad_dest	arguments
lo0	127.0.0.1	255.255.255.0		
lo0	210.210.210.1	255.255.255.0		

In this configuration, **netstat -in** produces the following output:

```
# netstat -in | grep lo0
lo0      1536 <link3> 909 0 909 0 0
lo0      1536 127/24 127.0.0.1 909 0 909 0 0
lo0      1536 210.210.210 210.210.210.1 909 0 909 0 0
lo0      1536 <GRIT> 0:0x48:0 909 0 909 0 0
```

In this configuration, the **ifconfig lo0** command produces:

```
# ifconfig lo0
lo0: loop flags=8009<UP,LOOPBACK,MULTICAST> mtu 1536
inet 127.0.0.1 netmask 0xffffffff00
grit 0:0x48:0
```

Enable IP loose source routing (option)

Loose source routing is an IP processing option. In regular source IP routing, the source provides every hop on the route to the destination address. Using source routing, a source can specify an exact route.

Using 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, loose source routing is disabled.

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

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

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

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


Selective packet discard (option)

Selective packet discard can be enabled on the 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 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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 specified per media card, 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.

ARP 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.

Use `grarp` command to display ARP information

To display ARP information, use the **`grarp -i interface hostname`** command. Its other 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 do not get routed.

Pinging opposite interface to invoke ARP

Given that a 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.

Pinging 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, 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.

Things to remember when updating software

- 1 Which files are updated , which are replaced ?
 - /etc configuration files are updated if needed
 - all other files are replaced
- 2 Check the current software version you are running:

```
# getver
Current Revision: 1.4.2   Version: default
```
- 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

# grwrite --perm
```

Changes to /etc/services are overwritten

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**.

On RMS node systems, your changes to /etc/services are saved and archived by the **grc** command.

Warning: 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 as you will need to manually reinstate them when you upgrade.

Test 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 --next
```

You can specify **grsite** to save the change to the currently-running version, it will continue to run:

```
# grsite --perm
```

Files saved using **grsite** without the **--perm** option are not carried forward during an upgrade when the **grfins** command is used.

Test 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 1.4.6,default (in this example) and save a copy of the current as 1.4.6,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 -si=1.4.6,default -di=1.4.6,newstuff_config
# setver 1.4.6,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, 1.4.6,default, use **setver** again and reboot:

```
# setver 1.4.6,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 **setver** to check:

```
# setver
```

Note:

If the 1.4.6,newstuff_config software performs as you intend, there is no reason you cannot continue to run this software.

Do a system backup

This section provides a brief backup process.

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. **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 -si=1.4.x,default -de=1.4.x,120797_backup
```

Later, you can use **grsnapshot** to restore the operating system from the external storage:

```
super> grsnapshot -se=1.4.x,120797_backup -di=1.4.x,default
```

You can also archive files to an NFS mounted file system using this command and the compress the directory afterward:

```
# grsnapshot -si=1.4.x,default -dd=/grf/backups/120797
```

RMS node systems

First, save the current `/etc` configuration directory with **grc**:

```
super> grc save -a archive_file
```

```
super> grc restore -a archive_file -d directory
```

Refer to the *GRF Reference Guide* for more information about the commands used in this section.

Duplicate configurations among GRF systems

As shown in Figure 2-3, the GRF control board supports the use of external 85MB and 175MB flash devices in PCMCIA slot A. (Slot B supports a modem disk.)

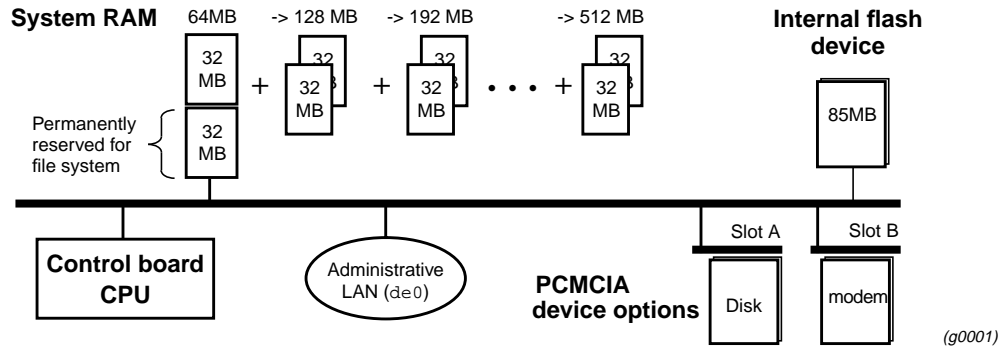


Figure 2-3. Support for external flash devices in PCMCIA slot A

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 release 1.4 default system, *1_4, test* would mean release 1.4 test system.

On GRF A :

- 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 -si=1_4,default -de=1_4,default`
- 4 Remove the flash device from GRF A.

On GRF B :

- 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 -se=1_4,default -di=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.

The GRF can panic when it tries to access a PCMCIA device that is now removed. 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.

Save an alternate Load configuration

This procedure describes how to switch the run time code for all media cards of a specific type by creating and running with a new Load profile. The last step describes how to restore your original binaries. The example changes run time code for the Ethernet cards.

- first, save your current Load profile configuration
- second, 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
- if necessary, restore the original Load profile

Begin by saving the current Load profile into memory so it can be reloaded if necessary:

```
super> save original.load load
```

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/xlload.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 +
devl = { /usr/libexec/portcards/devl_rx.run
        /usr/libexec/portcards/devl_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+
tl-v1 = { /usr/libexec/portcards/tl_rx.run.run
         /usr/libexec/portcards/tl_tx.run+
```

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 = < >
```

Edit the receive and transmit run-time code path names to `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 a unique abbreviation for write)
LOAD written
```

Save the test pathnames in the new Load profile named `new_ether`:

```
super> save new_ether.load load
save to new_ether.load successful
```

Verify the Ethernet test pathnames in the new `new_ether` Load profile:

```
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 = < >
```

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
```

Verify the contents of the `new_ether` Load profile:

```
super> list
hippi = { " N/A on 0 1 <{1 /usr/libexec/portcards/xlload.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+
fdi-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+
t1-v1 = { /usr/libexec/portcards/t1_rx.run.run
         /usr/libexec/portcards/t1_tx.run+
```

If necessary, restore your original Load profile:

```
super> load original.load
```

Run the config_netstart script

When you need to change the IP address or other configuration parameter, use the **config_netstart** script. **config_netstart** operates on all GRF and GR-II routers.

config_netstart can configure the host IP address and Ethernet interface. This script eliminates individually editing `/etc/hosts`, `/etc/hostname.txt`, `/etc/resolv.conf`, and so on.

The script enables you to configure the GRF or GR-II to a LAN. You can also specify 10BaseT (TP), AUI, or BNC cable types. GRF 400 and GRF 1600 connections operate at 10Base-T or 100Base-T, autosensing the connection rate.

To start the script, enter:

```
# config_netstart
```

The script is a series of questions briefly reviewed below.

Host name for this machine ?

*Enter a name, the default name set during manufacture
will appear in brackets.*

Do you wish to configure an ethernet interface ?

*This is the optional administrative LAN, ef0 on RMS node, de0 on GRF 400.
Type no if you are isolating the network.*

Which interface ?

You must enter "1" to use ef0 on the RMS node , de0 on GRF 400.

Which interface type ?

Enter the preferred type from among TP, BNC, AUI .

IP address of this machine ?

Enter IP address for ef0 or de0, as appropriate.

Netmask for this network ?

A default is provided.

IP address of router ?

Enter "none", default routes are not recommended.

At the end, the **config_netstart** script reviews your entries and allows you to make changes.

Reset the system to install files

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 (filtering and dynamic routing).

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.

```
# grwrite -v
```

To save an alternate configuration on the internal flash based upon the currently-running configuration on the internal flash device:

```
# grsnapshot -si -di=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
```

Use the **-a** option instead of **-F** to archive to a specified file. Here is the default list of files **grc** archives:

<code>etc/aitmd.conf</code>	<code>etc/gritd.conf</code>	<code>etc/namedb</code>
<code>etc/bridged.conf</code>	<code>etc/grlamap.conf</code>	<code>etc/netstart</code>
<code>etc/crontab</code>	<code>etc/group</code>	<code>etc/networks</code>
<code>etc/fstab</code>	<code>etc/grppp.conf</code>	<code>etc/passwd</code>
<code>etc/filterd.conf</code>	<code>etc/grpvc.conf</code>	<code>etc/printcap</code>
<code>etc/gated.conf</code>	<code>etc/grroute.conf</code>	<code>etc/pwd.db</code>
<code>etc/gettytab</code>	<code>etc/grstart</code>	<code>etc/rc</code>
<code>etc/grarp.conf</code>	<code>etc/hostname.txt</code>	<code>etc/rc.local</code>
<code>etc/gratm.conf</code>	<code>etc/hosts</code>	<code>etc/resolv.conf</code>
<code>etc/grclean.conf</code>	<code>etc/hosts.equiv</code>	<code>etc/services</code>
<code>etc/grclean.logs.conf</code>	<code>etc/localtime</code>	<code>etc/snmpd.conf</code>
<code>etc/grfr.conf</code>	<code>etc/master.passwd</code>	<code>etc/spwd.db</code>
<code>etc/grifconfig.conf</code>	<code>etc/motd</code>	<code>etc/syslog.conf</code>
<code>etc/grinchd.conf</code>	<code>etc/named.boot</code>	

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.

grms command (non-privileged login)

GRF 400 and GRF 1600

When working at the VT-100 terminal, use the **grms** command to halt, reboot, or shut down the GRF from the UNIX prompt. **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. **grms** has a man page and is also described in the *GRF Reference Guide*. To perform an orderly reboot of the system, enter:

```
# grms -r
```

RMS node systems

When working at the RMS node, use the **grms** command to halt, reboot, or shut down the system from the UNIX prompt. **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 RMS node. **grms** has a man page and is also described in the *GRF Reference Guide*. To perform an orderly reboot of the system, enter:

```
# grms -r
```

shutdown command (root login)

GRF 400 and 1600

If you manage the GRF from a terminal other than the VT-100, the **grms** command is not available on that terminal. To cleanly stop and reboot the system from `root` login, use the UNIX **shutdown** command, enter:

```
# shutdown -r now
```

RMS node systems

If you manage the GRF or GR-II from a terminal other than the RMS node, the **grms** command is not available on that terminal. To cleanly stop and reboot the system from `root` login, use the UNIX **shutdown** command, enter:

```
# shutdown -r now
```

Simple connectivity tests

- 1 Ping each interface.
When **ping** is properly returned, the interface is up, configured, and has an active media connection.
- 2 Ping an interface external (on some other router or station) to the GRF or GR-II via each interface. This can be to a workstation, an attached host, or another router.

Test remotely from a workstation or host

- 1 Ping each GRF or GR-II interface.
- 2 Ping another device on a different subnet on the GRF or GR-II.
- 3 Test `telnet`, `ftp`, etc., through the GRF or GR-II.

ATM OC-3c Configuration

3

Chapter 3 provides information needed to configure the ATM OC-3c media card.

The GRF can be configured in point-to-point or point-to-multipoint ATM topologies with either switches or hosts. The OC-3c media card provides two independent physical ATM interfaces, each of which supports 110 logical interfaces.

Note: 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.

This chapter contains:

ATM OC-3c functions and features	3-2
Physical and logical ATM interfaces	3-8
Configuration files and profiles	3-12
Assign IP addresses - grifconfig.conf	3-13
Specify ATM card parameters – Card profile	3-14
Change run-time code (optional) – Load profile	3-17
Change dump default (optional) – Dump profile	3-19
Configuring PVCs	3-21
PVC example	3-22
ATM on-the-fly PVCs	3-24
Support for SVCs	3-25
SVC example	3-27
Traffic shaping	3-28
Traffic shape names	3-33
Setting output rates	3-34
SVC creation process	3-35
Configuring selective packet discard	3-37
maint commands for ATM OC-3c media cards	3-39

ATM OC-3c functions and features

Note: 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.

Large route table support

ATM OC-3c card software maintains route tables containing up to 150K entries, and hardware support for full table lookups.

PVC on-the-fly reconfiguration

On ATM OC-3c cards you can reconfigure 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.

Selective packet discard

Selective packet discard can be enabled on the ATM OC-3c card to ensure that 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 routing packets. Setting a congestion and discard threshold is described in the “Selective packet discard” section of this chapter.

Selective packet discard and ATM high/low priority outbound queues are two independent functions. Selective packet discard only affects discarding, not queuing. When packets come off the switch to a media card, they are discarded if there are no buffers available on the card. 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.

After packets are taken off the switch, selected for discard, and the survivors moved to buffers on the transmit side of the ATM card, packets are handled according to the QOS parameters that are defined for their VCs. All packets are now subject to the “high priority” vs. “low priority” handling that determines which data is put on the outgoing line first.

Precedence field

IP packets can have the precedence field in the IP packet header set three ways:

- by GateD on routing packets
- by filters configured to set this field on incoming data that matches any filter definition
- by an end application, such as special administrative programs on the control board

Most routing traffic has 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)

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.

Special buffer management during congestion

This feature provides fair allocation of buffers on the transmit side of the ATM OC-3c card during periods of congestion. During congestion, a single source can overdrive a slow rate queue such that traffic will back up behind the slow rate queue until all buffers are consumed. This means buffers are made available to the switch at the rate of the slowest oversubscribed rate queue. As a result, high-speed traffic is metered at the same rate as low-speed traffic, which is clearly undesirable.

Fairness is achieved by rate-limiting the amount of traffic that can be transmitted on any VC to the sustained rate of that VC during periods of congestion. Currently, the problems arise due to an input/output mismatch. Since the rate at which buffers are made available to the switch is the same as the rate at which they are transmitted by the SAR, any slowdown in SAR transmission directly reduces the rate at which free buffers are available to the switch.

Each VC has a known peak rate and sustainable rate. These values are used to select a rate queue for the VC. By using the same parameters for the rate-limiting scheme, it is possible to ensure that the input to the SAR equals the output of the rate queue, and that all packets not conforming to the rate queue packets are discarded. This eliminates the input/output mismatch.

If the transmit side becomes congested, rate limiting is enabled automatically. Each VC is rate-limited to the sustainable rate configured for that VC. While the transmit side is uncongested, rate limiting is automatically disabled.

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.

Edit the `/etc/gratm.conf` file and specify `proto=isis` in the logical interface's PVC statement. Use the `proto=isis` field when IS-IS is the only protocol to run, specify `proto=isis_ip` if IP and IS-IS both will run. The `protocol` field enables an interface to accept IS-IS packets.

Here is an example:

```
pvc ga030 0/40 proto=isis traffic_shape=high_speed_high_quality
```

Refer to the *Introduction to IS-IS* chapter for more information.

Broadcasting

The ATM OC-3c media card does not use standard broadcast IP group address.

Broadcast addresses are entered in the *Service section* of the `gratm.conf` file, and the card's transmit interface routes broadcast datagrams using the `type=bcast` Service in `gratm.conf`. Here is an example:

```
# Broadcast Service info
Service name=bcl type=bcast addr=204.221.157.121
    addr=
    addr=
#
```

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 for more information.

Signaling

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.

Each physical interface (port) can support UNI 3.0 or UNI 3.1. To use UNI signaling, set the appropriate port to `UNI3.0` or `UNI3.1` in the *Signaling section* of `gratm.conf`. To disable UNI signaling, set the port to `NONE`.

If signaling is on, the other devices must use the same type of signaling, either UNI3.0 or UNI3.1, because the two versions are not compatible.

When UNI signaling is operating on a physical interface, 110 logical interfaces can run virtual circuits in any combination of PVCs and SVCs.

Packet buffering

Buffering on the ATM media cards is done in terms of IP packets, one packet per buffer. One IP packet may consist of several ATM cells. Buffering is provided for 256 IP packets on the receive side and 256 IP packets on the transmit side. Each packet can contain up to 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). The transmit and receive sides can each output 49152 cells (256 buffers x 192 cells).

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 provide an explanation of how local pings operate on ATM cards. Ping times are affected by:

- amount of traffic going through the router generally
- low or high priority of the assigned rate queue
- number of VCs assigned to low priority rate queues in relation to the number 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 supercedes 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.

SDH and SONET modes

The physical layer can be set to either SONET or SDH mode, SONET is the default.

Mode is a parameter in the `gratm.conf` file and is specified in the “*Signaling*” line. The example below shows mode specified first as SDH and then as SONET:

```
gratm.conf
# Signaling parameters
```

```
Signaling card=5 connector=bottom protocol=UNI3.0 mode=SDH
```

or

```
Signaling card=5 connector=bottom protocol=UNI3.0 mode=SONET
```

Notice that mode and signaling are configured per physical interface (`connector=`), and not on a logical interface.

Inverse ARP

The ATM OC-3c media card supports ATM inverse ARP.

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
```

The GRF takes the ARP entry learned via ATM inverse ARP as opposed to the one in the `grarp.conf` file. If no ARP entry exists for a given PVC when **grarp** is run, the ARP entry given in the `grarp.conf` file is accepted.

Internal/external clock source

The SUNI component has a receive and a transmit clock. The receive clock is *always* at the SUNI's internal setting.

However, the transmit side clock setting can be toggled between the recovered receive clock (the SUNI's own clock) and the external oscillator (the clock of the transmitting node).

Using the **maint 22 port value** command, you can set the SUNI's receive clock to external oscillator.

```
GR 06> maint 22 6 1
```

To set the receive clock to the recovered receive clock, enter:

```
GR 06> maint 22 6 0
```

These **maint** settings are *temporary*, and revert back to recovered receive clock (0) at ATM card reboot and system reset.

MTU

The maximum transmit unit for an ATM OC-3c packet is 9180 bytes, it cannot be set to a higher value.

ATM adaption layer support

The ATM OC-3c media card supports only AAL-5.

The ATM Adaptation Layer (AAL) supports the different types of traffic that can cross over ATM. The AAL consists of two parts, 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 what type of encapsulation is inside an ATM cell.

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 that follows the LLC header that specifies distinct routed or bridged PDUs.

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 near the end of this chapter. Other tools useful for looking at the ATM OC-3c media card include:

- **netstat -in**
- **ifconfig -a**
- **grstat**

Physical and logical ATM interfaces

Figure 3-1 shows the organization of physical and logical ATM interfaces on the ATM OC-3c media card:

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	
(220 / card)		(1024 / card) (G0048)		

Figure 3-1. ATM physical and logical interfaces

Physical interfaces

The ATM OC-3c media card supports two physical interfaces. Each physical interface supports the assignment of 110 logical interfaces out of a range of 128.

Logical interfaces

Logical interfaces provide a simple way of mapping many IP addresses onto a single ATM port. A logical interface serves as the connection between ATM and IP. Each logical interface is assigned a unique IP address in `grifconfig.conf`.

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

VPI/VCI are assigned to logical interfaces in `gratm.conf`, and provide the bridge between ATM and IP.

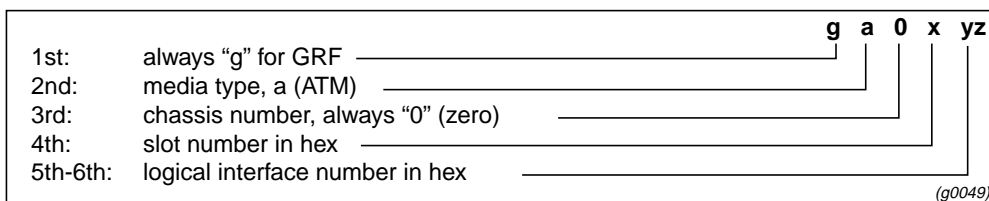


Figure 3-2. Interface name structure for an ATM logical interface

Interface name *ga0yz*

As shown in Figure 3-2, the generic form of an ATM interface name is: `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
- “z” ranges from 0 through ff to individually identify one of the 220 logical interfaces that can be configured per ATM card

Logical interfaces on connector 0 (upper) range from 0 to 7f.

On connector 1 (lower), they range from 80 to ff.

You can assign 110 out of the range of 128 logical interfaces per physical interface.

Virtual circuits

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, 0/2645). 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 *active* virtual circuits (VCs) as defined in the ATM Forum UNI 3.0 specification. VCs can be divided between the two physical interfaces in any manner required by the site, with 512 VCs active at any one time on each interface. Each virtual circuit has an associated IP address.

Virtual paths

A virtual path connects two end stations which may be separated by one or more network devices such as a router or switch. A path consists of one or more virtual circuits.

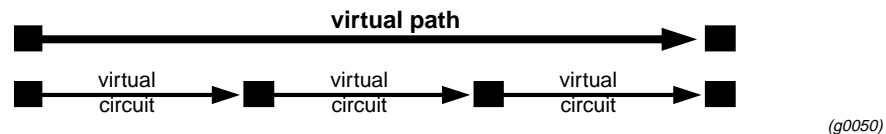


Figure 3-3. Components that form a virtual path

Virtual path identifiers (VPIs) 0 through 15 are available for configuration use. VPIs are assigned in the `/etc/gratm.conf` file.

VCIs

Virtual circuit identifiers (VCIs) name virtual circuits. VCIs are assigned in the `gratm.conf` file.

On VPI 0: VCI 0 through VCI 32767 can be used

On VPIs 1-15: VCI 0 through VCI 511 can be used

Note: Virtual circuits 0-31 on each VPI are reserved for use by UNI signaling.

VPI/VCI

In the `/etc/gratm.conf` file, **VPI/VCI** specifies the (numeric) Virtual Path Identifier and Virtual Circuit Identifier of the virtual circuit, separated by a slash (/). For example, 0/126.

VPI values = 0–15
VCI values = 0–511, 0–32767 depending upon the virtual path

Permanent virtual circuits

Permanent virtual circuits (PVCs) are created statically. PVCs are configured in `gratm.conf`.

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`.

The VPI/VCI must only be unique per physical ATM port, not per card. The following VPI/VCIs are supported:

VPI 0: VCI 0 . . 32767
VPIs 1-15: VCI 0 . . 511
VCI 0-31 on each VPI is reserved for UNI signaling.

On a given physical interface, 512 VCs can be active at any one time.

Note: VPI/VCI pairs must be unique per physical port.

It is not legal to have two or more circuits with the same VPI/VCI in the same logical interface.

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 `gratm.conf`. However, 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. Both sides of the ATM link must use the same version of the signaling protocol. If you do not wish to use SVCs, set the signaling for that interface to NONE.

UNI signaling uses ATM Format NSAP addresses, not IP addresses, and requires the use of an ARP server. 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 `gratm.conf`.

PVCs and SVCs can be used simultaneously on the same physical interface (port). PVCs and SVCs can also coexist on the same logical interface.

Verifying an ATM configuration

maint commands enable you to verify an ATM configuration. They are described at the end of this chapter and some examples are provided.

Operations include:

- verify VC configuration
- examine active VCs and traffic on them
- display ARP servers registered with
- display known ARP entries

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:

```
# greset <slot_number>
```

Configuration files and profiles

These are the steps to configure ATM cards:

1. Assign IP address to each logical interface

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

2. Specify ATM card parameters in the Card profile:

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard percentage in the `spd-tx-thresh` field
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

3. Configure PVCs and SVCs in `/etc/gratm.conf`

Configure PVCs and SVCs in the `/etc/gratm.conf` file.

4. 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 ATM card.

If you want to change the run-time code in one ATM card (per physical interface), make the change in the Card profile, in the `load` field.

5. 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 zero (0) which actually stores two dumps per day (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 (per interface), make the change in the Card profile, in the `dump` field.

Assign IP addresses - *grifconfig.conf*

Edit the `grifconfig.conf` file to assign an IP address to each logical ATM interface. You also can provide other information about the logical IP network to which that interface is physically attached, or specify a different MTU in the `arguments` field, for example.

When you configure a logical interface on a point-to-point media such as ATM, enter the destination IP address in the `broad_dest` address field. An entry in the `broad_dest` address field for an ATM interface 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.

The optional `arguments` field is currently used to specify MTU values on a logical interface basis. 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.

Each logical interface is identified in `grifconfig.conf` as to its:

- interface name, `ga0yz` (always lower case)
- Internet address
- netmask
- broadcast/destination address
- `arguments` field (optional)

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

name	address	netmask	broad_dest	arguments
ga030	xxx.xxx.xxx.xxx	255.255.255.0	-	mtu 9100
ga0d82	yyy.yyy.yyy.yyy	255.255.255.0	zzz.zzz.zzz.zzz	

This example sets an IP address for logical interface 0 on the ATM card in slot 3 (upper physical interface), and specifies a lower than default MTU value. A dash is used as a placeholder for the broadcast address.

The second entry sets an IP address for logical interface 130 on the ATM card in slot 13 (lower physical interface), and specifies a destination address.

Specify ATM card parameters – Card profile

Set specific ATM card configuration parameters at the Card profile. The fields to set are:

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard percentage in the `spd-tx-thresh` field
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

Fields you may want to set are in bold, default values are shown here.

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 }
```

Specify selective packet discard percentage

Here is where you enter the congestion threshold percentage:

```
super> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0
```

Change the threshold to 6 with this series of commands:

```
super> set spd-tx-thresh = 6
super> write
CARD8/read
```

Specify ICMP throttling

You can specify ICMP throttling changes for this ATM card in these settings. Refer to Chapter 1 for an explanation of each field or do a `set <field-name>?` for a brief description. Default values are shown:

```
super> list ic
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

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/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.

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 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
```

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 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)

ATM OC-3c Configuration

Specify ATM card parameters – Card profile

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
```

Change run-time code (optional) – 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** ATM cards.

Here is the path, default settings are shown:

```
super> read load
LOAD read

super> list
hippi = { "" N/A on 0 1 < { 1 /usr/libexec/portcards/xlload.run N/A } {
2 /usr+
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+
```

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> list atm-oc3-v2
type = atm-oc3-v2
```

ATM OC-3c Configuration

Change run-time code (optional) – Load profile

```
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 = < >

super> set enable-boot-seq = on
super> write
LOAD written
```


Change dump default (optional) – Dump profile

Global 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 accommodates the default setting of zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, default settings are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0} {fddi 20 /var/portcards/grdump 2} {rmb 20 /+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi core +
config-spontaneous = off
keep-count = 0
```

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 = { hippo 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.

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 mem+}
```

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
```

Configuring PVCs

Overview

To configure PVCs, use a UNIX editor to edit the `grarp.conf` and `gratm.conf` files. Open the UNIX shell:

```
super> sh
```

To configure a logical interface that supports PVCs, make entries in these configuration files:

grifconfig.conf

- assign IP address to the logical interface

grarp.conf

- supply IP-to-physical address mapping information for ARP service
Put an entry into `grarp.conf` ONLY if the remote destination does NOT support inverse ATM ARP. (The GRF supports inverse InATMARP.)

gratm.conf

- *Traffic Shaping section*
Set traffic shaping name and quality of service parameters, use any string, set a name for each type of service that will be assigned
- *Signaling section*
Set **protocol=NONE** on the signaling entry for the appropriate card and connector combination
- *Interface section*
Define traffic shaping profile for the logical interface to which the media card's PVCs are assigned
- *PVC section*
Specify characteristics for each PVC, including:
 - assigned logical interface name
 - VPI/VCI
 - protocol supported
 - AAL used
 - assigned traffic shaping profile (only if it would be different from the profile given above to the logical interface in the *Interface section*)

Templates of these configuration files are in the *GRF Reference Guide*.

Traffic shaping is discussed in detail later in this chapter.

PVC example

This example configures a PVC with the following attributes:

- resident on upper physical interface, card in slot 3
- on logical interface 153 (hex=99)
- connects to an endpoint that does not support inverse ARP

The ATM card supports inverse ATM ARP (InATMARP). This protocol is used to determine the IP address at the other end of a PVC.

- VPI/VCI, in this case 0/32
- runs IP protocol, AAL-5 (default, no matter what is set)
- the local IP address is 192.0.130.1
- the remote IP address is 192.0.130.111

grifconfig.conf

```
# name      address      netmask      broad_dest      arguments
ga0399      192.0.130.1    255.255.255.0
```

Here is the ISO address entry and format if the PVC will run IS-IS:

```
# <interface-name> <iso-address> <iso-area> - iso
ga030 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

grarp.conf

```
# [ifname]  hostname  phys_addr      [temp]  [pub]  [trail]
#
ga0399      192.0.130.111    0/32
```

gratm.conf

```
#
# Traffic shaping parameters
#
Traffic_Shape name=low_speed_high_quality peak=15000 qos=high
Traffic_Shape name=high_speed_low_quality peak=15000 qos=low

# Interfaces
#
Interface ga0399  traffic_shape=low_speed_high_quality

# PVC's
#
# Lines beginning with the keyword "PVC" define
# Permanent virtual circuits.
#
# The format of a PVC definition is:
#
#PVCifname/VPI/VCIproto=ip|raw|vc|ipnllc|isis|llc|ether|fdi|vc_atmp\
#      [input_aal=3|5|NONE] [traffic_shape=shape] \
```

```
#      [dest_if=logical_if [dest_vc=VPI/VCI]]
#
PVC ga0399  0/32 proto=ip traffic_shape=high_speed_low_quality
```

The *Traffic Shaping section* includes many different `Traffic_Shape name=` entries as shown above, one for each type of service that could be assigned.

The `proto=` field supports the following options:

- | | |
|----------------------|-----------------------------------------------------------------|
| <code>ip</code> | - for Internet Protocol (with LLC/SNAP headers) |
| <code>raw</code> | - for raw adaptation layer (AAL-5) packets |
| <code>vc</code> | - for VC-based multiplexing of IP as specified in RFC 1483 |
| <code>ipnllc</code> | - for any LLC-encapsulated protocol supported by the GRF |
| <code>isis</code> | - for the IS-IS protocol |
| <code>ether</code> | - for bridged Ethernet packets |
| <code>fddi</code> | - for bridged FDDI packets |
| <code>vc_atmp</code> | - for ATMP home network connections using VC-based multiplexing |

ATM on-the-fly PVCs

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=low_speed_high_quality \ peak=15000 qos=high
# Interfaces
Interface ga03c8  traffic_shape=low_speed_high_quality
# PVC's
PVC ga03c8  0/32  proto=ip  traffic_shape=low_speed_high_quality
```
- 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 hshq qos=low for bottom connector card 0.
gratm: Accepted traffic shape hshq qos=low for bottom connector card 1.
gratm: Begin on-the-fly PVC configuration for card 0x3
/usr/sbin/grinch -p 1 2.12.2.4.17.3.1=1
```
- 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
Dec  2 18:22:57 jan kernel: ga03c8:  GRF ATM, GRIT address 0:1:0xf0
gratm: Sent 12 grinchs for card 0x3
# Dec  2 18:22:57 jan kernel: ga03c8:  GRF ATM, GRIT address 0:1:0xf0
```

Then use the `ifconfig -a` command to see that the newly-added PVC is included.

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: This feature applies only to PVCs on ATM OC-3c cards. It does not apply to ARP servers, SVCs, 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 ARP server and signaling settings to apply.

Support for SVCs

An SVC is the name given to the connection that is dynamically set up between IP endpoints. The IP address is configured on a logical ATM interface. The GRF is an IP router and so uses the IP address to connect the endpoints. ATM devices connect using NSAP addresses, FDDI devices connect using a MAC address.

Switched virtual circuits (SVCs) and permanent virtual circuits (PVCs) are configured differently.

This is because an SVC only exists as a “dynamic” path across a logical interface. An SVC does not become a path until a device requests a path. This request specifies a set of parameters that the path must possess. Parameters include a destination IP address, service type, signaling type, and traffic shaping parameters.

A logical interface with matching parameters is found, and signaling software sets up an SVC on the interface. The transmission of packets begins immediately once the connection is set up. After the transmission completes, the SVC “identification” is switched back to the list of available SVCs.

An example describing the basic steps in the creation of an SVC is found later in this section.

ARP servers

ARP servers maintain tables of IP addresses mapped to NSAP or other device-specific addresses. The GRF queries ARP servers when it has IP addresses for which it needs associated NSAP or MAC addresses. ATM switches maintain ARP tables and act as ARP servers. When UNI support is set on (in `gratm.conf`), the GRF goes through ILMI registration with the connected switch to build the NSAP address. The ATM ARP Server **maint** command displays NSAP addresses.

Configuration overview

To configure SVCs, use an UNIX editor to edit the `grifconfig.conf` and `gratm.conf` files. Open the UNIX shell:

```
super> sh
```

To configure a logical interface that supports SVCs, make entries in these configuration files:

grifconfig.conf

- assign IP address to the logical interface

grarp.conf

- no entries needed

gratm.conf

- *Service section*)
Select ARP or broadcast service

- *Traffic Shaping section*
Set traffic shaping and quality of service parameters
An SVC assumes the traffic shaping parameters of the logical interface to which it is assigned.
- *Signaling section*
Specify signaling protocol for the two media card physical interfaces, either **UNI3.0**, **UNI3.1**, or **none**.

Specify either SONET or SDH mode, SONET is the default if nothing is specified.
- *Interface section*
Identify the logical interfaces (using the `ga0xyz` interface name) to which the media card's SVCs are assigned

The *GRF Reference Guide* contains templates for the configuration files listed above.

Traffic shaping is discussed in detail later in this chapter.

SVC example

This example configures a logical interface with the following attributes:

- resident on lower physical interface, card in slot 3
- on logical interface 200 (hex=c8)
- runs UNI3.0 protocol
- names one or more ARP servers and their specified NSAP addresses

The GRF can query an ATM ARP server (arp0) at NSAP address
47000580ffe1000000f21513eb0020481513eb00

for the IP address of a requested destination device (endpoint).

The GRF can also query another ARP server (arpX) at NSAP address
47000580ffe1000000f21513eb0020481513eb00

for the IP address of a requested destination device (endpoint).

Here are the entries in the `grifconfig.conf` and `gratm.conf` configuration files:

grifconfig.conf

```
# name      address      netmask      broad_dest      arguments
ga03c8  192.0.130.1    255.255.255.0
#
```

gratm.conf

```
# ARP Service info
#
Service name=arp0 type=arp \
      addr=47000580ffe1000000f21513eb0020481513eb00
Service name=arpX type=arp \
      addr=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# Traffic shaping parameters
Traffic_Shape name=medium_speed_low_quality peak=75000 qos=low

# Signaling parameters
Signaling card=5 connector=bottom protocol=UNI3.0 mode=SONET

# Interfaces
Interface ga03c8 service=arp0 traffic_shape=medium_speed_low_quality
```

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.

The GRF ATM card expects to forward cells as fast as it can. However, the next node in the virtual circuit connection may not be able to handle this rate of traffic. Traffic shaping provides a measure of control over the rate at which cells are transmitted.

Parameters

Traffic shaping on the ATM card uses three parameters described in UNI 3.0/3.1 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)
- sustainable cell rate, in kilobits/second (SCR)
- maximum burst size, in cells (MBS)

Quality of Service (qos) is also set in `/etc/gratm.conf`:

- priority, in Quality of Service (qos), either high or low

Note: Remember to specify PCR and SCR in kilobits/second, not in bits/second.

For example, use 155000, not 155. If you specify 155, the ATM card times out while trying to move 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 155000 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 them to go out. If the mismatch in speeds is large, packets on the faster incoming network eventually will be lost, and retransmission will be required.

Sustainable cell rate

The sustainable cell rate (SCR) is generally the effective transfer rate. The sustainable 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 (SPR).

Software adjusts each specified sustainable 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.

On the ATM OC-3c card, maximum burst size operates only as multiples of 32 cells. The smallest burst size allowed is 32 53-byte cells (13568 bits), the largest burst allowed is 2048 cells (868352 bits). Software rounds the requested burst size downward to the next such multiple, or 32, whichever is greater.

Maximum burst size has no meaning unless the specified sustainable rate (SCR) is less than the peak rate (PCR). As long as the VC has data to send, it sends its cells at the sustainable 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 sustainable rate. Within a certain latitude determined by the MBS, this allows a VC to transmit at the sustainable 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 sustainable rate transmit credits. This means that the virtual circuit (VC) has to have been transmitting below the sustainable rate in order for any burst rate credits to accumulate. For bursting to occur, the VC must average less than the sustainable rate. Unused sustainable rate transmit credits can accumulate due to recent idle and under-subscribed periods.

- In a recent idle period, the circuit usually transmits at the sustainable rate but has been idle for the last N cell times.
- In an under-subscribed period, the circuit usually transmits below the sustainable rate.

Burst credits are accumulated at the sustainable 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

The ATM card allows eight rate queues per physical ATM interface. Four queues are high-priority, four are low-priority.

Quality of Service (QOS) has two options, high and low. You can have four high QOS and four low. If no QOS is defined, the default is high. High QOS queues get serviced before low.

There are eight rate queues, 0–3 are high QOS, 4–7 are low QOS. The rate queues equate to priority. The first four queues defined for high priority and the first four queues defined for low priority become the available rate queues. You can then assign maximum, sustained, and burst rates to each of the rate queues. Only four different peak cell rates can be assigned per QOS.

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.

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. 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.

Priority is an attribute of the logical interface. SVCs have the priority level of their assigned logical interface.

A high-priority (for access) queue means a Quality of Service (qos) equal to high.
A low-priority (for access) queue equates to a Quality of Service (qos) = low.
qos is specified in `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 reported experiencing slow response time on this ATM OC-3c card and on remote interfaces. The low QOS rate queues showed high packet loss, the high QOS rate queues showed minimal packet loss.

The following problem is likely occurring: based on the number of switch receive overflows, either there are multiple cards sending lots of traffic to the ATM card, not unusual, 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 7 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.

At the very least, I would change the rate queue allocation to be:

```
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. Oversubscription also applies to high priority traffic, it is just worse with low priority because those packets must wait for all high priority packets to leave the buffers before being served.

Another technique is to use the sustainable rate per VC to meter the output.

In `/etc/gratm.conf`, let every VC have a peak=155000:

```
Traffic_Shapename=Bulldozerpeak=155000sustain=155000burst=2048qos=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_Shapename=Bulldozerpeak=155000sustain=155000burst=2048qos=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 sustainable 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 sustainable 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??? sustainable rate, because the SAR meters that on a per VC basis.

Traffic shape names

Peak cell rate, sustainable 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:

```
Traffic_Shape name=high_speed_high_quality \  
                peak=15500 sustain=15500 burst=2048 qos=high
```

(In `gratm.conf`, a backslash (\) is used to divide a single long line of characters.)

The name given above specifies the best possible service and access to bandwidth resources. The following name specifies the minimum service possible:

```
Traffic_Shape name=lowest_speed_lowest_quality \  
                peak=15500 burst=32 qos=low
```

Sustainable 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. The eight peak rates are combined with different sustainable rates and burst sizes to create more than eight `Traffic_Shape` names.

Peak rate is the only required parameter in a `Traffic_Shape` name. If you do not specify sustainable rate and burst size, they default to match the peak rate. Another optional parameter is Quality of Service (`qos`). Quality of Service defaults to high priority.

The maximum peak rate is 155000 kilobits/second, the maximum sustainable rate is also 155000 kilobits/second, and the largest burst size that can be specified is 2048 cells.

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.

Setting output rates

Sending at a controlled rate

To ensure that the transmission of cells does not exceed a specific rate, create a traffic shape specifying that peak rate in `gratm.conf`.

When the optional sustainable rate and maximum burst size are not specified, the ATM card automatically sets sustainable 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 sustainable cell rate (SCR), a maximum burst size (MBS), and a peak cell rate (PCR).

A *sustainable* rate is the upper bound of an average or *sustained* rate.

If SCR and MBS are specified, cells issue at the sustainable rate. The sustainable 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 sustainable rate.

Due to the time-slotted nature of ATM, the sustainable 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.

Changing a rate queue

Values in a rate queue cannot be changed on the fly. Changes must be made in the `gratm.conf` file and the ATM media card rebooted.

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 222.222.223.0 222.222.222.254`
- the following UNIX command to make an entry in the device's route table had previously executed on Device B:
`route add 222.222.222.0 222.222.223.254`

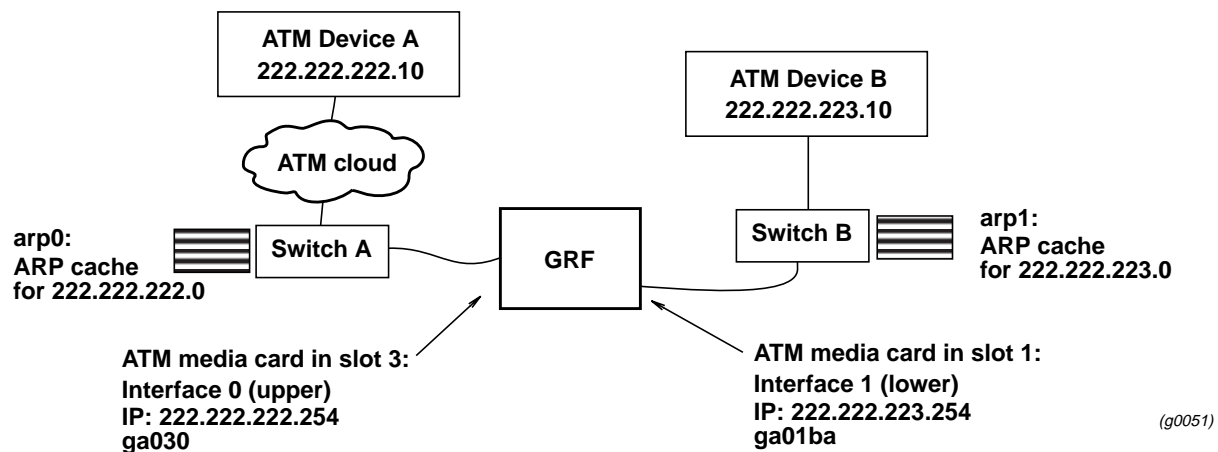


Figure 3-4. GRF role in ATM-ATM connection

Process

Device A:

- looks up the next hop entry (from route add) 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, creates two links (switched virtual circuits) for Device A, one to itself, one to the GRF

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 IP address (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, creates two links (switched virtual circuits) for the GRF, one to the switch itself, one to Device B

Once the connection is established, packets from the GRF flow through Switch B to Device B.

These are the entries already in `grifconfig.conf` and `gratm.conf` that would support the creation of SVCs from the ATM media card in slot 1:

grifconfig.conf

```
# name      address          netmask          broad_dest      arguments
ga01ba     222.222.223.254  255.255.255.0   222.222.223.10
#
```

gratm.conf

```
# ARP Service info
Service name=arp0 type=arp \
            addr=47000580ffe1000000f21513eb0020481513eb00
#
# Traffic shaping parameters
Traffic_Shape name=medium_speed_low_quality peak=75000 qos=low
#
# Signaling parameters
Signaling card=5 connector=bottom protocol=UNI3.0 mode=SDH
#
# Interfaces
Interface ga01ba service=arp0 traffic_shape=medium_speed_low_quality
#
```

Configuring selective packet discard

Selective packet discard can be enabled on the ATM OC-3c 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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 ATM OC-3c 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.

Example

Given an ATM OC-3c card installed in slot 2, this example shows where to set the threshold in the CLI:

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

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/2 written
super> grreset 2
```

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
```

maint commands for ATM OC-3c media cards

maint commands display a range of information about the 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.

Preparing to use maint

First, switch to the **maint** prompt. Enter:

```
# grmb
```

The new prompt appears:

```
GR 66>
```

Then change the prompt port to the ATM media card you are working with. For example, if you are working with a card in slot 3, enter:

```
GR 66> port 3
```

This message is returned along with the changed prompt:

```
Current port card is 03  
GR 03>
```

To leave the **maint** prompt, enter **quit**.

Transmit / receive side maint commands

To see the list of **maint** commands for the receive side, enter: `maint 1`, to see the list of **maint** commands for the transmit side, enter: `maint 101`

Receive side list

```
GR 11> maint 1  
[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 Combust 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 ]  
[RX] 16: Setup PVC [port vpi vci peak sust brst qos pro  
      (1=LLC,2=IP,5=uni,6=raw)]  
[RX] 17: Teardown PVC [ port vpi vci ]  
[RX] 20: Display SUNI Statistics [ port ]  
[RX] 21: Select SUNI Framing Mode [ port SONET=0 SDH=1 ]
```

ATM OC-3c Configuration

maint commands for ATM OC-3c media cards

```
[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 FRTLW route table NOT AVAILABLE
[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] Type maint 101 to see the list of TX side maint commands
```

Transmit side list

```
GR 11> maint 101
[TX] 101:   Display this screen of options for the TX 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] 162:   Display Switch Descriptor Ring [num_entries]
[TX] 70:    Display ATMP Home Network table
[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.
```

Sample ATM maint displays

Display active interfaces

The **maint 3** command gives you useful options for looking at a variety of active interfaces.

```
GR 02> maint 3
GR 02> [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
```

Note: The broadcast options are not available.

Check IP addresses

You can list the IP addresses configured to the logical interfaces on the specified port:

```
GR 02> maint 3 1 0
[RX]
[RX] Port 0: LINK UP
[RX] Port 1: LINK DOWN
[RX]
[RX] IF          IP          STATE | IF          IP          STATE
-----|-----
[RX] 00  222.222.0.1      UP      | 01  222.222.1.1      UP
[RX] 02  222.222.2.1      UP      | 03  222.222.3.1      UP
[RX] 04  222.222.4.1      UP      | 05  222.222.5.1      UP
[RX] 06  222.222.6.1      UP      | 07  222.222.7.1      UP
[RX] 08  222.222.8.1      UP      | 09  222.222.9.1      UP
[RX] 0a  222.222.10.1     UP      | 0b  222.222.11.1     UP
[RX] 0c  222.222.12.1     UP      | 0d  222.222.13.1     UP
[RX] 0e  222.222.14.1     UP      | 0f  222.222.15.1     UP
```

Check virtual circuits

You can list the VPI/VCIs configured to each logical interface on either ATM port.

```
GR 02> 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] 00   0/0     IPLLCC    01   0/1     IPLLCC    02   0/2     IPLLCC
[RX] 03   0/3     IPLLCC    04   0/4     IPLLCC    05   0/5     IPLLCC
[RX] 06   0/6     IPLLCC    07   0/7     IPLLCC    08   0/8     IPLLCC
[RX] 09   0/9     IPLLCC    0a  0/10    IPLLCC    0b  0/11    IPLLCC
[RX] 0c  0/12    IPLLCC    0d  0/13    IPLLCC    0e  0/14    IPLLCC
[RX] 0f  0/15    IPLLCC
```

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: 0000000000000090848
[RX] RX Bytes: 00000000000006498864
[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 SARA STATISTICS
-----
[TX] TX Packets: 00000000000068049018
[TX] TX Bytes: 00000000075260262768
[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 successfully reassembled.

PCQ overflow (Packet Complete Queue)

A packet was successfully 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 segmentation.

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
GR 02> [RX]
[RX]
[RX]                               Switch Statistics
[RX] input:
[RX]          Bytes          Packets          Errors
[RX] -----
[RX] 00000000075687116396 00000000000068054080 0000000053
[RX]
[RX] output:
[RX]          Bytes          Packets          Errors          Overruns
[RX] -----
[RX] 00000000000011990440 00000000000000094514 0000000000 0000000000
[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
```

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

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] f1   15/32    pvc    5    IP-LLC          pt-pt
[RX] f0   15/511   pvc    5    IPNULL

```

```
GR 02> maint 13 0
[RX]
[RX] IF   VPI/VCI  TYPE  AAL  ENCAPSULATION
-----
[RX] 00    1/510   pvc    5    IP-LLC          pt-pt

```

VPI/VCI traffic statistics

You can return VPI/VCI traffic statistics on a per port, per side basis:

```
GR 02> maint 14 0
[RX]
[RX] RECEIVE:
[RX] IF   VPI/VCI  PACKETS      BYTES      IP DISCARD  UNSUPP LLC
-----
[RX] f1   15/32      0000008416   0000807936   0000000000   0000000000
[RX] f0   15/511     0000088376   0005976816   0000000000   0000000000
0000000e 00000001
[TX]
[TX] TRANSMIT:
-----
[TX] f1     15/32              67964701      2133541408
[TX] f0     15/511             90459          121076976

```

RX/TX Packets

A count of the successfully 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 VC which use LLC/SNAP encapsulation, indicates the count of packets of an LLC/SNAP type not supported by the GRF.

Select SUNI framing mode

The **maint 21 1 0** command sets port 1 to SONET mode. This setting is temporary, make a permanent selection in the *Signaling* section of the `/etc/gratm.conf` file.

```
GR 02> maint 21
GR 02> maint 21 1 0
GR 02> [RX]
[RX] Port 1: SONET framing mode
```

Display ARP servers

```
GR 02> maint 109
GR 02>
ARP-S                      NSAPA                      STATE
-----
00:  0X47.0005.80ffe1000000f21513eb.0020481513eb.00  0/101
      -----
      -----
      -----
GR 02>
```

Display rate queues

This command displays the rates queues per port:

```
GR 02> maint 125 0
[TX]
[TX] RQ    State      Rate(Kbs)      VPCIs
-----
[TX] 00  ENABLE      155000         15/32  15/511
[TX] 01  DISABLE
[TX] 02  DISABLE
[TX] 03  DISABLE
[TX] 04  DISABLE
[TX] 05  DISABLE
[TX] 06  DISABLE
[TX] 07  DISABLE
```

Display ATMP home network table

This command displays the ATMP home networks assigned to this ATM media card.

- 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.
- `s0` = ATM VPI, `s1` ATM VCI.
- State can be either HomeAgent (the Ethernet card) or CirHomAg, the HSSI or ATM card owning the link to the home network.
- The home agent address is also provided.

```
GR 0> maint 70
GR 0> [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
[RX] -----      -
[RX]      0      01:01:0000:32767  HomeAgent 205.1.1.2
[RX]
[RX] Entries: 1
```

Display ATMP tunnel information

The **maint 73 tunnel_number** command shows tunnel information.

Obtain the tunnel number using the **maint 70** command shown above. The tunnel number is the entry under FRT-Index, in this case, 1.

```
GR 02> maint 73 1
GR 02> [RX]
[RX] Mobile node tree list
[RX] Mobile Node/Mask Flags Foreign Agent Tunnel Id Slot:Port:s0:s1
[RX] 204.101.10.6/32 0 => 205.1.1.1 0x00000a06 1:1:0000:32767
```

This **maint 73 1** command shows a tunnel for the mobile node using address 204.101.10.6, 32 netmask bits, connecting to a foreign agent at address 205.1.1.1. The tunnel ID is 0x00000a06. The ATMP gateway circuit to the home network is on slot 2, port 0, VCI 32767.

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 VPI/VCI number of the dedicated ATM link to the home network

Display UME

Physical interface is either 0 (upper) or 1 (lower).

Enter: **maint 136 <physical interface>**

```
GR 03> maint 136 0
GR 03>
[SM] [UME] Main version: 2      Main revision: 2
          Branch version: 0     Branch revision: 0
          Part number: 2000048

SAP: 0      VPI: 0      VCI: 16
Upper Interface: Connected  UME protocol: Registering
                  AAL interface: Connected
Network prefix: 0X47.0005.80ffe1000000f21513eb.000000000000.00
Rx PDUs: 322   Rx PDUs Drop: 0   TxPDUs: 322
```

Use grrt to display the route table

Use the **grrt -S -p<slot>** command to display the current contents of the ATM OC-3c card's route table.

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

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
10.20.2.0	255.255.255.0	10	10.205.1.150	ga00f0	FWD
10.205.1.0	255.255.255.0	9	0.0.0.0	ga00f0	FWD
10.205.1.133	255.255.255.255	8	0.0.0.0	ga00f0	LOCAL
10.205.1.255	255.255.255.255	7	0.0.0.0	ga00f0	BCAST
10.205.3.0	255.255.255.0	13	0.0.0.0	ga00f1	FWD
10.205.3.133	255.255.255.255	12	0.0.0.0	ga00f1	LOCAL
10.205.3.255	255.255.255.255	11	0.0.0.0	ga00f1	BCAST
192.0.0.0	255.0.0.0	5	0.0.0.0	inx 0	DROP
192.0.0.1	255.255.255.255	5	0.0.0.0	inx 0	DROP
192.168.11.0	255.255.255.0	6	206.146.160.1	inx 0	RMS
192.168.10.0	255.255.255.0	24	10.205.3.150	ga00f1	FWD
192.168.10.0	255.255.255.0	24	10.205.3.150	ga00f1	FWD
192.168.1.2	255.255.255.255	23	205.1.1.2	inx 0	ATMP
192.168.160.0	255.255.255.0	3	0.0.0.0	inx 0	RMS
192.0.0.0	240.0.0.0	1	0.0.0.0	inx 0	DROP
192.0.0.0	255.0.0.0	3	0.0.0.0	inx 0	RMS
192.0.0.0	255.255.255.255	3	0.0.0.0	inx 0	RMS
255.255.255.255	255.255.255.255	2	0.0.0.0	inx 0	BCAST

Using grstat

```
# grstat -w 70 ip ga00f1
```

```
ga00f1
```

```
ipstat
```

count	description
8416	total packets received
8416	packets forwarded to the RMS

```
ipdrop
```

count	last source addr	last dest addr	reason
-------	---------------------	-------------------	--------

FDDI Configuration

Chapter 4 provides information needed to set FDDI media card configurations.

This chapter contains these topics:

FDDI functions	4-2
How FDDI interfaces are named	4-4
Configuration files and profiles	4-6
Set up FDDI media card – Card profile	4-7
Set up FDDI media card – Load profile	4-11
Set up FDDI media card – Dump profile	4-13
Single attach (SAS)	4-16
Dual attach (DAS)	4-16
Configuring SAS vs DAS	4-17
Installing FDDI connector keys	4-18
Optical bypass switch interface	4-19
Support for dual homing	4-21
Selective packet discard	4-21
Monitoring FDDI media cards	4-23

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.

FDDI functions

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.

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. Setting a congestion and discard threshold is described in the “Selective packet discard” section later in this chapter.

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
```


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.

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 4-1 indicates files where various numbers are used to configure the interfaces on a FDDI media card:

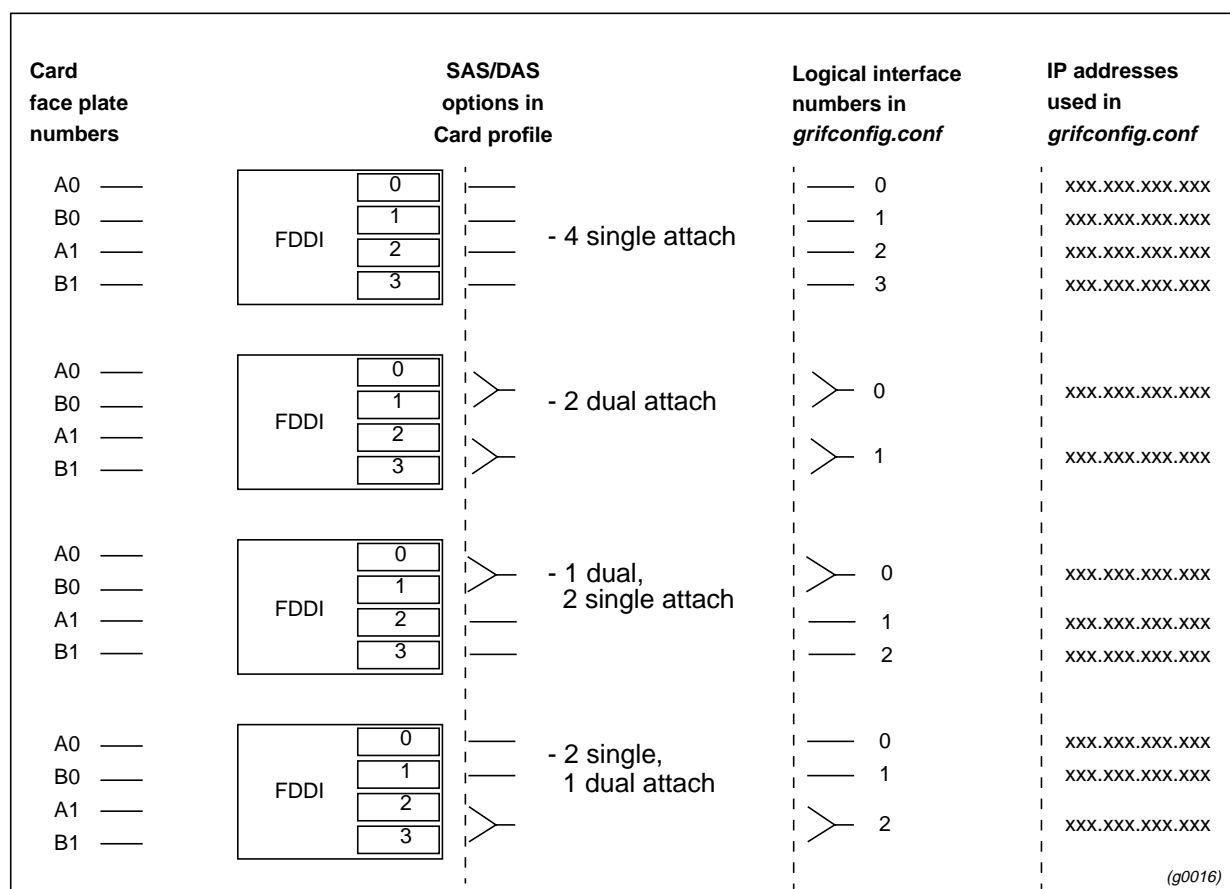


Figure 4-1. Assigning numbers to FDDI interfaces

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 4-2:

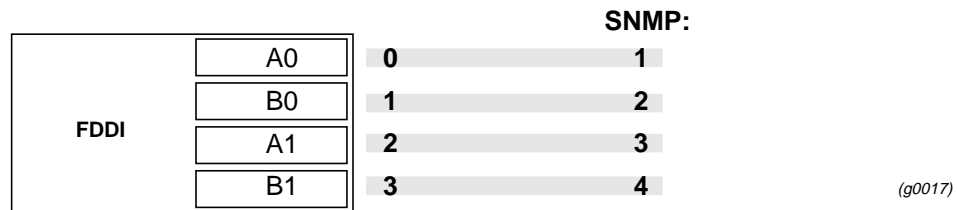


Figure 4-2. Physical interface numbering on FDDI media card

The diagram shows that the physical interface numbering is 0-based, 0–3, and that SNMP numbering is 1-based, 1–4.

GRF interface name gf0yz

The GRF interface name has five components that describe an individual FDDI interface in terms of its place in a GRF “world.”

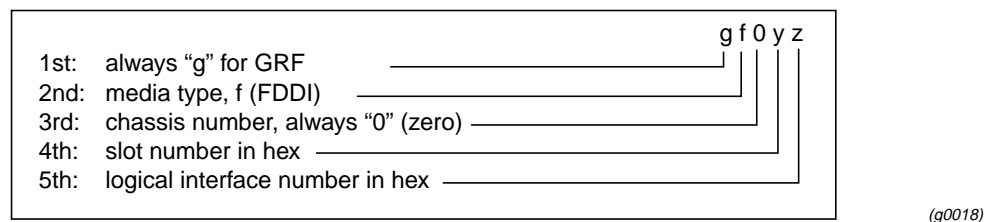


Figure 4-3. GRF interface name for FDDI interfaces

The interface name and IP address is specified in the `/etc/grifconfig.conf` file.

Here is the format of the `/etc/grifcong.conf` file:

```

name      address      netmask      broad_dest      arguments
gf031 xxx.xxx.xxx.xxx 255.255.255.0      -              mtu 4200

```

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

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.

Configuration files and profiles

These are the steps to configure FDDI cards:

1. *Identify each logical interface*

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

2. *Specify FDDI card parameters in the Card profile:*

Media card type, `fddi-v2`, is automatically read into the `media-type` field.

- OPTIONAL: specify ICMP throttling settings
- specify SAS and DAS settings as single or dual
- manually enable optical bypass on or off
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

3. *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 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. *Dump profile*

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 zero (0) which actually stores two dumps per day (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 (per physical interface), make the change in the Card profile, in the `dump / config` field.

Installing configurations or changes

In the command-line interface, use **set** and **write** commands to install configuration parameters.

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>
```

Set up FDDI media card – Card profile

Set specific FDDI card configuration parameters at the Card profile. The fields to set are:

- OPTIONAL: specify ICMP throttling settings
- specify SAS and DAS settings as single or dual
- manually enable optical bypass on or off
- OPTIONAL: specify selective packet discard percentage
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

Media card type, `fddi-v2`, is automatically read into the read-only `media-type` field. Other values shown are defaults. At the top level, you see ICMP throttling fields:

```
super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
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 inter-
nal-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 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> list ic
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

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/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. Specify SAS and DAS

In the `ports` section you specify SAS/DAS settings for each FDDI interface, 0 through 3:

```
super> cd ..      (if you listed icmp-throttling)

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

Here is a shortcut you also could use to get to interface 1 if you did not list ports first:

```
super> list ports 1
```

In port 1 you see the `fddi` field:

```
super> list 1
port_num = 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
disabled di+
```

Go into the `fddi` field to set SAS/DAS in the `single-dual` field. The value is preset to `single`. The following sets FDDI interface 1 to DAS:

```
super> list fddi
single-dual = single
optical-bypass = off

super> set single-dual = dual
super> write
CARD/8 written
```

Tip: A quick way to set interface 1 in slot 8 as DAS without moving “down” into the profile:

```
super> read card 8
CARD/8 read
super> set port 1 fddi single-dual = dual
super> write
CARD/8 written
```

3. Enable optical bypass

In the `ports` section you can specify an optical settings for each FDDI interface, 0 through 3:

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

Here is a shortcut you also could use to get to interface 1 if you did not list ports first:

```
super> list ports 1
```

In port 1 you see the `fddi` field:

```
super> list 1
port_num = 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
disabled di+
```

Go into the `fddi` field to set the `optical-bypass` field. The value is preset to `off`. The following sets the optical bypass on interface 1 to on:

```
super> list fddi
single-dual = single
optical-bypass = off

super> set optical-bypass = on
super> write
```

```
CARD/8 written
```

4. Specify selective packet discard %

This example shows how to set the SPD to 7%, refer to the “Selective packet discard” section in this chapter for more information.

```
super> read card 8
CARD/8 read

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/8 written
super>
```

5. 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 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_fddiq
super> write
CARD/8 written
```

6. 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 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
```

Set up FDDI media card – 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** FDDI 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/xlload.run
N/A } { 2 /usr+
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/port-
cards/hssi_tx.run +
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/port-
cards/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/port-
cards/fddiq-1.r+
atm-ocl2-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/port-
cards/sonet_t+
```

FDDI Configuration

Set up FDDI media card – Load profile

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 = < >
```

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> 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 = < >

super> set enable-boot-seq = on
super> write
LOAD written
```

Set up FDDI media card – 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 zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0} {fddi 20 /var/portcards/grdump 2} {rmb
20 /v+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi
core mem+
config-spontaneous = off
keep-count = 0
```

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 = { hippo 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

FDDI Configuration

Set up FDDI media card – Dump profile

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 mem-
ory" 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 a portion 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

super> cd ..

super> list s 2
index = 2
description = "fddi/Q IPC memory"
start = 8388608
length = 32800

super> cd ..

super> list s 3
index = 3
description = "fddi/Q shared descriptor memory"
start = 16777216
length = 131072
```

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:

```
# greset <slot_number>
```

This completes the procedure to configure FDDI cards. The rest of the chapter discusses FDDI connection options and describes the sets of **maint** monitoring commands.

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 4-4, 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.

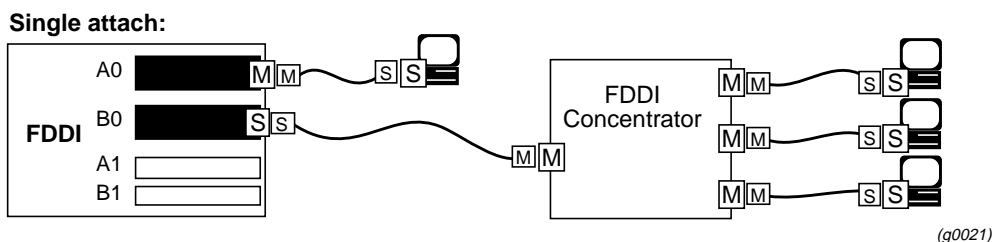


Figure 4-4. Master/slave connector keys for single-attach interfaces

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 4-5, 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.

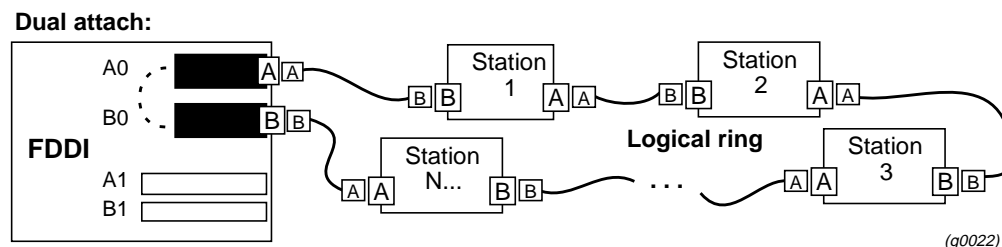


Figure 4-5. A/B connector keys for dual-attach interfaces

Configuring SAS vs DAS

Figure 4-6 shows the SAS and DAS port options. They are set in the Card profile to configure an interface as single or dual attached.

All possible FDDI configurations are shown in Figure 4-6:

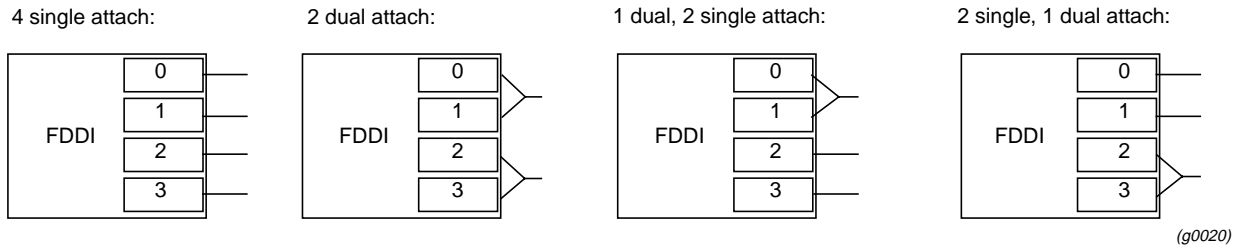


Figure 4-6. DAS and SAS configuration options

Only the top or the bottom pair of FDDI interfaces can be set to dual attach. Interfaces 1 and 2, for example, cannot be paired. Ascend recommends setting unused FDDI interfaces to single. If a site adds a single FDDI connection, the configuration is already set.

At the Card profile, specify SAS/DAS as single or dual in the `ports / fddi` field. This example shows how to set interfaces 2 and 3 as DAS and do a write at the end to save the changes. Here is the path:

```
super> read card 8
CARD/8 read
super> list
card-num* = 8
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 inter-
nal-oscillat+
load = { 0 < > 1 0 0 }
dump = { 0 < > off }
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

A typing shortcut is used here.

```
super> list ports 2 fddi
single-dual = single
optical-bypass = off
super> set single-dual = dual
```

A typing shortcut is also used here.

```
super> list ports 3 fddi
single-dual = single
optical-bypass = off
super> set single-dual = dual
super> write
```

CARD/8 written

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.

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 4-7 illustrates different types of receptacle and connector keys:

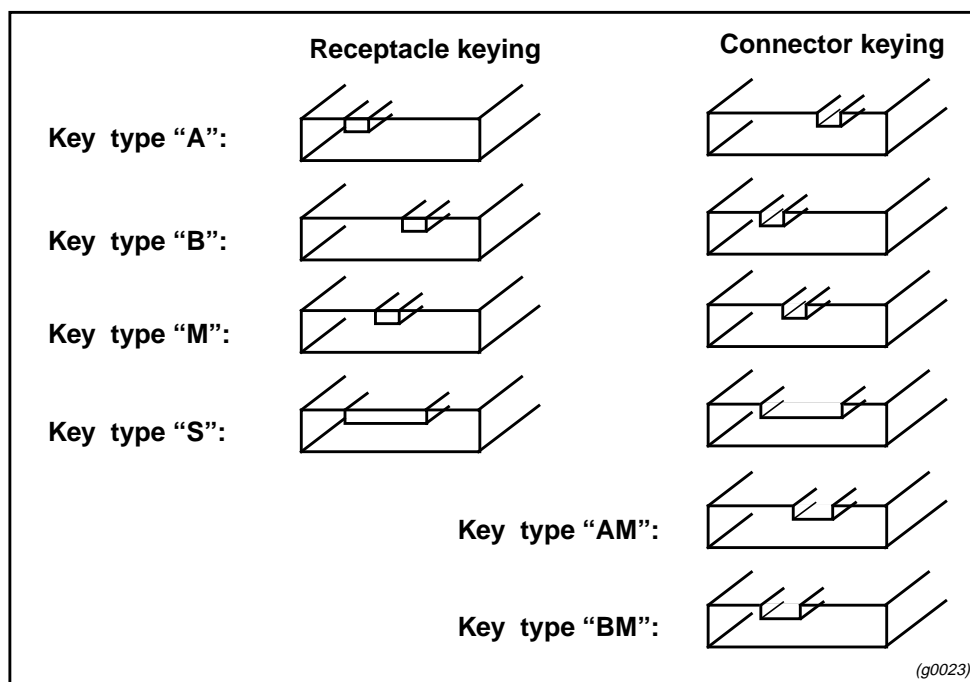


Figure 4-7. Types of FDDI connector keys

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.

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.

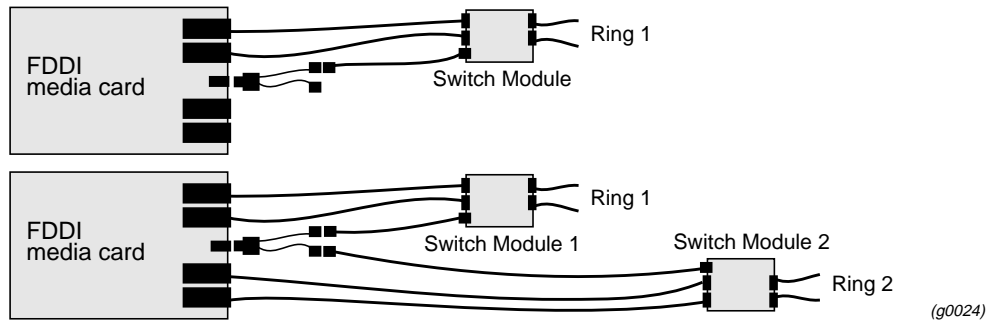


Figure 4-8. Optical bypass switch attachments

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.

Manually activating optical bypass

To manually activate/deactivate the optical bypass on a particular media card, use the Card profile in the `ports / fddi` field.

Here is the path:

```
super> read card 8
CARD/8 read
super> list
card-num* = 8
media-type = fddi
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 inter-
nal-oscillato+
load = { 0 < > 1 0 0 }
dump = { 0 < > off }
```

FDDI Configuration

Optical bypass switch interface

```
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
super> list ports 0 fddi    (if you are at the top level of the profile)
```

or

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

```
super> list 0
port_num = 0
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
disa+
```

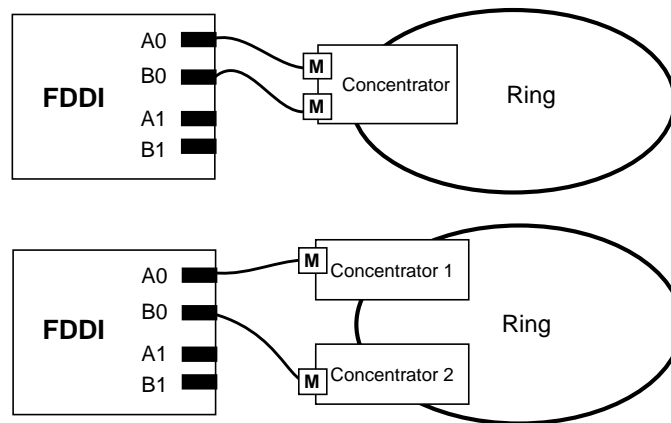
```
super> list ports 0 fddi
single-dual = single
optical-bypass = off
```

```
super> set optical-bypass = on
super> write
CARD/8 written
```

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 4-9, the M ports can be on either one or two FDDI concentrators on the ring.



(g0025)

Figure 4-9. Dual homing options

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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 FDDI 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.

Example

Given an FDDI card installed in slot 2, this example shows where to specify the `spd-tx-thresh=` setting in the Card 2 profile:

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

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/2 written
super> greset 2
```

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
```

Monitoring FDDI media 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

Preparing to use maint

First, switch to the GR 66> prompt. Enter:

```
# grmb
```

The new prompt appears:

```
GR 66>
```

Then change the prompt port to the FDDI media card you are working with.

For the card in slot 7, enter:

```
GR 66> port 7
```

This message is returned along with the changed prompt:

```
Current port card is 07  
GR 07>
```

To leave the **maint** prompt, enter **quit**.

FDDI maint commands

To obtain a list of FDDI CPU0 **maint** commands, enter: `maint 1`

This list is displayed on the screen:

```
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 Combust 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
[CPU0]         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] 70: Send maint command to cpu-1.
```

CPU1 maintains data for certain functions such as ARP. Enter the **maint 70** command to switch to the set of CPU1 commands.

To obtain a list of FDDI card CPU 1 **maint 70** commands, enter:

```
GR 03> maint 70
```

This list is displayed on the screen:

```
GR 03> maint 70 1
```

```
[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]
[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: 0xe (SDC2).
[CPU0] SDC2 Combust Xilinx version: 0x6.
[CPU0] SDC2 Switch Transmit Xilinx version: 0x5.
[CPU0] SDC2 Switch Receive Xilinx version: 0x0.
[CPU0]
```

```
[CPU0] SW:
[CPU0] FDDI Code Version: A1_4_3, Compiled Mon Nov 17 18:12:41 CST
1997,
[CPU0]           in directory: /nit/A1_4_3/fddiq/common.
[CPU0] IPv4 Library Version: 1.4.3, Compiled on Mon Nov 17
18:06:09 CST 1997.
[CPU0] Route Library Version: 1.4.3, Compiled on Mon Nov 17
18:04:23 CST 1997.
[CPU0] IF Library Version: 1.1.0.0, Compiled Mon Nov 17 18:04:47 CST
1997.
```

Verify FDDI configuration

The **maint 3** command returns configuration information for each interface:

```
GRF> 13> maint 3
[CPU0]
[CPU0]           FDDI Slot 13 Configuration and Status Summary:
[CPU0]
[CPU0]           Interface 0      Interface 1      Interface 2      Inter-
face 3
[CPU0]           =====      =====      =====
=====
[CPU0] Single/Dual:  Single      Single      Single      Single
[CPU0] Port(s):    0            1            2            3
[CPU0] Station Cfg:  Wrap S      Isolated     Isolated     Iso-
lated
[CPU0] Primary MAC:
[CPU0] Mac Address: 00c080892906 c080892907 00c080892908 00c080892909
[CPU0] Upstr Nbor:  00c0808929d3 000000000000 000000000000 00001f000000
[CPU0] Dnstr Nbor:  00c0808929d3 000000000000 000000000000
00001f000000
[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

```
GR 13> maint 4
GR 13> [CPU0]
[CPU0]           FDDI statistics
[CPU0]
[CPU0] Input:
[CPU0]  IF      Bytes      Packets      Discards      Errors
```



```
[CPU0]
-----
-
[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]
[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]
[CPU0] Combus Status:
[CPU0] Last interrupt status:      0x0
[CPU0] Combus Statistics:
[CPU0] Message ready interrupts:      1096411
[CPU0] Truncated input messages:      0
[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:                      1096411
[CPU0]  Grid echo requests:                  81082
[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:                   0      Received Long Message:          26

```

Clear all statistics

```

Enter:  maint 7
GR 07> maint 7
All FDDI Statistics Cleared.
All Switch Statistics Cleared.
All Combus Statistics Cleared.

```

Display current ARP table contents

```

Enter:  maint 70 8

GR 13> maint 70 8
GR 13> [CPU0]
[CPU1]
[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

```

Set history trace

Use **maint 9 1** to turn history trace on.
 Use **maint 9 0** to turn history trace off (by default, history trace is off).

Display history trace

Enter: maint 10 9 (9 is number of entries to show)

```

GR 07> maint 10
[CPU1] History Trace:
[CPU1] Index Event      Time      Param1      Param2
[CPU1] =====
[CPU1]   9  arpT      00012176  00000000  00000000
[CPU1]   8  smtS      00008313  010f4034  00000003
[CPU1]   7  rcvD      00011023  00000003  00270b66
[CPU1]   6  smtR      00012541  010f4034  00000003
[CPU1]   5  osRM      00014341  010f4034  00000003

```

```
[CPU1]    4  desc  00015195 010ed220 346c004c
[CPU1]    3  fsiI  00015337 00000003 00001000
[CPU1]    2  fddI  00000288 00000003 00000000
[CPU1]    1  arpT  00011972 00000000 00000000
```

Display SMT MIB variables

Enter: maint 11

```
GR 07> 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> maint 13
[CPU0]
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:0 flags=0xb043<UP,BROADCAST,RUNNING,LINK0,LINK1,MULTI-
CAST> mtu 4352
[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]
[CPU0]      CAM addresses
[CPU0]
[CPU0] Interface 0: +Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
GR 13> [CPU0] 0 01:00:5e:00:00:01
[CPU0] 0:d:0 flags=0xb043<UP,BROADCAST,RUNNING,LINK0,LINK1,MULTI-
CAST> mtu 4352
```

Read MAC_CNTRL_A IF

Enter: maint 15

```
GR 13> maint 15
[CPU0]
[CPU0] 0 8010
[CPU0] 1 0000
[CPU0] 2 0010
[CPU0] 3 0010
```

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]
[CPU0] Interface 0: -Multicast -Promiscuous -BridgeStrip
[CPU0] 0 01:00:5e:00:00:01
GR 13> [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
```

Display CPU1 memory

```
Enter: maint 70 60 address number of words
      maint 70 60: Display CPU-1 private memory.
GR 03> maint 70 60 0x200000 32
GR 03> [CPU0]
      [CPU1]
      [CPU1] CPU-1 displaying 32 words of memory at 0x200000
      [CPU1] 00200000:      43524620 74696d65 6f757420 66646469
      [CPU1] 00200010:      2025640a 00000000 54696d65 64206f75
      [CPU1] 00200020:      74207265 736f6c76 696e6720 4152502e
      [CPU1] 00200030:      20204966 3a202564 20616464 72657373
      [CPU1] 00200040:      2025642e 25642e25 642e2564 0a000000
      [CPU1] 00200050:      42555354 45442120 20546869 73207061
      [CPU1] 00200060:      636b6574 20776173 206d6973 726f7574
      [CPU1] 00200070:      65642c20 616e6420 646f6573 206e6f74
```

Print FSI indirect registers

This command displays additional information on the internal state of FDDI interface hardware. Enter: maint 61 *interface*

```
GR 11> maint 61 0
GR 11>
FSR indirect registers 0:
0:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00
1:      0/ 0x01 1/ 0x03 2/ 0x00 3/ 0x00 4/ 0x02 5/ 0x08 6/ 0x00 7:
0x00
2:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x20 5/ 0x00 6/ 0x00 7:
0x00
3:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x10 5/ 0x80 6/ 0x00 7:
0x00
4:      0/ 0x00 1/ 0x10 2/ 0x20 3/ 0x04 4/ 0x10 5/ 0x20 6/ 0x00 7:
0x00
5:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00
6:      0/ 0x00 1/ 0xC5 2/ 0xD9 3/ 0x00 4/ 0xC2 5/ 0xD6 6/ 0x00 7:
0x00
7:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00
```

```

8:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x3F 7:
0x3F
9:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00
a:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00
b:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x81 7:
0x81
c:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x42 5/ 0x80 6/ 0x03 7:
0x00
d:      0/ 0x00 1/ 0x00 2/ 0x00 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00
e:      0/ 0x00 1/ 0x83 2/ 0x83 3/ 0x81 4/ 0x84 5/ 0x84 6/ 0x00 7:
0x00
f:      0/ 0x00 1/ 0x00 2/ 0xA0 3/ 0x00 4/ 0x00 5/ 0x00 6/ 0x00 7:
0x00

```

Print PHY registers/counters

This command displays internal state information from FDDI
PHY hardware. Enter: `maint 62 interface`

```

GR 13> maint 62 0
[CPU0]
[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 11> maint 63 0
GR 11>
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
```


HIPPI Configuration

5

Chapter 5 describes how to configure HIPPI media cards. The first sections are a brief HIPPI tutorial.

This chapter contains these topics:

Introduction to HIPPI	5-2
Starting a HIPPI connection... ..	5-3
Taking stock... ..	5-8
HIPPI configuration options	5-11
Example 1: Source routing – host selects the path	5-12
Example 2: Using logical addresses	5-14
Example 3: IP routing – HIPPI-to-HIPPI across a switch	5-18
Example 4: IP routing – HIPPI-to-IP media	5-22
Special case 1: HIPPI IPI-3 routing	5-25
Special case 2: IBM H0 HIPPI option	5-26
Configuration files and profiles	5-28
Set up HIPPI media card – Card profile	5-30
Set up HIPPI media card – Load profile	5-33
Set up HIPPI media card – Dump profile	5-34
Monitoring HIPPI media cards	5-36

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 `rfcnxxx.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.

How the I-field is used

Camp-on bit

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 5-2, 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):

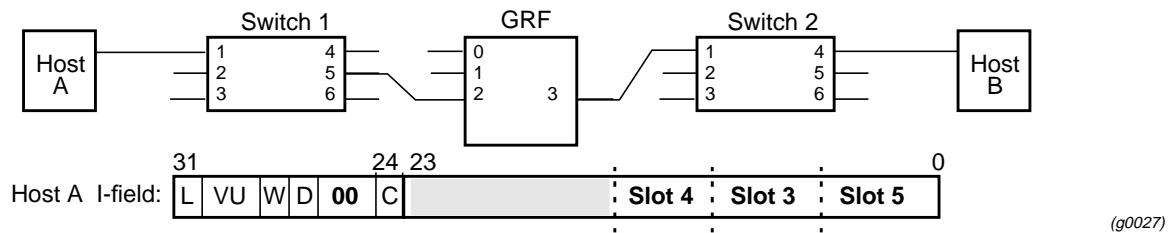


Figure 5-2. I-field for source-directed routing

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 5-3 shows the return path created in the source routing example illustrated in Figure 5-2.

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.

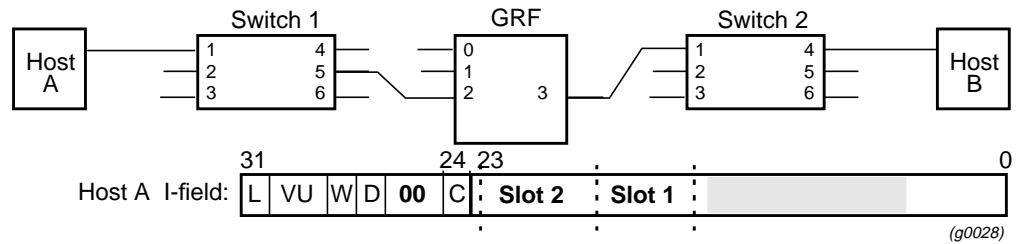


Figure 5-3. Return path created in source routing

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 5-4, the leftmost 12 bits contain the logical address of the Source host; host A.

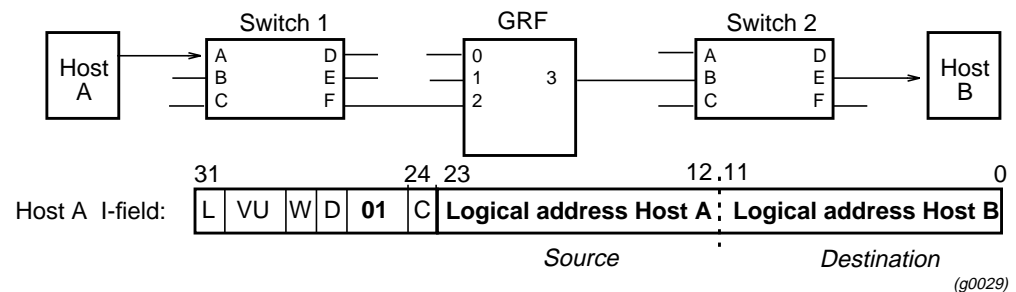


Figure 5-4. I-field for logical addressing (PS is set to 01)

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 = 0xf0, 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:

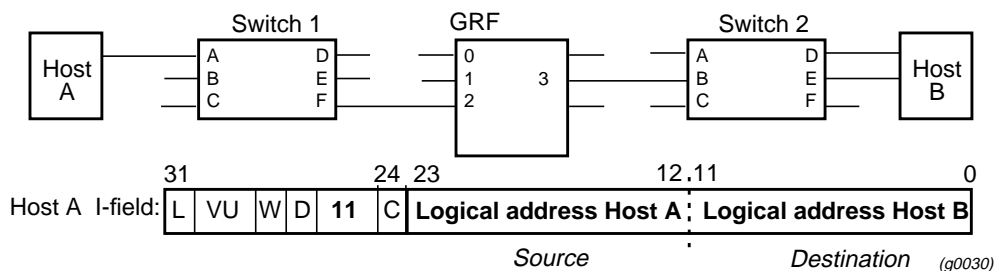


Figure 5-5. *I-field for logical addressing (PS is set to 11)*

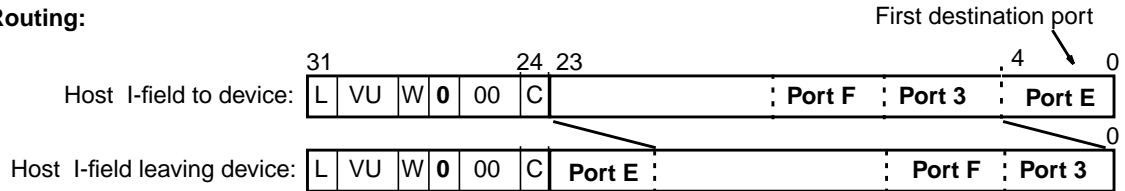
When PS is set to 11, the rightmost 12 bits of the I-field contain the logical address of the target endpoint. The leftmost 12 bits contain the logical address of the Source host, in the example above, host A. Each switch/router has to look up the destination host logical address in its own tables and 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:

Source Routing:



Logical Addressing:

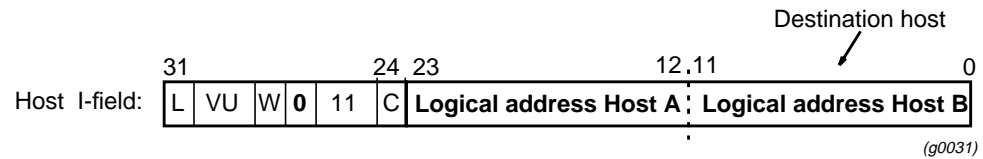
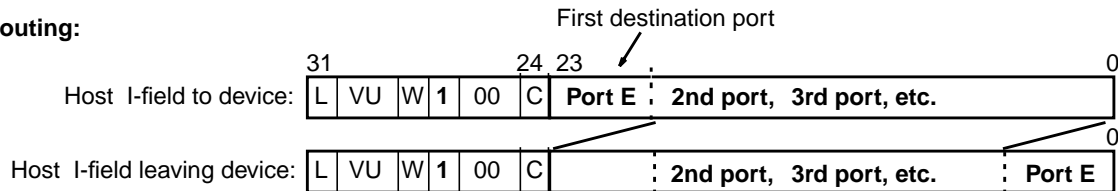


Figure 5-6. Source routing and logical addressing with $D = 0$

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.

Source Routing:



Logical Addressing:

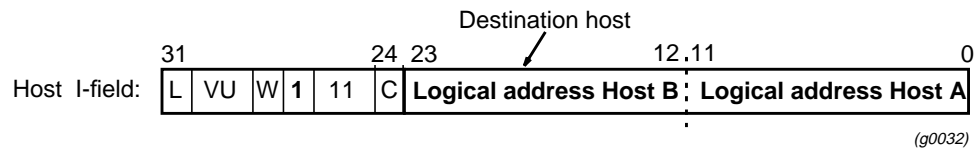


Figure 5-7. Source routing and logical addressing with $D = 1$

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.

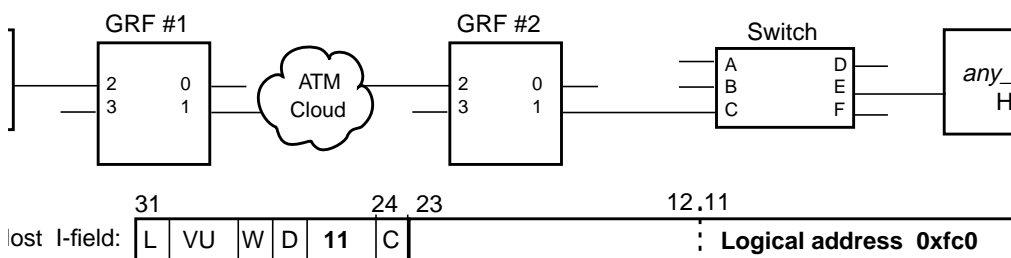


Figure 5-8. I-field 0xfc0 entry that triggers IP routing

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 `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 `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 `grifconfig.conf` file, in the **arguments** field. Refer to the *GRF Reference Guide* for the format of `grifconfig.conf` entries.

ARP

HIPPI ARP tables are manually configured for remote devices connected to GRF HIPPI interfaces. The `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 hippy-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 hippy
ftp> get hippy-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.

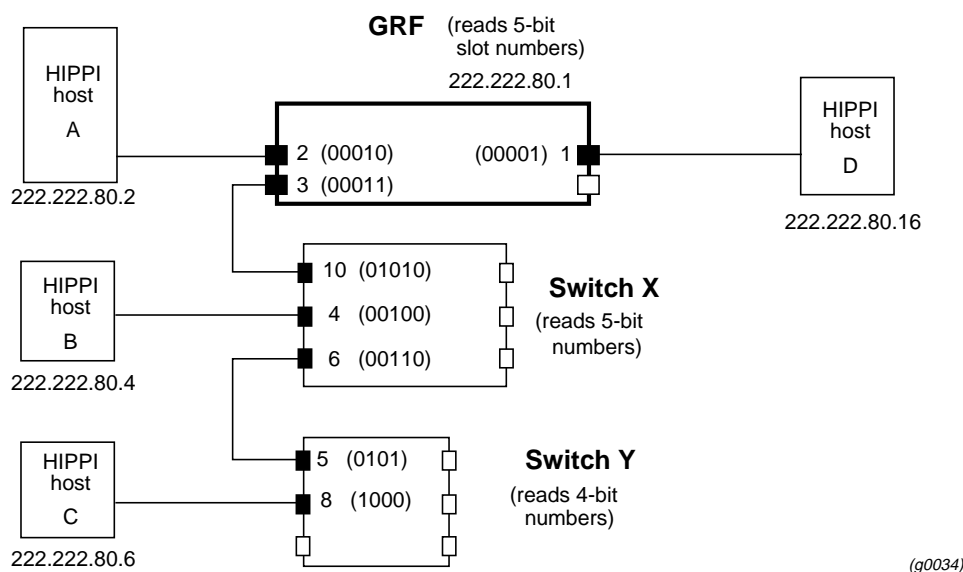


Figure 5-9. Planning diagram for source routing example

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.

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

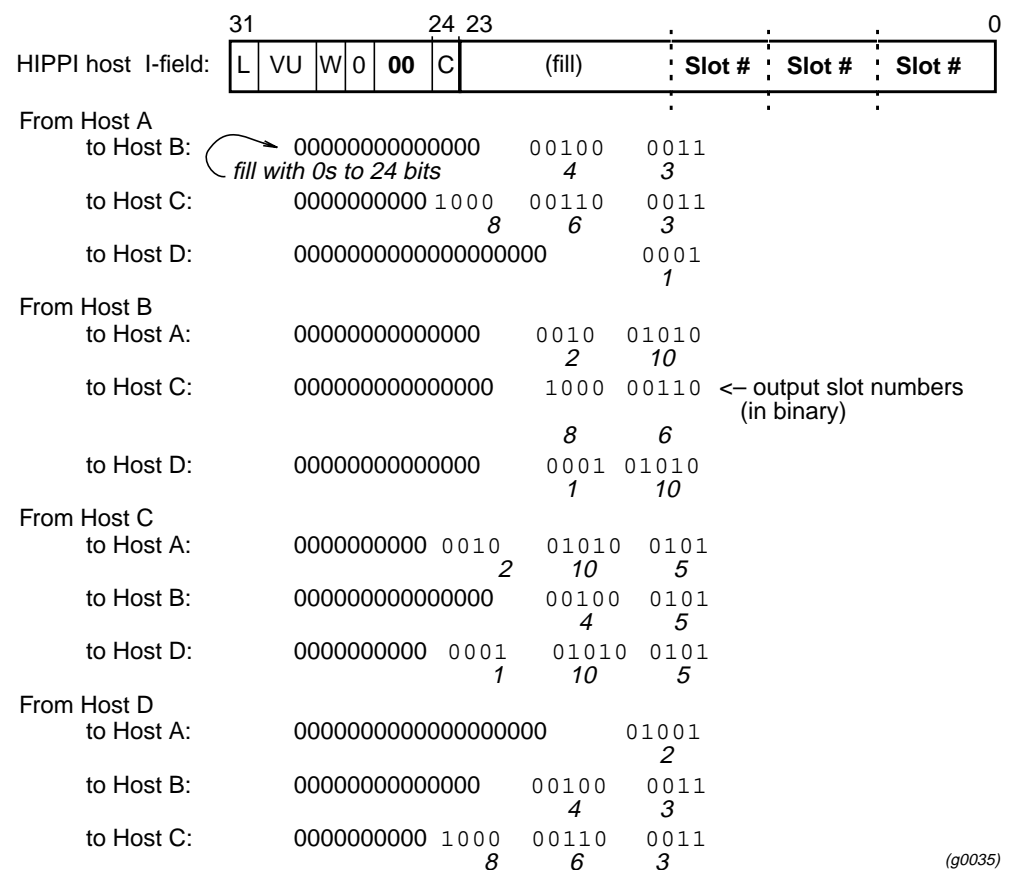


Figure 5-10. I-field list for source routing example

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 **grlmap** 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:

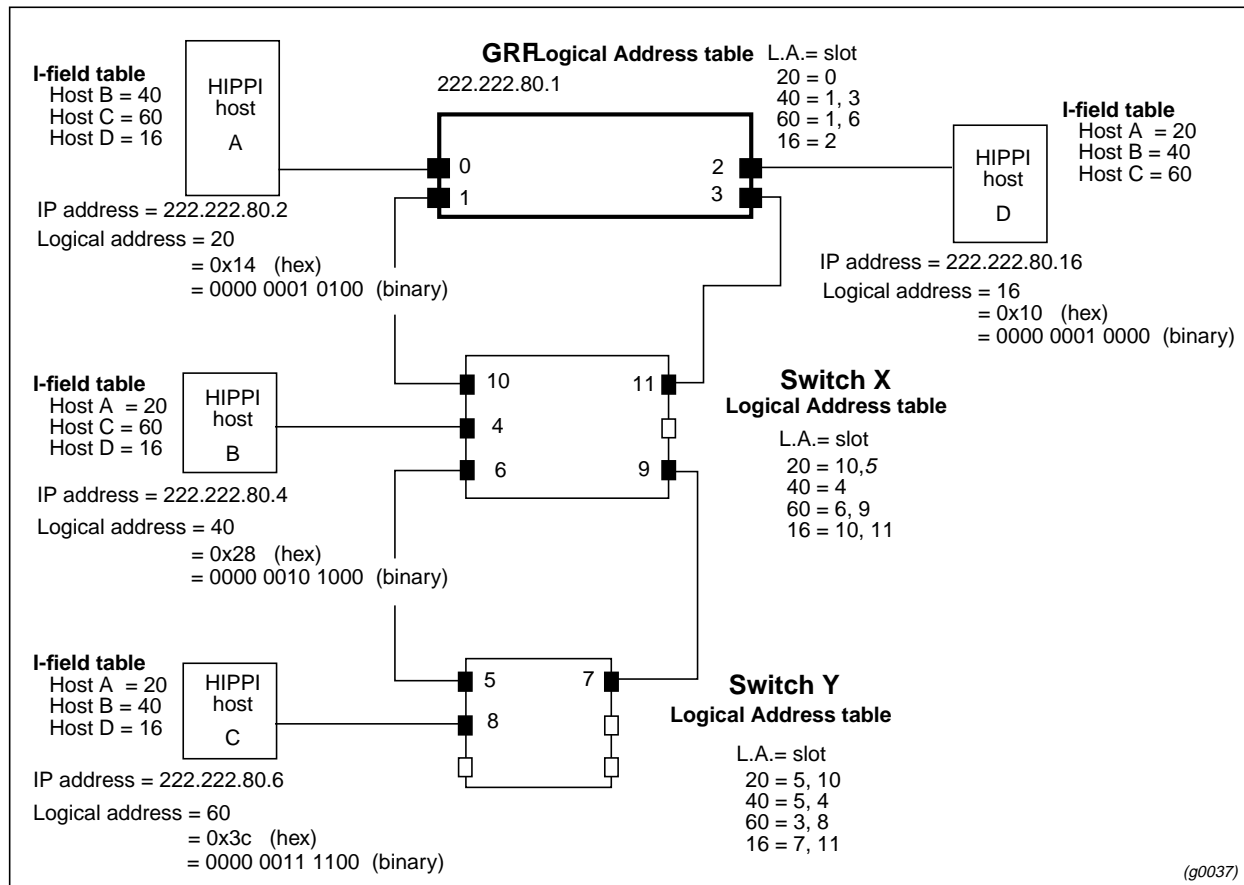


Figure 5-11. Planning diagram for logical addressing

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:

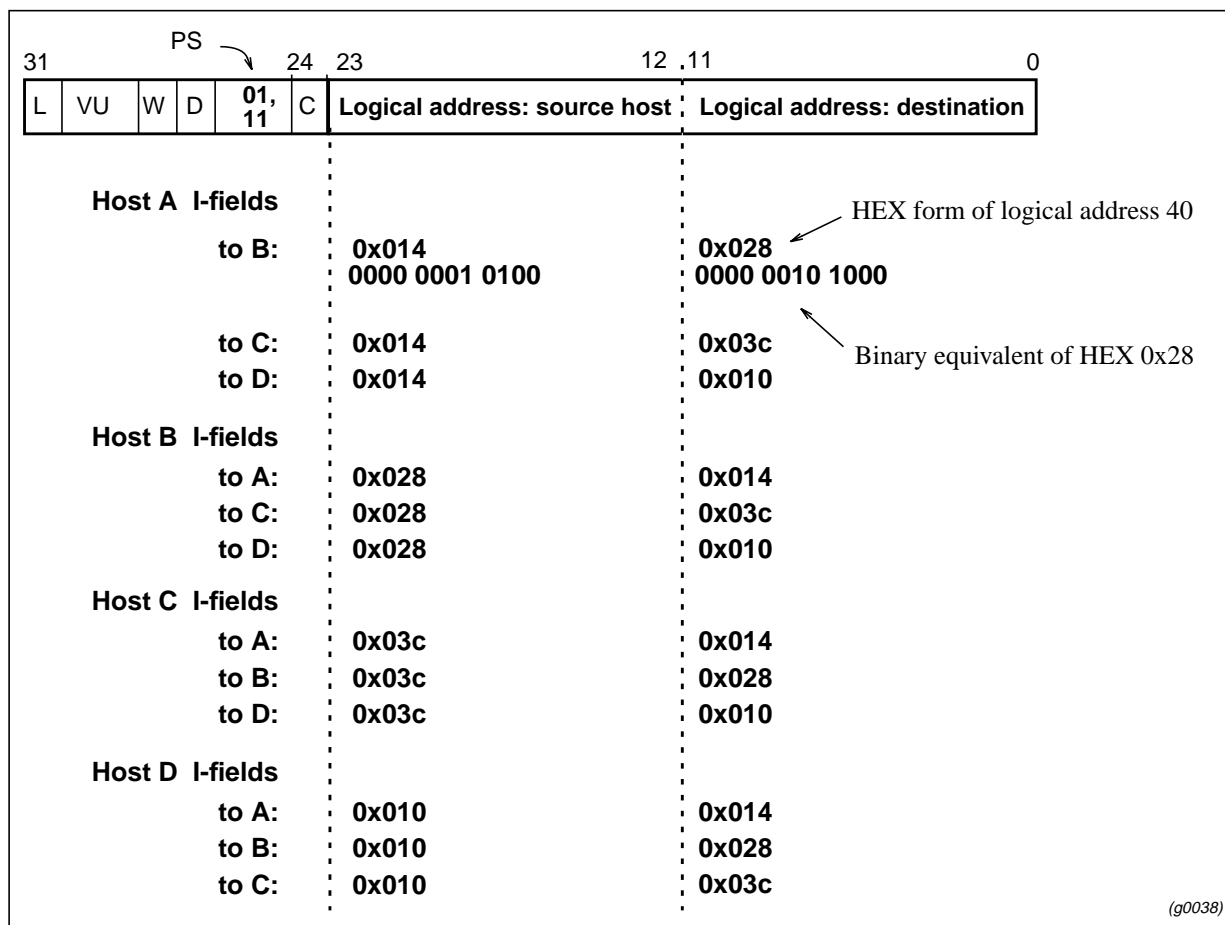


Figure 5-12. Sample host I-field table for logical addressing

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:

<code>portcard</code>	<code>logical_addr</code>	<code>dest_portcard</code>
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 5-13:

- 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

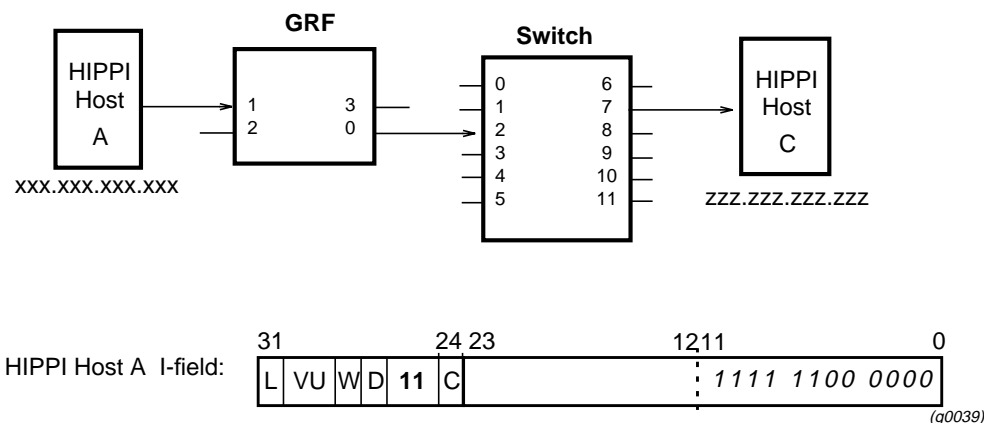


Figure 5-13. Planning diagram for HIPPI-HIPPI IP routing with switch

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 **grarp** 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

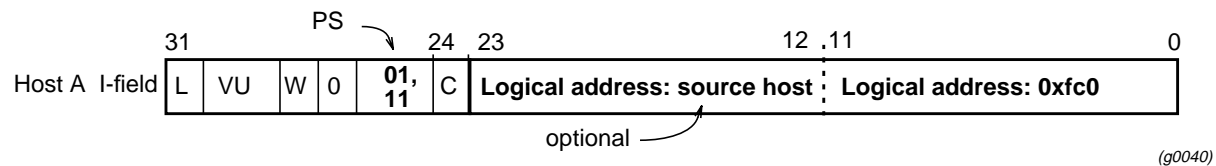


Figure 5-14. I-field for HIPPI-HIPPI IP routing example

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, and `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 `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.

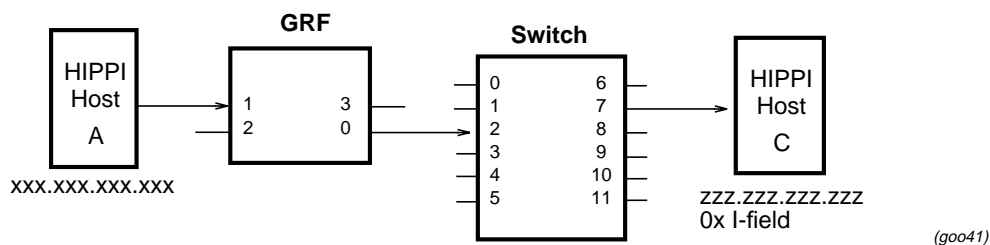


Figure 5-15. Mapping an IP address to a destination I-field

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 5-15.

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:

```
# garp -s zzz.zzz.zzz.zzz <0x i-field_hostC> -i gh000
# garp -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 5-16 contains the example's planning diagram. Both GRFs function as high-performance LAN backbones, and connect over ATM or FDDI.

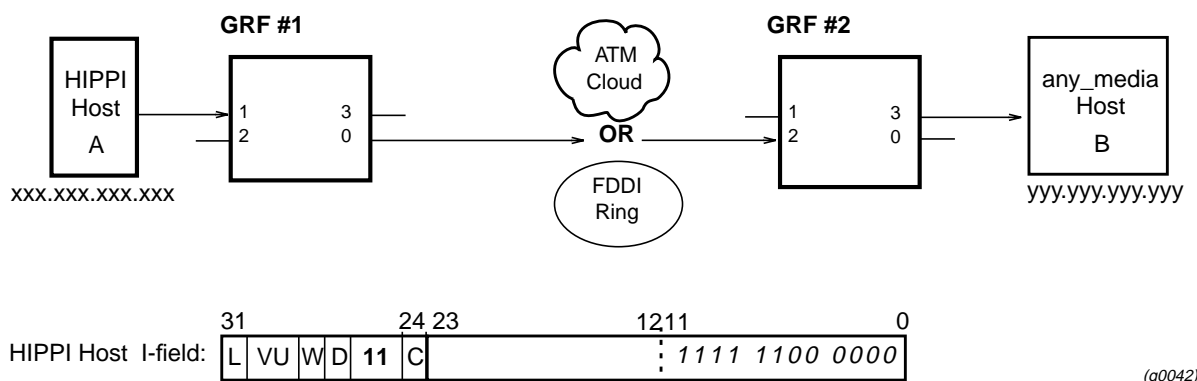


Figure 5-16. Planning diagram for HIPPI IP routing

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 data-grams 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 data-grams 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

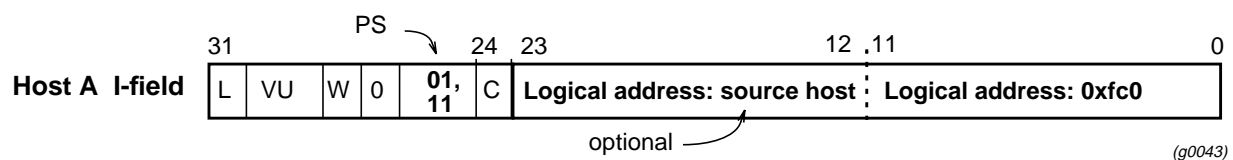


Figure 5-17. I-field for IP routing example

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.

HIPPI Configuration

Example 4: IP routing – HIPPI-to-IP media

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 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 5-18 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.

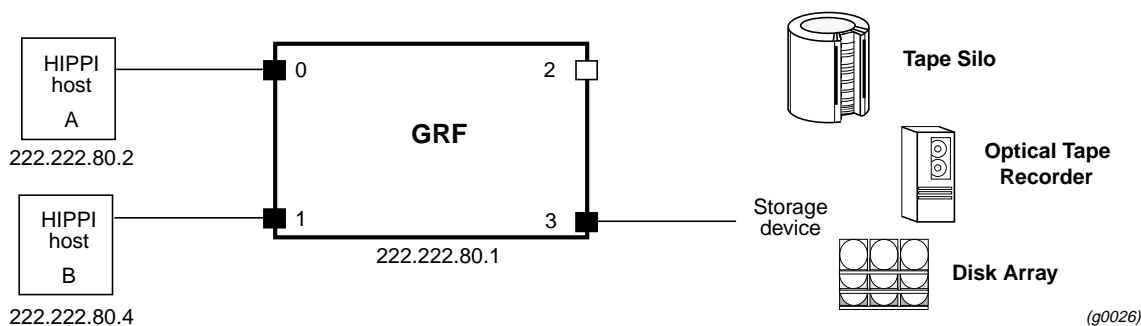


Figure 5-18. Planning diagram for HIPPI IPI-3 configuration

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. Set IBM-H0-mode to enabled.

Here is the path:

```
super> read card 8
CARD/8 read
super> list card 8

card-num* = 8
media-type = hippi
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < {0 {single off} { "" "" 1 sonet internal-oscillator 0} {0
16-bit+
load = { 0 < { fddi 0 "" 0 "" off 0 1 < { 1 fddi "" "" } > } > 1 0
0 }
```

```
dump = { 0 empty 0 }
config = { 0 1 1 0 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
super> list ports
0 = {0 {single off } { "" "" 1 sonet internal-oscillator 0 }}{ 0
16-bit +
1 = {1 {single off } { "" "" 1 sonet internal-oscillator 0 }}{ 0
16-bit +
2 = {2 {single off } { "" "" 1 sonet internal-oscillator 0 }}{ 0
16-bit +
3 = {3 {single off } { "" "" 1 sonet internal-oscillator 0 }}{ 0
16-bit +

super> list 0
port_num = 0
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 disabled di+

super> cd 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-H0-mode = enabled
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
```

The default is IBM_H0 mode disabled.

When you change any setting, you must reset the HIPPI card or reboot the system to start the new mode.

Configuration files and profiles

These are the steps to configure HIPPI cards.

1. Identify each logical interface

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

2. Check I-field shift in the System profile:

There is one configurable item in the System profile for HIPPI, by default, the I-field shift is set to 5 bits:

```
SYSTEM read
super> list
os-level = TA1.4
hostname = r.netstar.com
ip-address = 198.174.11.135
netmask = 255.255.255.0
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 = 40
hardware-revision = "Not Available"
chassis-revision = "Not Available"
xilinx-revision = "Not Available"
num-fans = "Not Available"
num-pwr-supply = "Not Available"
```

3. Specify HIPPI card parameters in the Card profile:

Media card type, `hissi`, is automatically read into the `media-type` field. Other values shown are defaults.

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

3. 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 every HIPPI card.

If you want to change the run-time code in one HIPPI card (per physical interface), make the change in the Card profile, in the `load` field.

4. Dump profile

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 zero (0) which actually stores two dumps per day (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 (per physical interface), make the change in the Card profile, in the `dump / config` field.

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:

```
# greset <slot_number>
```

Set up HIPPI media card – Card profile

Set specific HIPPI card configuration parameters at the Card profile. The fields to set are:

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: change HIPPI host time-out setting
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

Fields you may want to set are in bold, default values are shown here.

Media card type, `hippi`, is automatically read into the read-only `media-type` field. Other values shown are defaults. At the top level, you see ICMP throttling fields:

```

super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
media-type = hippo
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 }
```

Specify ICMP throttling

You can specify ICMP throttling changes for this HIPPI 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> list ic
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

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/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.

Specify HIPPI host time-out values

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> cd hipp
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
CARD8/written
```

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 HIPPI 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_hippi
> super> write
CARD/8 written
```

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 8
card/8 read
super> list dump
> config = 0
hw-table = < >
config-spontaneous = 0
```

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
```


Set up HIPPI media card – 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** 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/xxload.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/port-
cards/hssi_tx.+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/port-
cards/dev1_tx.+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run /usr/libexec/port-
cards/at+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/port-
cards/fddiq+
atm-oc12-v1 = { /usr/libexec/portcards/atm-12.run N/A off 0 1 < > }
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run /usr/libexec/port-
cards/+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/port-
cards/son+
```

Look at the HIPPI card settings:

```
super> list hipp
type = hipp
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/xxload.run N/A}{2
/usr/lib+

super> list boot
1 = { 1 /usr/libexec/portcards/xxload.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.

HIPPI Configuration

Set up HIPPI media card – Dump profile

```
super> list 1
index = 1
hw-type = hippi
rx-path = /usr/libexec/portcards/xlload.run
tx-path = N/A

super> cd ..
super> list bo 2
index = 2
hw-type = hippi
rx-path = /usr/libexec/portcards/runload.run
tx-path = N/A

super> cd ..
super> list bo 3
index = 3
hw-type = hippi
rx-path = /usr/libexec/portcards/hippi.run
tx-path = N/A
```

Set up HIPPI media card – 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 accomodates the default setting of zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0} {fddi 20 /var/portcards/grdump 2} {rmb 20
/v+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi core
mem+
config-spontaneous = off
> keep-count = 0
>
```

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 }
devl = { devl 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>
```

Monitoring HIPPI media cards

The HIPPI **maint** command displays a range of information about the media card:

- **maint** command options
- media card configuration and status
- HIPPI media statistics
- print logical address mappings
- print IP statistics

Several management operations can be performed with the **maint** command:

- clear statistics counters
- enable debug printouts
- dump trace buffers and tables

Preparing to use maint

First, switch to the GR 66> prompt. Enter:

```
# grrmb
```

The new prompt appears:

```
GR 66>
```

Then change the prompt to the slot occupied by the HIPPI media card. For slot 3, enter:

```
GR 66> port 3
```

This message is returned along with the changed prompt:

```
Current port card is 03  
GR 03>
```

To leave the **maint** prompt, enter **quit**.

List of HIPPI maint commands

To obtain a list of **maint** commands, type:

```
GR 03> maint 1
```

This list is displayed on the screen:

```
GR 03> HIPPI maint commands:  
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
cploop_count
```

Print IP statistics

```
Enter: maint 133
GR 03> HIPPI IP statistics:
13894 total HIPPI IP connections
13894 total packets received
15 packets forwarded
13879 packets not forwardable
Interface IP statistics:
ipstat[0].dropped = 13879
ipstat[0].forwarded = 15
ICMP statistics:
icmpstat[0].echo_req_returned = 13894
```

15 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 03> maint 132
GR 03>      0.0 CP 0002 60020401 0000055f Switch Output DMA start
1667.3 MP 00ce 00080028 00430000 SDC XMIT intrpt (ctrl | stat, swsctrl)
5.4 MP 00cf 00200004 00430000 SDC RECV intrpt (ctrl | stat, portaddr)
3143.2 CP 0004 60020401 00900124 HIPPI Output DMA start
285411.6 MP 00a5 00000026 00000006 Main loop probably hung
1353373.9 MP 0021 00000001 00000000 Initializing HIPPI Destination
16.6 CP 0004 40220000 0090f400 HIPPI Output DMA start
9.2 CP 0001 40000400 0090d400 HIPPI Input DMA complete
5.9 MP 0010 40000400 00000000 hipp_i_in, start of connection
19.6 MP 0019 00000000 00000000 Accepting connection
27.4 CP 0004 9802060f 0090f000 HIPPI Output DMA start
14.3 CP 0001 9820000f 00912000 HIPPI Input DMA complete
36.9 CP 0004 20220200 00000000 HIPPI Output DMA start
9.0 CP 0001 20000600 00000000 HIPPI Input DMA complete
5.5 MP 0011 20000600 00000000 hipp_i_in, end of connection
12.2 MP 0021 00000000 00000000 Initializing HIPPI Destination
43058.5 MP 0060 00000009 00000000 IPADDR command
7.7 MP 0061 00000000 00010007 IPADDR IA_SET command
7.7 MP 0061 c6ae3201 00010001 IPADDR IA_SET command
20.8 MP 0060 00000006 ffffffff IPADDR command
7.6 MP 0067 ffffffff 0007ffff IPADDR IA_SAVE command
49090421.503316482 MP 0045 00000000 00000000 Maint command 133
29519363.335544323 MP 0040 00000001 00000000 Maint command 128
11861814.167772169 MP 0040 00000001 00000001 Maint command 128
83657445.838860805 MP 0042 00000001 00000010 Maint command 130
```

29513727.335544324 MP 0044 00000000 00000000 Maint command 132

Print switch error counts

Enter: maint 141

```
GR 03> maint 141
GR 03> 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 receive 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
```

HIPPI tunnel statistics

Enter: maint 151

```
GR 03> maint 151
GR 03>
Tunnel: In = DropAll Out = DropAll, HIPPI: In = DropAll Out = DropAll
Debug -----> 0x00000000
I-field -----> 0x00000000
Credit Count --> 0x00000000
Old Credit ----> 0x00000000
Init Credit ----> 0x00000000
New Credit ----> 0x00000000
Pdu Count -----> 0x00000000
Pdu Expect -----> 0x00000000
Total Readys --> 0x00000000
Readys Sent ----> 0x00000000
TO Connect 0x00000000 Init 0x00000000 Credit RX 0x00000000 Credit TX 0x00000000
Drop HRX 0x00000000 HTX 0x00000000 TRX 0x00000000 TTX 0x00000000 XRX 0x00000000
LB Dropped ----> 0x00000000
STAT RX V = 0x00000000 I = 0x00000000 D = 0x00000000 P = 0x00000000
          p = 0x00000000 E = 0x00000000 R = 0x00000000 L = 0x00000000
          H = 0x00000000 B = 0x00000000 N = 0x00000000
STAT TX V = 0x00000000 I = 0x00000000 D = 0x00000000 P = 0x00000000
          p = 0x00000000 E = 0x00000000 R = 0x00000000 L = 0x00000000
          H = 0x00000000 B = 0x00000000 N = 0x00000000
```

Show ARP table entries

Enter: maint 156

GR 03> maint 156

GR 03>

ARP entries						
IP	MAC	TTL	I-field	flags		
192.168.023.146	00 00 00 00 00 00	600	03000fc0	permanent		

GR 09

HSSI Configuration

6

Chapter 6 provides information needed to configure framing protocols on the HSSI media card. Three framing protocols are supported over HSSI: Frame Relay, Point-to-Point (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.

This chapter contains these topics:

HSSI (High Speed Serial Interface) implementation	6-2
Framing protocol options	6-2
Physical and logical interfaces	6-5
Card connection options	6-7
Cyclic redundancy checking (CRC)	6-9
Configuration file and profile overview	6-11
Set up HSSI media card – Card profile	6-13
Set up HSSI media card – Load profile	6-18
Set up HSSI media card – Dump profile	6-19
Selective packet discard	6-21
Configuring the HDLC protocol	6-23
Configuring the Frame Relay protocol	6-26
Configuring Point-to-Point Protocol	6-32
Contents of grppp.conf file	6-37
Monitoring HSSI media cards	6-38

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.

Framing protocol options

The HSSI card can run the same or different 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*

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.

Protocol per interface support

The HSSI media card has two physical interfaces. Each interface can run a different protocol. Interface 0 can run Frame Relay while interface 1 runs HDLC.

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

HSSI card software maintains route tables containing up to 150K entries, and hardware support for full table lookups.

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.

Setting a congestion and discard threshold is described in the “Selective packet discard” section later in this chapter.

Controlled-load (class filtering)

Controlled-Load is supported on 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 for information about constructing class filters.

Physical and logical interfaces

This diagram shows the organization of physical and logical interfaces on the GRF HSSI media card:

HSSI card:	Frame Relay (256 / card)	PPP	Cisco HDLC
Physical interface 0 (top)	128 logical interfaces <i>Numbered 0 – 7f</i>	1 logical interface <i>Numbered 0</i>	1 logical interface <i>Numbered 0</i>
Physical interface 1 (bottom)	128 logical interfaces <i>Numbered 80 – ff</i>	1 logical interface <i>Numbered 1</i>	1 logical interface <i>Numbered 1</i>

(g0053)

Figure 6-1. Logical interfaces supported per HSSI physical interface

Physical interfaces

A HSSI media card supports two physical interfaces (connectors).

As shown in Figure 6-1, a physical interface supports either 1 or 128 logical interfaces, depending upon which protocol is running on the HSSI card.

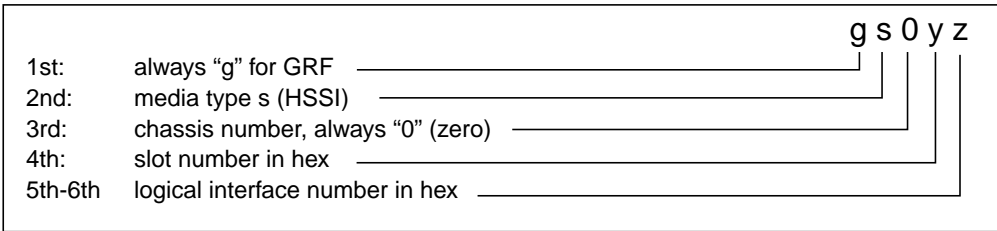
Each physical HSSI interface on the same card can run a different protocol. A HSSI card is not dedicated to a single protocol.

Logical interfaces

A logical interface is configured by its entry in the `grfconfig.conf` file where it is assigned an IP address and netmask. A logical interface is uniquely identified by its HSSI interface name.

Interface name *gs0yz*

As shown in Figure 6-2, the generic form of a HSSI interface name is: *gs0yz*



(g0054)

Figure 6-2. HSSI interface name

where:

- the "gs" prefix indicates a GRF HSSI interface

- the chassis number is always “0”
- “y” is a hex digit (0 through f) for the chassis slot number
- the “z” logical interface number ranges from 0 through ff and identifies each of the logical interfaces to be configured per HSSI card

Identify logical interfaces in grifconfig.conf

Configure each logical interface in the `grifconfig.conf` file by assigning it an IP address, a GRF interface name, and if required, a netmask and destination/broadcast address. The template for the `grifconfig.conf` file is in the *GRF Reference Guide*.

Here is the format for `grifconfig.conf` file entries:

#	name	address	netmask	broad_dest	arguments
gs037e		192.0.2.1	255.255.255.0	192.0.2.255	mtu xxxx
gs037f		192.0.99.1	255.255.255.0	192.0.99.255	

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

A HSSI card that is to run the PPP or HDLC protocol requires one or two entries into the `grifconfig.conf` file since these protocols support one logical interface on each of the physical interfaces.

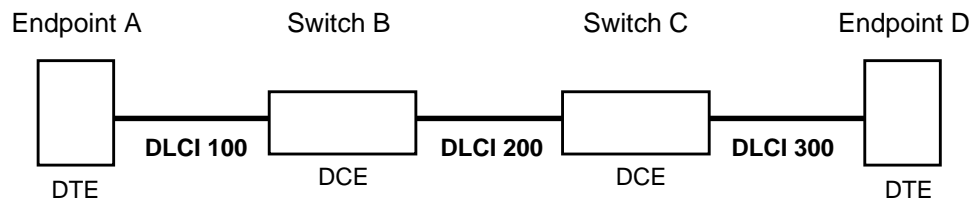
A HSSI card that is to run Frame Relay will have as many as 256 entries since Frame Relay supports 128 logical interfaces per physical interface.

Card connection options

Link type

You can specify a Frame Relay link to be UNI-DTE, UNI-DCE or NNI.

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. An NNI link is the link between two switches inside the network. Both switching and routing can be performed on UNI-DCE and NNI links. Only routing can be performed on UNI-DTE links.



In the above example, 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.

DCE connection

The HSSI card can connect to a CSU/DSU or to a DCE switch via copper twisted-pair cable.

DTE connection

A null-modem cable is required to connect the HSSI card directly to another DTE device.

Null modem cabling (HSSI crossover cables)

When a null-modem cable is used, each DTE must be set to enable internal clock generation.

Where to set

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. The procedure is included in the Card profile section.

Here is the path:

```
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-oscillator+
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 = {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+
```

Now list the specific HSSI port you want to set, 0 or 1:

```
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 32-bit }
ether = { autonegotiate }
hippi = {1 32 no-mode 999999 4 increment 5 300 10 10 03:00:0f:c0
disable+
```

Now list the HSSI field for that interface (a typing shortcut is used):

```
super> list hs
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
```


Cyclic redundancy checking (CRC)

Cyclic redundancy checking (CRC) is a frame verification system, and can be based on either 16- or 32-bit checking.

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 CRC bit setting must match between the two connected endpoint nodes/devices. An intervening CSU/DSU is not involved in the setting.

Devices running Frame Relay usually are set to 16 bits. The Cisco HDLC default is 16-bit CRC. PPP devices usually use 16-bit checking. To disable CRC, set the parameter to zero.

Where to set

The `CRC-type` field is set at the Card profile, in the `ports / hssi` field.

Here is the path:

```
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 {single off} { "" "" 1 sonet internal-oscillator 0} {0
16-bit+
load = { 0 < { fddi 0 "" 0 "" off 0 1 < { 1 fddi "" "" } > } > 1 0
0 }
dump = { 0 empty 0 }
config = { 0 1 1 0 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }

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

Now list the specific HSSI port you want to set, 0 or 1:

```
super> list 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { dual 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+
```

Now list the HSSI field for that interface (a typing shortcut is used):

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

Tip:

To set this value to 16-bit for the card in slot 8, you can use this shortcut:

```
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.

Configuration file and profile overview

These are the steps to configure HSSI interfaces and protocols:

1. *Identify each logical interface*

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

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 %
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

3. *Configure the framing protocol*

Protocol is assigned per physical interface. One HSSI interface can run PPP while the other interface runs Frame Relay.

Cisco HDLC - after steps 1 and 2, set the HDLC parameters specified in the Card profile

Frame Relay - after steps 1 and 2, set Frame Relay and PVC parameters in the `grfr.conf` configuration file

Point-to-Point Protocol - after steps 1 and 2, set PPP parameters in the `grppp.conf` configuration file

4. *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 every HSSI card.

If you want to change the run-time code in one HSSI card (per interface), make the change in the Card profile, in the `load` field.

5. *Dump profile*

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 zero (0) which actually stores two dumps per day (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 HSSI card (per interface), make the change in the Card profile, in the `dump` field.

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:

```
# greset <slot_number>
```

Set up HSSI media card – Card profile

Set specific HSSI card configuration parameters at the Card profile. The fields to set are:

- framing protocol: `cisco-hdlc`, `ppp`, `frame-relay` (default is Frame Relay)
- source clock: 0 and 1, with 1 equal to null modem) (default is 0)
- CRC type: the CRC type to match connecting endpoint (not the CSU/DSU), options are 16-bit, 32-bit, and 0 (default is 16-bit)
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard %
- OPTIONAL: set card-specific load variables
- OPTIONAL: set card-specific dump variables

Fields you may want to set are in bold, default values are shown here.

1. Specify framing protocol

At the top level, you can set the framing protocol. Values are:

- `Cisco-HDLC`
- `PPP`
- `Frame-Relay`

Media card type, `hssi`, is automatically read into the read-only `media-type` field. Other values shown are defaults.

```
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 inter-
nal-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. Specify source clock

You specify clock settings for HSSI interfaces 0 and 1 in the `ports` section:

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

Here is a shortcut you also could use to get to interface 1:

```
super> list ports 1
super> list 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+
```

(Note that you can specify HDLC settings in the port 1 `cisco-hdlc` field (described in the HDLC configuration section).)

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 bits:

```
super> cd hssi
source-clock = 0
CRC-type = 16-bit
```

Change the `source-clock = 0` setting to 1 to put internal clock generation on:

```
super> set source-clock = 1
super> write
CARD/8 written
```

3. Specify 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. The `CRC-type = 16-bit` default is shown:

```
super> list ports 1
super> list 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+
super> cd hssi
source-clock = 0
CRC-type = 16-bit
super> set CRC-type = 32-bit
super> write
CARD/8 written
```

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
```

4. Specify ICMP throttling

You can specify ICMP throttling settings 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 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]
```

Default values are shown:

```
super> list ic
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 (SPD)*

Specify a SPD percentage for this HSSI card in the `spd-tx-thresh` field. This field is contained in the `config` field of the top-level Card profile.

Given a HSSI card installed in slot 8, this example shows where to specify the `spd-tx-thresh=` setting in the Card 8 profile:

```
super> read card 8
CARD/2 read

super> list
card-num* = 8
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 inter-
nal-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 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 decide an appropriate SPD percentage is provided in the “Selective packet discard” section later in this chapter.

6. *Specify 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/port-
card/test_executable_for_hssi
super> write
CARD/8 written
```

7. 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 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
```

Set up HSSI media card – 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** 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/xlload.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/port-
cards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/port-
cards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run
/usr/libexec/portcards+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/port-
cards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-l2.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/port-
cards/+
```

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:

```
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 = < >
super> set enable-boot-seq = on
super> write
LOAD written
```

Set up HSSI media card – 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 zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0}{fddi 20 /var/portcards/grdump 2} {rmb
20 /v+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi
core +
config-spontaneous = off
keep-count = 0
```

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 = { hippy 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.

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 mem-
ory" 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 ..
```

HSSI Configuration

Set up HSSI media card – Dump profile

```
super> list s 2
index = 2
description = "hssi shared SRAM memory"
start = 131072
length = 32768
```

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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 HSSI 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 GateD 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.

Example

Given an HSSI card installed in slot 2, this example shows where to specify the `spd-tx-thresh=` setting in the Card 2 profile:

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

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/2 written
super> grreset 2
```

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
```

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:        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
```


Configuring the HDLC protocol

1. Specify interface names in *grifconfig.conf*

Identify the two logical interfaces on the HSSI card. Here are sample entries in *grifconfig.conf* for two logical interfaces on the HSSI card in slot 2:

#	name	address	netmask	broad_dest	argument
gs020		192.0.2.1			
gs021		192.0.99.1			

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

2. Specify framing protocol

Set the framing protocol in the Card profile.

```
super> read card 8
CARD/8 read

super> list
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 }

super> set hssi-frame-protocol = cisco-hdlc
super> write
CARD/8 written
```

3. Set Cisco HDLC settings

- check `cisco-hdlc` default settings
- leave internal clocks set to 0 (unless you are using a null-modem cable)
- set CRC to 32-bit if the MTU for the device on the other end of the wire is over 4096, other wise, do not change 16-bit setting.

```
super> list ports
0 = { 0 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
1 = { 1 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
2 = { 2 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
3 = { 3 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
```

```
4 = { 4 { off on 10 3 } {single off} {"" "" 1 sonet internal-oscilla-
tor 0}+
5 = { 5 { off on 10 3 } {single off} {"" "" 1 sonet internal-oscilla-
tor 0}+
6 = { 6 { off on 10 3 } {single off} {"" "" 1 sonet internal-oscilla-
tor 0}+
7 = { 7 { off on 10 3 } {single off} {"" "" 1 sonet internal-oscilla-
tor 0}+

super> list ports 0
port_num = 0
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
dis+
```

Note: remember to specify the `keepalive-interval` setting in milliseconds.

The default Cisco HDLC settings are:

```
super> list cis
debug = off
```

- 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-enabled = on
```

- Keepalive activity can be turned off, the default is on.

```
keepalive-interval = 10
```

- The default keepalive interval setting specifies how often the HSSI interface will send keepalive messages, the default is every 10 milliseconds.

```
keepalive-error-thresh = 3
```

- 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.

Tip: Use `set <field_name>?` to find out the available values to set.

```
super> set keepalive-enabled?
keepalive-enabled:
  Turn Cisco HDLC keepalive on and off
  Boolean field, 'off' or 'on'

super> set keepalive-interval?
keepalive-interval:
  Number of seconds between keepalives
  Numeric field, range [0 - 32767]

super> set keepalive-error-thresh?
keepalive-error-thresh:
  Number of keepalives sent without response before marking the
  interface
```

```
down.  
Numeric field, range [0 - 32767]
```

Make sure you have set source clock and SRC settings

Make sure you have the correct settings for the port on which you want to run HDLC. Check the `port 0 hssi` or `ports 1 hssi` field in the appropriate Card profile.

```
super> list ports 0  
port_num = 0  
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  
dis+
```

Configuring the Frame Relay protocol

1. Specify interface names in *grifconfig.conf*

Identify the logical interfaces on the HSSI card. As an example, we will configure Frame Relay on four logical interfaces on the HSSI card in slot 2. They are logical interfaces 126 and 127 on the upper physical interface, and logical interfaces 128 and 129 on the lower physical interface. Null-modem cables are attached.

# name argument	address	netmask	broad_dest
#			
gs027e	192.0.2.1	255.255.255.0	192.0.2.24
gs027f	192.0.99.1	255.255.255.0	192.0.99.25
gs0282	192.0.130.1	255.255.255.0	
gs0283	192.0.126.1	255.255.255.0	
gs020	192.0.2.4	255.255.255.0	
gs0280	192.0.130.2	255.255.255.0	

Always use lower case letters when defining interface names.

If an interface is nonbroadcast (NBMA), do not include a destination address in its *grifconfig.conf* entry.

First interface configuration requirement

When used in Frame Relay mode, HSSI card configuration requires that the first logical interface on each physical interface must be configured. This enables the frame relay daemon (**fred**) to properly communicate LICS traffic via the card.

For example, if a HSSI card in slot 2 is configured for Frame Relay, then the following interfaces need to exist in the card configuration:

gs020	- slot 2, physical interface 0, logical interface 0
gs0280	- slot 2, physical interface 1, logical interface 0

They also need to be configured as active PVCs in */etc/grfr.conf*. For example:

```
#/etc/grfr.conf file entry :  
pvc gs020 335 192.0.2.4 Name="north subnet"  
pvc gs0280 642 192.0.130.2 Name="south subnet"
```

2. Specify framing protocol

Be sure the framing protocol is set to Frame-Relay the Card profile.

```
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-oscillator 0}+
load = { 0 < > 1 0 0 }
dump = { 0 < > off off}
config = { 0 1 1 4 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

3. Set internal clocks and the CRC

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

super> list ports 0

port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { dual 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
dis+

super> cd hssi
source-clock = 0
CRC-type = 16-bit

super> set source-clock = 1
super> write
CARD8/written
```

In this example, null-modem cables are used with both connectors.

The next steps are in the `/etc/grfr.conf` configuration file. A copy template of the `grfr.conf` configuration file is in Appendix A of the *GRF Reference Guide*.

Configure link parameters

4. Set Link parameters in *grfr.conf*.

Open the UNIX shell:

```
super> sh
```

Use a UNIX editor to edit the *grfr.conf* file to set (or change) the link parameters, then execute the **grfr** command to make the configuration (or changes) permanent. Refer to the **grfr** man page for more information.

Note: For Link parameter settings and changes to take effect, you must reboot the HSSI card.

Required link parameter descriptions

The format for the link parameter entry is:

```
Link slot port
```

where *slot* is the GRF chassis slot and *port* is upper (0) or lower (1). Examples are:

```
link 3 1
link 3 0
```

Optional link descriptors

Name=	- string describing specific interface
Enabled=	- enables/disables link, Y N
LMIType=	- None Standard AnnexA AnnexD
N391=	- sets polling intervals per full status message, range is 1–255
N392=	- error reporting threshold, range is 1–10
N393=	- error recovery threshold, range is 1–10
T391=	- heartbeat poll interval, settings are 5 10 20 25 30
AutoAddGrif	- automatically adds PVCs from LMI to this logical interface

Link descriptor examples

```
link 3 0 LMIType=AnnexD AutoAddGrif=gs060
link 3 1 LMIType=AnnexD name="Lower Port"
```

Configure PVCs

5. Create PVCs and set parameters.

Open the UNIX shell:

```
super> sh
```

Use a UNIX editor and edit `grfr.conf` to add/delete PVCs or to change PVC parameters. Then execute the **grfr** command (as `super_user`) to immediately install the configuration:

The **grfr** command format to install the `grfr.conf` file parameters is:

```
# grfr -p slot_number
```

Note: To add or edit PVCs at any time, on the fly, use the **grfr** command. See section below.

Number of configurable PVCs

The HSSI card supports a total of 905 permanent virtual circuits (PVCs) per physical interface. These can be divided according to site requirements among the 256 logical interfaces.

DLCI numbering

PVC numbering is called the data link circuit identifier, DLCI. The range of valid DLCI numbers is 16 through 921. DLCI 0 to 15 are reserved. When ANNEX A or D is selected, DLCI-0 is used for LICS traffic.

Required PVC parameters

Three parameters are mandatory: Logical Interface, DLCI, and Peer IP address.

<code>lif</code>	- a logical interface described in the Interface Section, <code>gs0yz</code>
<code>DLCI</code>	- the data link circuit identifier, ranges from 16–921
<code>Peer IP Address</code>	- IP address of station at other end of the PVC, specify 0.0.0.0 to resolve the address using Inverse ARP.

Optional PVC parameters

<code>Name=</code>	- string describing PVC
<code>Enabled=</code>	- enables/disables PVC, Y N

PVC examples

```
PVC gs027e 100 0.0.0.0 Name="station0"
PVC gs027f 101 222.222.222.101 Name="station1" Enabled=N
PVC gs0280 102 222.222.222.102 Name="station2"
PVC gs0281 103 222.222.222.103 Name="station3"
```

Adding a logical interface on-the-fly

You can add a logical HSSI interface to `grifconfig.conf` and activate the new PVC without resetting the media card. Do the following in this order

```
# grfr -p <slot number>
# grifconfig -I gs0yz
```

where `gs0yz` is the new logical interface just added. The **grfr** command creates the logical interface and brings up the PVC. The **grifconfig -I** command assigns an IP address to it.

If the interface already exists and you are just adding another PVC (DLCI) from it, then all you need to do is execute the **grfr** command. You cannot change link types on the fly, you can only add, delete, or modify PVCs.

Adding 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 add a PVC or **grfr -c crp** to disable a PVC. You cannot change link type dynamically.

To add a PVC to the 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**.

Enter the **grfr -c ccp** command to add a PVC. The configuration file and the PVC DLCI, slot, and link must be specified:

```
# grfr -c ccp -f /etc/grfr.conf -i 606 -s 13 -l 0
```

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** command is sufficient. Specify the target DLCI to be disabled:

```
# grfr -c crp -i 600 -s 13 -l 0
```

Here is the response:

```
PVC slot 13, link 0, dlci 600 deleted
```

Assigning multiple DLCIs

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:

#	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

grfr command set

The **grfr** command has display commands that return useful information about Frame Relay links.

Display commands

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 **dbb**, display board status

Configuration and debug commands

- ```

-c cel, enable link, enable link on slot 3, port 1: # grfr -c cel -s 3 -l 1
-c cdl, disable link, disable link on slot 3, port 0: # grfr -c cdl -s 3 -l 0
-c cep, enable PVC, requires PVC to be specified: # grfr -c cep -i 888 -s 3 -l 0
-c cdp, disable PVC, requires PVC to be specified: # grfr -c cdp -i 888 -s 3 -l 0
-c ccp, configure PVC, requires the configuration file and the PVC to be specified:
 # grfr -c ccp -f /etc/grfr.conf -i dlci
-c crp, remove PVC, requires the PVC to be specified: # grfr -c crp -i dlci
-c ddl, display debug level: # grfr -c ddl
-c csd, set debug level, requires -d option to specify new level 0–4: # grfr -c csd -d 3

```

Refer to the *GRF Reference Guide* for more information about the **grfr** command.

## Configuring Point-to-Point Protocol

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.

### 1. Specify interface names in `grifconfig.conf`

These entries in `grifconfig.conf` identify the two logical interfaces, upper and lower, on the HSSI card in slot 2.

| # name | address    | netmask | broad_dest | argument |
|--------|------------|---------|------------|----------|
| #      |            |         |            |          |
| gs020  | 192.0.2.1  | -       | -          | 1500     |
| gs021  | 192.0.99.1 | -       | -          | 1500     |

**Note:** Interface names are case sensitive. Always use lower case letters when defining interface names.

### 2. Specify framing protocol

```
super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
media-type = hssi
debug-level = 0
hssi-frame-protocol = ppp
```

### 3. Set internal clocks and the CRC

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

super> list ports 0
port_num = 0
cisco-hdlc = { off on 10 3 }
fddi = { dual 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
dis+

super> cd hssi
source-clock = 1
CRC-type = 16-bit
```

In this example, null-modem cables are used for both connectors.

#### **4. Set PPP parameters in */etc/grppp.conf***

Open the UNIX shell:

```
super> sh
```

Use a UNIX editor and edit the `grppp.conf` file. The template for the `grppp.conf` configuration file is provided at the end of the parameter descriptions.

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.

To make immediate, temporary changes, use the **grppp** command. Refer to the **grppp** man page for information.

**Note:** Settings done with **grppp** are lost when the HSSI card is reset.

## **Required interface parameter**

To set up a PPP interface, specify the interface with the `interface_name` in `gs0yz` format:  
`interface interface_name`

## **Optional negotiation automata parameters**

```
enable negotiation trace
 - writes PPP negotiation traces into /var/log/gr.console file.

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 3000 milliseconds).
```

## **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, 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)

set lcp mru = INTEGER
 - defines maximum packet size (default is 1500 octets)
```

## 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,
 a value specified in 1/100 seconds begins the exchange of statistics between
 endpoints (default is 0)
```

## IPCP parameter

```
enable ipcp - required for IP traffic across PPP link (default is disabled)
```

## PPP status reporting

The **grppp** command displays status for PPP objects and configuration values. Use the **maint** commands to look at packet statistics.

## grppp status commands

Use the appropriate **grppp** command to display the desired information. Commands are entered in lower case, short forms of words can be used. Refer to the **grppp** man page for more information. These are the **grppp** status commands:

```
grppp show configuration
grppp show negotiation trace status
grppp show maximum configuration request count
grppp show maximum failure count
grppp show maximum terminate count
grppp show restart timer interval
grppp show lcp keepalive interval
grppp show lcp keepalive packet threshold
grppp show lcp mru
grppp show lcp status
grppp show lqr timer interval
grppp show lqr status
grppp show ipcp status
```

You must declare a specific logical interface with an **interface gx0yz** statement, then the **gx0yz** prompt is generated. You can now enter the desired **show** commands against this interface. Use another **interface** statement to change interfaces.

Here are two examples:

```
grppp
```

```
> interface gs090
gs090> 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 milliseconds, disabled
 Keepalive packet threshold: 5
LQR Configuration:
 LQR is disabled
 Timer interval: 0 milliseconds
IPCP Configuration:
 enable IPCP
OSINLCP Configuration:
 disable OSINLCP

gs090> show lcp status
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 milliseconds, disabled
 Keepalive packet threshold: 5
```

## Looking at a PPP configuration

Here is the output from a **netstat** command looking at the HSSI interfaces:

```
netstat -in | grep gs
gs0b0 4352 <link9> 0 0 0 0 0
gs0b0 4352 202.5.1 202.5.1.129 0 0 0 0 0
gs0b0 4352 47.0000.3355.5555.5555.5500 0 0 0 0 0
gs0b80 4352 <link10> 0 0 0 0 0
gs0b80 4352 192.168.24 192.168.24.129 0 0 0 0 0
gs0b80 4352 47.0000.3355.5555.5555.5500 0 0 0 0 0
```

Here is the output from a **netstat** command looking at the SONET interfaces:

```
netstat -in | grep go
go090 1500 <link13> 0 0 0 0 0
go090 1500 208.1.1 208.1.1.129 0 0 0 0 0
```

Here is the output from a **grppp show config** command:

```
grppp
```

```
>interface gs090
gs090> 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 milliseconds,
 disabled Keepalive packet threshold: 5
LQR Configuration:
LQR is disabled
Timer interval: 0 milliseconds
IPCP Configuration:
 enable IPCP
OSINLCP Configuration:
 disable OSINLCP
```

## grppp.conf file

Figure 6-1 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
override 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.con-
sole sole
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 6-1. Template for grppp.conf file*



## Monitoring HSSI media cards

The **maint** commands operate on the IP switch control board and require the GR> prompt. Execute the **grmb** 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.

### Preparing to use maint

First, switch to the GR 66> prompt. Enter:

```
grmb
```

The new prompt appears:

```
GR 66>
```

Then change the prompt to the HSSI media card you are working with. Enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 02
```

```
GR 2>
```

To leave the **maint** prompt, enter quit.

### Display maint commands

To view the list of HSSI card **maint** commands, enter:

```
GR 2> 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 Combust 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.
[RX] 81: Display PVC table
```

## Read S/W and H/W revisions

Use **maint 2** to read the revision levels of the operating software and media card hardware.

```
GR 2> 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 Combust 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 2> maint 3
GR 2> [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 2> maint 4
GR 2> [RX]
[RX]
[RX] Media Statistics
[RX]
[RX] Port Bytes Packets Errors Discards
[RX] -----
[RX] 0 0000000000050979765 000000000000432042 0000000000
0000000000
[RX] 1 0000000000066766723 000000000000144902 0000000000
0000000000
[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 0000000000
[RX] 1 0000000000054470578 000000000000492390 0000000000
[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 2> maint 5
```

```
GR 2> [RX]
[RX]
[RX] Switch Statistics
[RX] input:
[RX] Bytes Packets Errors
[RX] -----
[RX] 00000000001592268 000000000000016581 0000000000
[RX]
[RX] output:
[RX] Bytes Packets Errors
[RX] -----
[RX] 000000000002918212 000000000000016581 0000000000
[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 2> maint 7
GR 2> [RX]
[RX] All Media Statistics Cleared.
[RX] All Switch Statistics Cleared.
[RX] All Combust Statistics Cleared.
```

## Display PVC status

**maint 8** provides status and information about each configured PVC:

```
GR 2> maint 8
GR 2> [RX]
[RX] Frame-Relay PVC Status:
[RX] ('*' = address obtained via inverse arp)
[RX] ('+' = Enabled for ISIS)
[RX]
[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
```

## Display IPC statistics

**maint 11** returns a set of IPC statistics:

```
GR 2> maint 11
GR 2> [RX]
[RX] IPC Stats:
[RX] =====
[RX] RX IPC Message Received: 10346629
[RX] RX IPC Message Sent: 391
[RX] RX Grid Packets Received: 0
[RX] RX Overruns: 0
[RX] RX Local Messages: 0
[RX] TX IPC Message Received: 426
[RX] TX IPC Message Sent: 10346604
[RX] TX Grid Packets Received: 373
[RX] TX Overruns: 0
[RX] TX Local Messages: 0
```

## Display next hop data

Use **maint 45** to view next hop data for card interfaces:

```
GR 2> maint 45
GR 2> [RX] Location is: 0
[RX] Add: 51 Delete: 12 noNH: 0
[RX] 0: 0.0.0.0 (NULL) 3:00 0:f8 UNREACH
[RX] 1: 0.0.0.0 (NULL) 3:00 0:f8 DROP
[RX] 2: 0.0.0.0 (NULL) 3:00 0:f8 BCAST
[RX] 3: 0.0.0.0 (NULL) 3:00 0:f8 RMS
[RX] 4: 206.146.160.1 (NULL) 3:00 0:fc RMS
[RX] 5: 203.1.3.1 (10) 3:00 0:00 ATMP
[RX] 6: 0.0.0.0 (NULL) 3:00 0:fc RMS
[RX] 7: 221.1.1.1 (12) 3:00 0:00 ATMP
[RX] 8: 221.1.1.2 (11) 3:00 0:00 ATMP
[RX] 9: 0.0.0.0 (6) 3:00 0:00 BCAST
[RX] 10: 0.0.0.0 (6) 3:00 0:00 LOCAL
[RX] 11: 0.0.0.0 (9) 3:00 2:00 BCAST
[RX] 12: 0.0.0.0 (9) 3:00 2:00 LOCAL
[RX] 13: 0.0.0.0 (7) 3:00 0:81 BCAST
[RX] 14: 0.0.0.0 (7) 3:00 0:81 LOCAL
[RX] 15: 0.0.0.0 (6) 3:00 0:00 FWD
```

```
[RX] 16: 0.0.0.0 (19) 3:00 3:04 FWD
[RX] 17: 0.0.0.0 (11) 3:00 3:05 FWD
[RX] 18: 0.0.0.0 (27) 3:00 2:80 BCAST
[RX] 19: 0.0.0.0 (27) 3:00 2:80 LOCAL
[RX] 20: 0.0.0.0 (9) 3:00 2:00 FWD
[RX] 21: 0.0.0.0 (10) 3:00 3:03 BCAST
[RX] 22: 0.0.0.0 (10) 3:00 3:03 LOCAL
[RX] 23: 0.0.0.0 (11) 3:00 3:05 BCAST
[RX] 24: 0.0.0.0 (19) 3:00 3:04 BCAST
[RX] 25: 0.0.0.0 (19) 3:00 3:04 LOCAL
[RX] 26: 0.0.0.0 (11) 3:00 3:05 LOCAL
[RX] 27: 0.0.0.0 (27) 3:00 2:80 FWD
[RX] 28: 221.1.2.1 (9) 3:00 0:00 ATMP
[RX] 29: 221.1.2.2 (8) 3:00 0:00 ATMP
[RX] 30: 221.1.2.3 (7) 3:00 0:00 ATMP
[RX] 31: 221.1.2.4 (6) 3:00 0:00 ATMP
[RX] 32: 221.1.2.5 (5) 3:00 0:00 ATMP
[RX] 33: 221.1.2.6 (4) 3:00 0:00 ATMP
[RX] 34: 221.1.2.7 (3) 3:00 0:00 ATMP
[RX] 35: 221.1.2.8 (2) 3:00 0:00 ATMP
[RX] 36: 221.1.2.9 (1) 3:00 0:00 ATMP
[RX] 37: 221.1.2.10 (0) 3:00 0:00 ATMP
[RX] 38: 10.20.3.133 (9) 3:00 2:00 FWD
[RX] Location is: 1
[RX] Add: 0 Delete: 0 noNH: 0
```

## List of filters

**maint 50** returns the list of filters by filter ID:

```
GR 2> maint 50
GR 2> 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 2> 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.

## Display ATMP home network table

The **maint 70** and **maint 73** ATMP commands are documented in the ATMP Configuration chapter.







# Ethernet Configuration

# 7

Chapter 7 provides information needed to configure Ethernet 10/100Base-T media cards.

This media is also known as “Fast Ethernet”.

Two types of Ethernet 10/100Base-T card are available, one with eight ports and one with four ports. Configuration is the same for each type. Each independent physical interface is capable of connecting at speeds of 10 or 100 megabits per second.

This chapter contains:

|                                                 |      |
|-------------------------------------------------|------|
| Ethernet implementation .....                   | 7-2  |
| Physical and logical interfaces .....           | 7-5  |
| Configuration file and profile overview .....   | 7-6  |
| Identify logical interfaces .....               | 7-7  |
| Set up Ethernet media card – Card profile ..... | 7-8  |
| Set up Ethernet media card – Load profile ..... | 7-13 |
| Set up Ethernet media card – Dump profile ..... | 7-14 |
| Monitoring Ethernet media cards .....           | 7-16 |
| Media statistics .....                          | 7-18 |

## 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                                                   |
| RFC 1643 | Definitions of managed objects for Ethernet-like interface types     |

### 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.

Setting a congestion and discard threshold is described in the “Selective packet discard” section later in this chapter.

### 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 netmask 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
```

## 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 supports LLC/SNAP (IEEE 802.3) frames for IP and ARP.

## ARP support

For Ethernet, the `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 Ethernet MTU size is 1500 bytes.

## CSMA/CD (flow control)

Collision sensing capability is based upon the standard MAC-level CSMA/CD algorithm.

Interframe gap ? can we set this or detect it ?

## 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 Mbps or 100 Mbps 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.

## 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.

## Transfer rates

Operating as 10 Base-T half-duplex, an interface transfers 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 transfers 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).

## 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 for information about class filters.

## Physical and logical interfaces

### 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 GRF Fast Ethernet media card to interoperate with 10Base-T and 100Base-T devices. These capabilities 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 `grifconfig.conf` file where it is assigned an IP address and netmask. A logical interface is uniquely identified by its Ethernet interface name.

### Interface name

The generic form of a Ethernet interface name is: `ge0yz`  
where:

- the “ge” indicates an Ethernet (10/100Base-T) interface
- the GRF chassis number is always “0”
- “y” is a hex digit (0 through 3) for the chassis slot number
- the “z” logical interface number ranges from 0 through 7 (in hex) to identify each individual logical interface configurable per Ethernet card

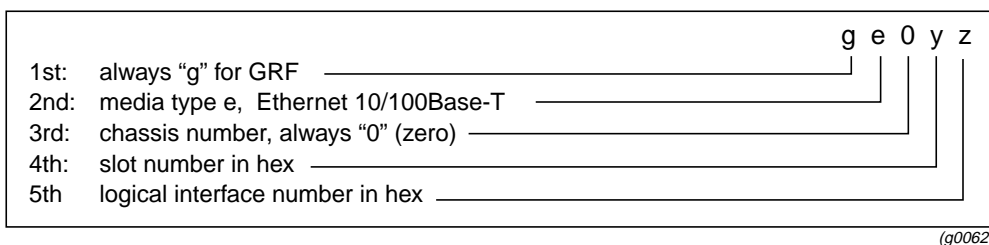


Figure 7-1. Components in Ethernet interface name

The interface name is used in the `/etc/grifconfig.conf` file to specify an IP interface. Here is an example specifying the interface name for interface 7 on the Ethernet card in slot 3:

```
#name address netmask broad_dest arguments
ge037 xxx.xxx.xxx.xxx 255.255.255.0 - mtu 1500
```

**Note:** Interface names are case sensitive. Always use lower case letters when defining interface names.

## Configuration file and profile overview

These are the steps to configure Ethernet interfaces:

### 1. *Identify each logical interface*

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

### 2. *Specify Ethernet card parameters in the Card profile:*

These are the configurable items in a Card profile for an Ethernet media card:

- specify verbose option for messages from Ethernet card
- OPTIONAL: specify ICMP throttling settings
- configure interface mode: autonegotiate, 10 or 100 Base-T, full or half duplex
- OPTIONAL: specify selective packet discard %
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

### 3. *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 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. *Dump profile*

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 zero (0) which actually stores two dumps per day (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 (per interface), make the change in the Card profile, in the `dump` field.

## 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:

```
greset <slot_number>
```

## Identify logical interfaces

Configure each logical interface in the `grifconfig.conf` file by assigning it 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:

| #      | Internet    |                 | broadcast/  |           |
|--------|-------------|-----------------|-------------|-----------|
| # name | address     | netmask         | destination | 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 |           |

## Set up Ethernet media card – Card profile

Set operating parameters for the Ethernet media card physical interfaces in the Card profile.

- specify verbose option for messages from Ethernet card
- OPTIONAL: specify ICMP throttling settings
- configure interface mode: autonegotiate, 10 or 100 Base-T, full or half duplex
- 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, 1 provides a higher number of messages, you can specify up to 9:

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

super> list
card-num* = 2
media-type = ethernet-vl
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 inter-
nal-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> set ether = 8
super write
CARD2/ written
```

### 2. Specify ICMP throttling

Set `icmp-throttling` fields.

You can specify ICMP throttling settings for this Ethernet 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 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]
```

Default values are shown:



```
super> list ic
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/2 written
```

### 3. Set the negotiation or transfer rate

Set the negotiation or transfer rate in the ports / ether field. By default, the setting for each interface is autonegotiate.

```
super> cd ..
card-num* = 2
media-type = ethernet-vl
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 inter-
nal-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 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
disab+

super> list ether
if-config = autonegotiate
```

At this level, 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
```

```
super> set if-config = 100-half
super> write
CARD2/ written
```

#### **4. *Specify selective packet discard (SPD)***

Specify a SPD percentage for this Ethernet card in the `spd-tx-thresh` field. This field is contained in the `config` field of the top-level Card profile.

Given an Ethernet card installed in slot 2, this example shows where to specify the `spd-tx-thresh` setting in the Card 2 profile:

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

super> list
card-num* = 2
media-type = ethernet-vl
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 inter-
nal-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 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/2 written
```

A discussion of how to decide an appropriate SPD percentage is provided in the “Selective packet discard” section later in this chapter.

#### **5. *Specify 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 2
card/2 read

super> list
card-num* = 2
media-type = ethernet-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 }

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/port-
card/test_execut_for_ethernet
super> write
CARD/2 written
```

## 6. 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 2
card/2 read

super> list
card-num* = 2
media-type = ethernet-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 }

super> list dump
config = 0
hw-table = < >
```

## Ethernet Configuration

### *Set up Ethernet media card – Card profile*

---

```
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
CARD2/ written
```

## ***Set up Ethernet media card – 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** 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/xlload.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/port-
cards/hssi_+
dev1 = { /usr/libexec/portcards/dev1_rx.run /usr/libexec/port-
cards/dev1_+
atm-oc3-v2 = { /usr/libexec/portcards/atmq_rx.run
/usr/libexec/portcards+
fddi-v2 = { /usr/libexec/portcards/fddiq-0.run /usr/libexec/port-
cards/fd+
atm-oc12-v1 = { /usr/libexec/portcards/atm-l2.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/port-
cards/+
```

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. 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> 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 = < >

super> set enable-boot-seq = on
super> write
LOAD written
```

## ***Set up Ethernet media card – 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 zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0}{fddi 20 /var/portcards/grdump 2} {rmb
20 /v+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi
core +
config-spontaneous = off
keep-count = 0
```

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 = { hippo 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 = /mae/peter.dumps/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.

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 mem-
ory" 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
```

## ***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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 Ethernet 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 GateD 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.

## Example

Given an Ethernet card installed in slot 2, this example shows where to specify the `spd-tx-thresh=` setting in the Card 2 profile:

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

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/2 written

super> grreset 2
```

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
```

## Monitoring Ethernet media cards

The **maint** commands operate on the IP switch 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.

### Preparing to use maint

First, switch to the **maint** GR 66> prompt. Enter:

```
grrmb
```

The new prompt appears:

```
GR 66>
```

Then, change the prompt to the Ethernet media card you are working with. If the card is in slot 2, enter:

```
GR 66> port 2
```

This message is returned along with the changed prompt:

```
Current port card is 02
```

```
GR 02>
```

To leave the **maint** prompt, enter **quit**.

### Display maint commands

Use **maint 1** to view the list of receive-side Ethernet **maint** commands:

```
GR 02> maint 1
[RX]
[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 Combust 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
```

Use **maint 101** to view the list of transmit-side Ethernet **maint** commands:

```
GR 02> maint 101
[TX]
[TX] 101: Display this screen of Options
[TX] 106: Display Combust 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 02> 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 port side is reported on first:

```
GR 02> maint 4
[RX]
[RX] Media Statistics
[RX] input:
[RX] Port Bytes Packets Errors Discards
[RX] -----
[RX] 0 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 1 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 2 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 3 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 4 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 5 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 6 000000000000000000 000000000000000000 0000000000 0000000000
[RX] 7 000000000000000000 000000000000000000 0000000000 0000000000
[RX]
[RX] Port 0:
[RX] Unsupported type: 0
[RX] CRC errors: 32896
[RX] Runt errors: 32896
[RX] Oversize Frames: 128
[RX] Alignment errors: 32896
[RX] Collision errors: 128
[RX] Out of buffers: 0
[RX] .
[RX] .
[RX] .
[RX] Port 3:
[RX] Unsupported type: 0
[RX] CRC errors: 1
[RX] Runt errors: 1
[RX] Oversize Frames: 0
[RX] Dribble errors: 3
[RX] Collision errors: 0
[RX] Out of buffers: 0
[RX] Port 4:
[RX] Unsupported type: 0
[RX] CRC errors: 35441
[RX] Runt errors: 36140
[RX] Oversize Frames: 162
[RX] Dribble errors: 40637
[RX] Collision errors: 0
[RX] Out of buffers: 0
[RX] Port 5:
[RX] Unsupported type: 0
[RX] CRC errors: 8
[RX] Runt errors: 2
[RX] Oversize Frames: 0
[RX] Dribble errors: 11
[RX] Collision errors: 0
[RX] Out of buffers: 0
```

```
[RX] Port 6:
[RX] Unsupported type: 0
[RX] CRC errors: 154331584
[RX] Runt errors: 154331584
[RX] Oversize Frames: 600512
[RX] Dribble errors: 154331584
[RX] Collision errors: 154331584
[RX] Out of buffers: 0
[RX] Port 7:
[RX] Unsupported type: 0
[RX] CRC errors: 0
[RX] Runt errors: 0
[RX] Oversize Frames: 0
[RX] Dribble errors: 0
[RX] Collision 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 0000000000
[RX] 1 000000000000000000 000000000000000000 0000000000
[RX] 2 000000000000000000 000000000000000000 0000000000
[RX] 3 000000000000000000 000000000000000000 0000000000
[RX] 4 000000000000000000 000000000000000000 0000000000
[RX] 5 000000000000000000 000000000000000000 0000000000
[RX] 6 000000000000000000 000000000000000000 0000000000
[RX] 7 000000000000000000 000000000000000000 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 Collision Errors: 0
[RX] Port 1:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
[RX] Port 2:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
[RX] Port 3:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
[RX] Port 4:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
```

```
[RX] Port 5:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
[RX] Port 6:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
[RX] Port 7:
[RX] Odd Length TX Packets: 0
[RX] TX Dropped Fifo Full: 0
[RX] TX Dropped Line Down: 0
[RX] TX Collision Errors: 0
```

When you specify the interface number (0-7), only the statistics for that interface are returned:

```
GR 02> maint 4 0
[RX]
[RX] Media Statistics
[RX] input:
[RX] Port Bytes Packets Errors Discards
[RX] -----
[RX] 0 00000000000000000000 000000000000000000 0000000000
0000000000
[RX]
[RX] Port 0:
[RX] Unsupported type: 0
[RX] CRC errors: 32896
[RX] Runt errors: 32896
[RX] Oversize Frames: 128
[RX] Alignment errors: 32896
[RX] Collision errors: 128
[RX] Out of buffers: 0
[RX]
[RX] output:
[RX] Port Bytes Packets Discards
[RX] -----
[RX] 0 00000000000000000000 000000000000000000 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 02> maint 5
[RX] Switch Statistics
[RX] input:
[RX] Bytes Packets Errors
[RX] -----
[RX] 00000000000000000000 000000000000000000 000000000
[RX]
[RX] output:
[RX] Bytes Packets Errors Overruns
[RX] -----
[RX] 00000000000000000000 00000000000000000000 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 Combust statistics

The communications bus provides the 80-Mbs inter-card and IP switch control board communications path. Use **maint 6** to view Combust statistics:

```
GR 02> maint 6
[RX]
[RX] Combust Status:
[RX] Last interrupt status: 0x50503055
[RX] Combust Statistics:
[RX] Message ready interrupts: 10256
[RX] Truncated input messages: 0
[RX] Grit messages for TX-CPU: 401
[RX] Ip messages Rcvd (non-bypass): 0
[RX] Raw messages: 0
[RX] ISO messages: 0
[RX] Grid messages: 9855
[RX] Grid echo requests: 9629
[RX] Port available messages: 0
[RX] Segmented Packets: 0
[RX] Segments Sent: 0
[RX] Combust Errors:
[RX] Bus in timeouts: 1 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: 2
```



## Clear status info

Use **maint 7** to clear the current collected statistics:

```
GR 02> maint 7
[RX]
[RX] All Media Statistics Cleared.
[RX] All Switch Statistics Cleared.
[RX] All Combust Statistics Cleared.
```

## Display ARP tables

**maint 8** displays the ARP table for one interface or, if no interface is specified, for all interfaces:

```
GR 02> maint 8
[TX]
[TX] Arp Table for Interface 0:
[TX] IP Address Mac Address Status TTL
[TX] ===== ===== =====
[TX]
[TX] Arp Table for Interface 1:
[TX] IP Address Mac Address Status TTL
[TX] ===== ===== =====
[TX]
[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]
[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:** Chapter 8, *IP Packet Filtering*, has information about using the filtering **maint** command set.

**maint 50** returns the list of filters by filter ID:

```
GR 02> maint 50
GR 02> 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 02> maint 58
GR 02> [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
```

# SONET OC-3c Configuration

## 8

Chapter 8 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 (PPP), and HDLC.

The SONET card provides one redundant connection using two physical interfaces capable of connecting at speeds of 155 megabits per second.

This chapter contains:

|                                               |      |
|-----------------------------------------------|------|
| SONET implementation .....                    | 8-2  |
| Logical and physical interfaces .....         | 8-5  |
| Card connection options .....                 | 8-7  |
| Configuration file and profile overview ..... | 8-9  |
| Configure logical interfaces .....            | 8-12 |
| Set up SONET media card – Card profile .....  | 8-12 |
| Set up SONET media card – Load profile .....  | 8-17 |
| Set up SONET media card – Dump profile .....  | 8-18 |
| Selective packet discard .....                | 8-20 |
| Configuring the HDLC protocol .....           | 8-22 |
| Configuring the Frame Relay protocol .....    | 8-24 |
| Configuring Point-to-Point Protocol .....     | 8-30 |
| grppp.conf file .....                         | 8-33 |
| Monitoring SONET OC-3c media cards .....      | 8-34 |

## SONET implementation

The GRF SONET OC-3c supports the APS 1+1 Architecture automatic protection switching, 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”

## 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 SONET 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*

## HDLC

Cisco HDLC is the name given to Cisco’s default protocol over SONET 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 `/etc/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 PPP MTU is 1500, it can be specified differently 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 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 *Introduction to IS-IS* chapter for more information.

Here is an example of SONET IP and IS-IS entries in `/etc/grifconfig.conf`:

```
#name address netmask broad_dest arguments
go030 xxx.xxx.xxx.xxx 255.255.255.0 - mtu 4352
#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.

## Selective packet discard

Selective packet discard can be enabled on the SONET 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.

Setting a congestion and discard threshold is described in the “Selective packet discard” section later in this chapter.

## 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 for information about constructing class filters.

## Logical and physical interfaces

### Physical interfaces

The SONET OC-3c media card provides two bi-directional physical interfaces.

#### APS 1+1 architecture

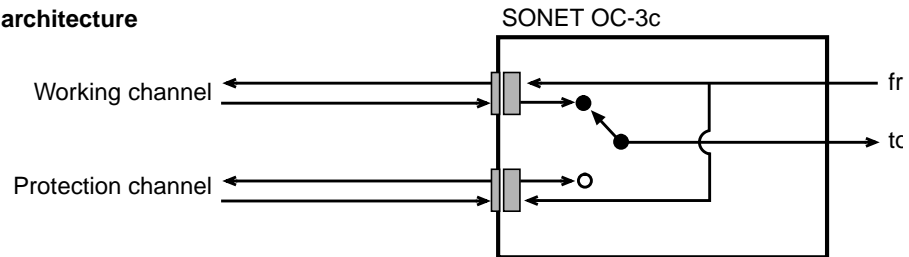


Figure 8-1. SONET support for redundant links

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

The single 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 SONET OC-3c interface name.

### Interface name

The generic form of a SONET OC-3c interface name is: `go0y0`  
where:

- the “go” indicates an SONET OC-3c interface
- the GRF chassis number is always “0”
- “y” is a hex digit (0 through 3) for the chassis slot number
- the “z” logical interface number is always 0 regardless of which interface is active

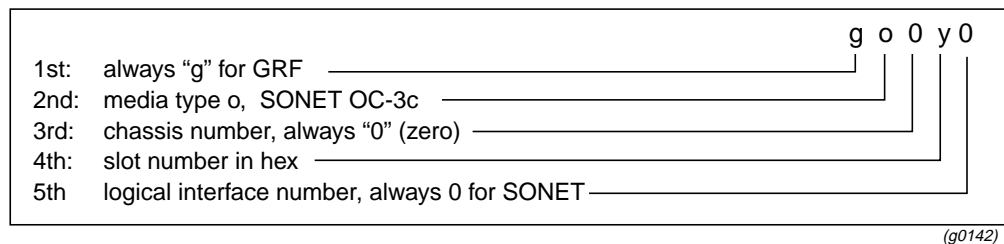


Figure 8-2. Components in SONET OC-3c interface name

### *Identify logical interfaces in grifconfig.conf*

Configure each logical interface in the `grifconfig.conf` file by assigning it an IP address, a GRF interface name, and if required, a netmask and destination/broadcast address. The template for the `grifconfig.conf` file is in the *GRF Reference Guide*.

Here is the format for `grifconfig.conf` file entries:

| #     | name | address    | netmask       | broad_dest   | arguments |
|-------|------|------------|---------------|--------------|-----------|
| go030 |      | 192.0.2.1  | 255.255.255.0 | 192.0.2.255  | mtu xxxx  |
| go030 |      | 192.0.99.1 | 255.255.255.0 | 192.0.99.255 |           |

**Note:** Interface names are case sensitive. Always use lower case letters when defining interface names.

A SONET card that is to run the PPP or HDLC protocol requires one or two entries into the `grifconfig.conf` file since these protocols support one logical interface on each of the physical interfaces.

A SONET card that is to run Frame Relay will have as many as 256 entries since Frame Relay supports 128 logical interfaces per physical interface.

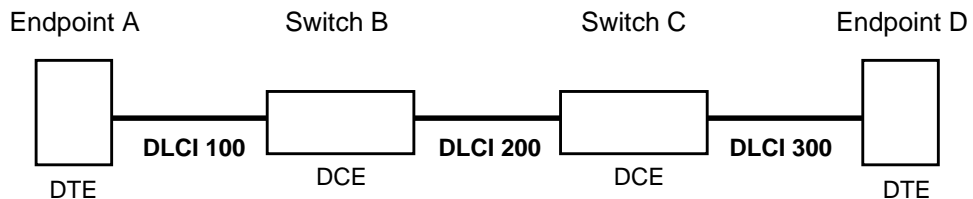


## Card connection options

### Link type

You can specify a Frame Relay link to be UNI-DTE, UNI-DCE or NNI.

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. An NNI link is the link between two switches inside the network. Both switching and routing can be performed on UNI-DCE and NNI links. Only routing can be performed on UNI-DTE links.



In the above example, 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.

### *DCE connection*

The SONET card can connect to a CSU/DSU or to a DCE switch via copper twisted-pair cable.

### *DTE connection*

A null-modem cable is required to connect the SONET card directly to another DTE device.

## Null modem cabling (SONET crossover cables)

When a null-modem cable is used, each DTE must be set to enable internal clock generation. By default, the clock is set to internal.

### *Where to set*

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 `lock =` parameter at the Card profile, in the `ports / sonet` field. The procedure is included in the Card profile section.

Here is the path where you check the clock setting:

```
super> read card 8
CARD/8 read
super> list card 8

card-num* = 8
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-oscillator 0 20+
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 = {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+

```

Now list the specific SONET port you want to set, 0 or 1:

```

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 32-bit }
ether = { autonegotiate }
hippi = { 1 32 no-mode 999999 4 increment 5 300 10 10 03:00:0f:c0
disable+

```

Now list the SONET field for that interface (a typing shortcut is used):

```

super> list sonet
path_trace_message = ""
section_trace_message = ""
aps-command = 1
mode = sonet
clock = internal-oscillator
aps-status = 0
payload-identifier = 207

```

The clock = internal-oscillator default is shown, you do not need to change the setting. If the setting is clock = recovered-clock, here is how you change it:

```

super> set clock = int
super> write
CARD/8 written

```



## Configuration file and profile overview

These are the steps to configure the SONET interface and PPP protocol:

### *1. Identify each logical interface*

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

### *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
- specify SONET payload identifier
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard %
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

### *3. Configure the framing protocol*

**Cisco HDLC** - after steps 1 and 2, set the HDLC parameters specified in the Card profile

**Frame Relay** - after steps 1 and 2, set Frame Relay and PVC parameters in the `grfr.conf` configuration file

**Point-to-Point Protocol** - after steps 1 and 2, set PPP parameters in the `grppp.conf` configuration file

### *4. 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.

If you want to change the run-time code in one SONET card, make the change in the Card profile, in the `load` field.

Here is the path, defaults are shown:

```
super> read load
LOAD read
super> list .
```

```
hippi = { " N/A on 0 1 < {1 /usr/libexec/portcards/xlload.run
N/A} {2 +
rmb = { /usr/libexec/portcards/rm.run N/A off 0 1 < > }
hssi = {/usr/libexec/portcards/hssi_rx.run /usr/libexec/port-
cards/hssi_+
dev1 = {/usr/libexec/portcards/dev1_rx.run /usr/libexec/port-
cards/dev1_+
atm-oc3-v2 = {/usr/libexec/portcards/atmq_rx.run
/usr/libexec/portcard+
fddi-v2 = {/usr/libexec/portcards/fddiq-0.run /usr/libexec/port-
cards/fd+
ethernet-v1 = {/usr/libexec/portcards/ether_rx.run
/usr/libexec/portcar+
sonet-v1 = {/usr/libexec/portcards/sonet_rx.run /usr/libexec/port-
cards/+

super> list . 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 = 0
boot-seq-table = < { 1 " " " } >
```

## 5. 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 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 (per interface), make the change in the Card profile, in the `dump` field.

Here is the path, defaults are shown:

```
super> read dump
DUMP read
super> list .
hw-table = <{hippi 20 var 0} {fddi 20 /var/portcards/grdump 2}{rmb
20 /v+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi
core +
config-spontaneous = 0
boot = 0
keep-count = 0
```

The `hw-table` field describes where the dumps are stored. Here is the path (notice the shortcuts used):

```
super> list . hw sonet-v1
media = sonet-v1
config = 20
path = /var/portcards/grdump
vector-index = 2
```

The segment-table fields in the dump-vector-table describe the areas in core memory that get dumped.

Here is the path, you need to **cd ..** back up to the first 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 mem-
ory" 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+

super> list 11
index = 7
hw-type = sonet-v1
description = "SONET default dump vectors"
segment-table =<{1 "sonet core memory" 2097152 1048576}{2 "hssi
core me+

super> list s
1 = { 1 "sonet-v1 core memory" 2097152 1048576 }
2 = { 2 "sonet-v1 core memory 2" 16777216 1048576 }
```

## 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:

```
greset <slot_number>
```

## Configure logical interfaces

Configure each SONET logical interface in the `/etc/grifconfig.conf` file by assigning it 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
#
go030 192.0.2.1 255.255.255.0 192.0.2.255
```

## Set up SONET media card – Card profile

Set specific SONET card configuration parameters at the Card profile. The fields to set are:

- framing protocol: `cisco-hdlc`, `ppp`, `frame-relay` (default is Frame Relay)
- APS: specify APS command, 1–6
- mode: specify card mode, SONET or SDH
- clock: specify internal oscillator or recovered clock
- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard %
- OPTIONAL: set card-specific load variables
- OPTIONAL: set card-specific dump variables

Fields you may want to set are in bold, default values are shown here.

### 1. Specify framing protocol

At the top level, you can set the framing protocol. Values are:

- `Cisco-HDLC`
- `PPP`
- `Frame-Relay`

Media card type, `sonet-v1`, is automatically read into the read-only `media-type` field. Other values shown are defaults. The `sonet-frame-protocol` field contains `ppp` by default. You can also specify Frame-Relay or Cisco-HDLC.

```
super> read card 8
CARD/8 read
super> list card 8
card-num* = 8
media-type = sonet-v1
debug-level = 0
hssi-frame-protocol = Frame-Relay
sonet-frame-protocol = PPP
ether-verbose = 0
ports = < { 0{ single off}{" " 1 sonet internal-oscillator 0}{0
16-bit+
```



```
load = { 0 < { fddi 0 "" 0 "" off 0 1 < { 1 fddi "" "" } > } > 1 0
0 }
dump = { 0 empty 0 }
config = { 0 1 1 0 0 0 }
icmp-throttling = { 10 10 2147483647 10 10 10 }
```

To change the framing protocol to Cisco HDLC and save your change:

```
super> set sonet-frame-protocol = cisco-hdlc
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. Specify ICMP throttling

You can specify ICMP throttling settings for this SONET 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 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]
```

Default values are shown:

```
super> list ic
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
```

## 3. Specify selective packet discard (SPD)

Specify a SPD percentage for this SONET card in the `spd-tx-thresh` field. This field is contained in the `config` field of the top-level Card profile.

Given a SONET card installed in slot 2, this example shows where to specify the `spd-tx-thresh=` setting in the Card 2 profile:

```
super> read card 8
CARD/2 read

super> list
card-num* = 6
```

## SONET OC-3c Configuration

### Set up SONET media card – Card profile

---

```
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 inter-
nal-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 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 decide an appropriate SPD percentage is provided in the “Selective packet discard” section later in this chapter.

#### 4. Specify SONET APS, mode, clock, and payload settings

The fields are located in the `ports 0` or `ports 1` field of the top-level 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 = PPP
ether-verbose = 0
ports = < { 0 {off on 10 3}{single off}}{" " " 1 sonet inter-
nal-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 = { 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 that correspond to the set of six commands:

- 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 read-only, it displays the actual state of the active interface.

Payload identifier can be set between 0 and 255, the default is 207.

```
super> set aps-command = 2
super> set mode = sdh
super> set clock = recovered-clock
super> payload-identifier = 200
super> write
CARD/5 written
```

## ***5. Specify 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 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/port-
card/test_executable_for_sonet
super> write
CARD/8 written
```

## *6. 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 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:

```
greset <slot_number>
```

## Set up SONET media card – 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/xlload.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/port-
cards/hssi_+
dev1 = { /usr/libexec/portcards/dev1_rx.run /usr/libexec/port-
cards/dev1_+
atm-oc3-v2 = { /usr/libexec/portcards/atmq_rx.run
/usr/libexec/portcards+
fddi-v2 = { /usr/libexec/portcards/fddiq-0.run /usr/libexec/port-
cards/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/port-
cards/+
```

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> 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 = < >
super> set enable-boot-seq = on
super> write
LOAD written
```

## Set up SNET media card – 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 zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, defaults are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0}{fddi 20 /var/portcards/grdump 2} {rmb
20 /v+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi
core +
config-spontaneous = off
keep-count = 0
```

The `hw-table` field has settings for when dumps are taken and where dumps are stored. Here is the path to examine the SNET settings:

```
super> list hw-table
hippi = { hippy 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 mem-
ory" 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 s
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 ..
```

## **SONET OC-3c Configuration**

*Set up SONET media card – Dump profile*

---

```
super> list s 2
index = 2
description = "SOMET shared SRAM memory"
start = 131072
length = 32768
```

## Selective packet discard

Selective packet discard can be enabled on the SONET 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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 SONET 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 GateD 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.

## Example

Given a SONET card installed in slot 2, this example shows where to specify the `spd-tx-thresh=` setting in the Card 2 profile:

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

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/2 written
super> grreset 2
```

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
```

## 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: 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
```

# Configuring the HDLC protocol

## 1. Specify interface names in grifconfig.conf

Identify the two logical interfaces on the SONET card. Here are sample entries in grifconfig.conf for two logical interfaces on the SONET card in slot 2:

| #     | name | address    | netmask | broad_dest | argument |
|-------|------|------------|---------|------------|----------|
| go020 |      | 192.0.2.1  |         |            |          |
| go021 |      | 192.0.99.1 |         |            |          |

**Note:** Interface names are case sensitive. Always use lower case letters when defining interface names.

## 2. Specify framing protocol

Set the framing protocol in the Card profile.

```
super> read card 8
CARD/8 read
super> list
card-num* = 8
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 inter-
nal-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> set sonet-frame-protocol = cisco-hdlc
super> write
CARD/8 written
```

## 3. Set Cisco HDLC settings

- check cisco-hdlc default settings

```
super> list ports
0 = { 0 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
1 = { 1 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
2 = { 2 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
3 = { 3 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
4 = { 4 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
5 = { 5 { off on 10 3}{single off} {" " 1 sonet internal-oscilla-
tor 0}+
```

```
6 = { 6 { off on 10 3 } {single off} { "" "" 1 sonet internal-oscilla-
tor 0 } +
7 = { 7 { off on 10 3 } {single off} { "" "" 1 sonet internal-oscilla-
tor 0 } +

super> list ports 0
port_num = 0
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
dis+
```

**Note:** remember to specify the `keepalive-interval` setting in milliseconds.

The default Cisco HDLC settings are:

```
super> list cisco
debug = off
```

- 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-enabled = on
```

- Keepalive activity can be turned off, the default is on.

```
keepalive-interval = 10000
```

- The default keepalive interval setting specifies how often the SONET interface sends keepalive messages, the default is every 10 seconds. - must specify in milliseconds

```
keepalive-error-thresh = 3
```

- 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.

*Tip: Use `set <field_name>?` to find out the available values to set.*

```
super> set keepalive-enabled?
keepalive-enabled:
 Turn Cisco HDLC keepalive on and off
 Boolean field, 'off' or 'on'

super> set keepalive-interval?
keepalive-interval:
 Number of seconds between keepalives
 Numeric field, range [0 - 32767]

super> set keepalive-error-thresh?
keepalive-error-thresh:
 Number of keepalives sent without response before marking the
 interface
 down.
 Numeric field, range [0 - 32767]
```

# Configuring the Frame Relay protocol

## 1. Specify interface names in *grifconfig.conf*

Identify the logical interfaces on the SONET card. As an example, we will configure Frame Relay on four logical interfaces on the SONET card in slot 2. They are logical interfaces 126 and 127 on the upper physical interface, and logical interfaces 128 and 129 on the lower physical interface. Null-modem cables are attached.

| #      | name | address     | netmask       | broad_dest  | argument |
|--------|------|-------------|---------------|-------------|----------|
| #      |      |             |               |             |          |
| go027e |      | 192.0.2.1   | 255.255.255.0 | 192.0.2.24  |          |
| go027f |      | 192.0.99.1  | 255.255.255.0 | 192.0.99.25 |          |
| go0282 |      | 192.0.130.1 | 255.255.255.0 |             |          |
| go0283 |      | 192.0.126.1 | 255.255.255.0 |             |          |
| go020  |      | -           | -             | -           | up       |
| go0280 |      | -           | -             | -           | up       |

Always use lower case letters when defining interface names.

If an interface is nonbroadcast (NBMA), do not include a destination address in its *grifconfig.conf* entry.

### *First interface configuration requirement*

When used in Frame Relay mode, SONET card configuration requires that the first logical interface on each physical interface must be configured. This enables the frame relay daemon (**fred**) to properly communicate LICS traffic via the card.

For example, if a SONET card in slot 2 is configured for Frame Relay, then the following interfaces need to exist in the card configuration:

```
go020 - slot 2, physical interface 0, logical interface 0
go0280 - slot 2, physical interface 1, logical interface 0
```

They also need to be configured as active PVCs in */etc/grfr.conf*. For example:

```
#/etc/grfr.conf file entry :
pvc go020 335 192.0.2.4 Name="north subnet"
pvc go0280 642 192.0.130.2 Name="south subnet"
```

## 2. Specify framing protocol

Be sure the framing protocol is set to Frame-Relay in the Card profile.

```
super> read card 8
CARD/8 read
super> list card 8

card-num* = 8
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 inter-
nal-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> set sonet = Frame-Relay
super> write
CARD/8 written
```

The next steps are in the `/etc/grfr.conf` configuration file. A copy template of the `grfr.conf` configuration file is in Appendix A of the *GRF Reference Guide*.

## Configure link parameters

### 3. Set Link parameters in `grfr.conf`.

Open the UNIX shell:

```
super> sh
```

Use a UNIX editor to edit the `grfr.conf` file to set (or change) the link parameters, then execute the **grfr** command to make the configuration (or changes) permanent. Refer to the **grfr** man page for more information.

**Note:** For Link parameter settings and changes to take effect, you must reboot the SONET card.

### Required link parameter descriptions

The format for the link parameter entry is:

```
Link slot port
```

where `slot` is the GRF chassis slot and `port` is upper (0) or lower (1). Examples are:

```
link 3 1
link 3 0
```

### Optional link descriptors

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Name=    | - string describing specific interface                           |
| Enabled= | - enables/disables link, Y N                                     |
| LMIType= | - None Standard  AnnexA AnnexD                                   |
| N391=    | - sets polling intervals per full status message, range is 1–255 |
| N392=    | - error reporting threshold, range is 1–10                       |
| N393=    | - error recovery threshold, range is 1–10                        |
| T391=    | - heartbeat poll interval, settings are 5 10 20 25 30            |



AutoAddGrif        - automatically adds PVCs from LMI to this logical interface

*Link descriptor examples*

```
link 3 0 LMType=AnnexD AutoAddGrif=go060
link 3 1 LMType=AnnexD name="Lower Port"
```

## Configure PVCs

### *4. Create PVCs and set parameters.*

Open the UNIX shell:

```
super> sh
```

Use a UNIX editor and edit `grfr.conf` to add/delete PVCs or to change PVC parameters. Then execute the **grfr** command (as `super_user`) to immediately install the configuration:

The **grfr** command format to install the `grfr.conf` file parameters is:

```
grfr -p slot_number
```

**Note:** To add or edit PVCs at any time, on the fly, use the **grfr** command. See section below.

### *Number of configurable PVCs*

The SONET card supports a total of 905 permanent virtual circuits (PVCs) per physical interface. These can be divided according to site requirements among the 256 logical interfaces.

### *DLCI numbering*

PVC numbering is called the data link circuit identifier, DLCI. The range of valid DLCI numbers is 16 through 921. DLCI 0 to 15 are reserved. When ANNEX A or D is selected, DLCI-0 is used for LICS traffic.

### *Required PVC parameters*

Three parameters are mandatory: Logical Interface, DLCI, and Peer IP address.

|                              |                                                                                                            |
|------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>lif</code>             | - a logical interface described in the Interface Section, <code>go0yz</code>                               |
| <code>DLCI</code>            | - the data link circuit identifier, ranges from 16–921                                                     |
| <code>Peer IP Address</code> | - IP address of station at other end of the PVC, specify 0.0.0.0 to resolve the address using Inverse ARP. |

### *Optional PVC parameters*

|                       |                             |
|-----------------------|-----------------------------|
| <code>Name=</code>    | - string describing PVC     |
| <code>Enabled=</code> | - enables/disables PVC, Y N |

### *PVC examples*

```
PVC go027e 100 0.0.0.0 Name="station0"
PVC go027f 101 222.222.222.101 Name="station1" Enabled=N
PVC go0280 102 222.222.222.102 Name="station2"
PVC go0281 103 222.222.222.103 Name="station3"
```

## Adding a logical interface on-the-fly

You can add a logical SONET interface to `grifconfig.conf` and activate the new PVC without resetting the media card. Do the following in this order

```
grfr -p <slot number>
grifconfig -I go0yz
```

where `go0yz` is the new logical interface just added. The **grfr** command creates the logical interface and brings up the PVC. The **grifconfig -I** command assigns an IP address to it.

If the interface already exists and you are just adding another PVC (DLCI) from it, then all you need to do is execute the **grfr** command. You cannot change link types on the fly, you can only add, delete, or modify PVCs.

## Adding 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 add a PVC or **grfr -c crp** to disable a PVC. You cannot change link type dynamically.

To add a PVC to the 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 go0d0 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 DLCI, slot, and link must be specified:

```
grfr -c ccp -f /etc/grfr.conf -i 606 -s 13 -l 0
```

Here is the response:

```
grfr Adding type 1, lif=go0d0, 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` command is sufficient. Specify the target DLCI to be disabled:

```
grfr -c crp -i 600 -s 13 -l 0
```

Here is the response:

```
PVC slot 13, link 0, dlci 600 deleted
```

## Assigning multiple DLCIs

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:

| #   | lif    | DLCI | Peer IP address |
|-----|--------|------|-----------------|
| PVC | go047e | 405  | 222.222.10.5    |
| PVC | go047e | 406  | 222.222.10.6    |
| PVC | go047e | 407  | 222.222.10.7    |
| PVC | go047e | 408  | 222.222.10.8    |

If the peer IP addresses are in different subnets, you need multiple interfaces:

| #   | lif   | DLCI | Peer IP address |
|-----|-------|------|-----------------|
| PVC | go040 | 405  | 222.222.10.5    |
| PVC | go041 | 406  | 222.222.11.5    |
| PVC | go042 | 407  | 222.222.12.5    |
| PVC | go043 | 408  | 222.222.13.5    |

## grfr command set

**grfr** has display commands that return useful information about Frame Relay links.

### Display commands

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

## Configuration and debug commands

```
-c cel, enable link, enable link on slot 3, port 1: # grfr -c cel -s 3 -l 1
```

```
-c cdl, disable link, disable link on slot 3, port 0: # grfr -c cdl -s 3 -l 0
```

**-c cep**, enable PVC, requires PVC to be specified: `# grfr -c cep -i 888 -s 3 -l 0`

```
-c cdp, disable PVC, requires PVC to be specified: # grfr -c cdp -i 888 -s 3 -l
0
```

**-c ccp**, configure PVC, requires the configuration file and the PVC to be specified:

```
grfr -c ccp -f /etc/grfr.conf -i dlci
```

**-c crp**, remove PVC, requires the PVC to be specified: `# grfr -c crp -i dlci`

```
-c ddl, display debug level: # grfr -c ddl
```

**-c csd**, set debug level, requires -d option to specify new level 0–4: # grfr -c csd -d  
3

Refer to the *GRF Reference Guide* for more information about the **grfr** command.

## Configuring Point-to-Point Protocol

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.

### 1. Identify the PPP logical interface

This entry in `grifconfig.conf` identifies the logical interface on the SONET card in slot 2. Type your entries in lower case.

```
name address netmask broad_dest argument
#
go020 192.0.2.1
```

### 2. Set framing protocol in Card profile

The framing protocol default for SONET cards is PPP. Check that the setting has not been changed:

```
super> read card 8
CARD/8 read
super> list
card-num* = 8
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 inter-
nal-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 }
```

### 3. Set PPP parameters in `grppp.conf` file

Open the UNIX shell:

```
super> sh
```

Use a UNIX editor and edit the `grppp.conf` file. The template for the `grppp.conf` configuration file is provided at the end of the parameter descriptions.

To make immediate, temporary changes, use the **grppp** command. Refer to the **grppp** man page for information.

**Note:** Settings done with **grppp** are lost when the SONET card is reset.

## Required interface parameter

To set up a PPP interface, specify the interface with the *interface\_name* in go0yz format:  
`interface interface_name`

```
configure SONET i/f in slot 6
interface go060
 enable negotiation trace
 enable ipcp
```

## Optional negotiation automata parameters

```
enable negotiation trace
 - writes PPP negotiation traces into /var/log/gr.console file.
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).
```

## 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, 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)
```

## 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,
 a value specified in 1/100 seconds begins the exchange of statistics between
 endpoints (default is 0)
```

## IPCP parameter

```
enable ipcp
 - required for IP traffic across PPP link (default is disabled)
```

## PPP status reporting

The **grppp** command displays status for PPP objects and configuration values. Use the **maint** commands to look at packet statistics.

## grppp status commands

Use the appropriate **grppp** command to display the desired information. Commands are entered in lower case, short forms of words can be used. Refer to the **grppp** man page for more information. These are the **grppp** status commands:

```
grppp show configuration
grppp show negotiation trace status
grppp show maximum configuration request count
grppp show maximum failure count
grppp show maximum terminate count
grppp show restart timer interval
grppp show lcp keepalive interval
grppp show lcp keepalive packet threshold
grppp show lcp mru
grppp show lcp status
grppp show lqr timer interval
grppp show lqr status
grppp show ipcp status
```

## Looking at a PPP configuration

Here is the output from a **netstat** command looking at the SONET interfaces:

```
netstat -in | grep go
go060 1500 <link46> 0 0 0 0 0
go060 1500 208.1.11 208.1.11.156 0 0 0 0 0
```

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 milliseconds, disabled
 Keepalive packet threshold: 5
LQR Configuration:
 LQR is disabled
 Timer interval: 0 milliseconds
IPCP Configuration:
 enable IPCP
OSINLCP Configuration:
 disable OSINLCP
go060>
```



## grppp.conf file

Figure 8-3 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
override 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.con-
sole sole
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 8-3. Template for grppp.conf file*

## Monitoring SONET OC-3c media cards

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.

### Preparing to use maint

First, switch to the **maint** GR 66> prompt. Enter:

```
grrmb
```

The new prompt appears:

```
GR 66>
```

Then, change the prompt to the SONET OC-3c media card you are working with. If the card is in slot 9, enter:

```
GR 66> port 6
```

This message is returned along with the changed prompt:

```
Current port card is 06
```

```
GR 06>
```

To leave the **maint** prompt, enter **quit**.

### Display maint commands

Use **maint 1** to view the list of receive side SONET OC-3c **maint** commands:

```
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 Combust 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] 82: Display K1/K2
[RX] 83: Display signal detect
[RX] 87: Frame Relay Arp Debug.
[RX] 88: Display Frame Relay PVC table
[RX] 89: Display Frame Relay PVC stats
```

Use **maint 101** to view the list of transmit side SONET OC-3c **maint** commands:

```
GR 06> maint 101
[TX]
[TX] 101: Display this screen of Options
[TX] 106: Display Combust Statistics
[TX] 112: Display HW Registers
[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] 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: 0xe (SDC2).
[RX] SDC2 Combust Xilinx version: 0x6.
[RX] SDC2 Switch Transmit Xilinx version: 0x5.
[RX] SDC2 Switch Receive Xilinx version: 0x0.
[RX] SW:
```

```
[RX] SONET Code Version: A1_4_3, Compiled Mon Nov 17 18:46:58 CST
1997,
[RX] in directory: /nit/A1_4_3/sonet/rx.
[RX] IF Library Version: 1.1.0.0, Compiled Mon Nov 17 18:41:12 CST
1997
```

## Display card configuration and status

```
GR 06> maint 3
[RX]
[RX] SONET Configuration and Status.
[RX] Framing Protocol: PPP.
[RX] Free Memory: 977636
[RX] SONET Line Mode:
[RX] Active Channel: PROTECTION
[RX] Channel Condition, State, or Request: Do Not Revert
```

## Display media statistics

Use **maint 4** to display media statistics:

```
GR 06> maint 4
[RX] Media Statistics
[RX] input:
[RX] Port Bytes Packets Errors Discards
[RX] -----
[RX] 0 00000000000000000044 000000000000000004 0000000000
0000000000
[RX]
[RX] output:
[RX] Port Bytes Packets Discards
[RX] -----
[RX] 0 00000001090156839632 00000000030282134437 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: 0
```

## 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] 00000002180460861576 00000000030284178633 0000000000
[RX]
```

```
[RX] output:
[RX] Bytes Packets Errors Overruns
[RX] -----
[RX] 00000000000000000000 000000000000000000 0000000000 0000000000
[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 IP switch 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 ARP table

Use **maint 8** to display the ARP table, here are the fields displayed:

```
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] =====
[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 signal detect

The **maint 83** command identifies which is the working channel, usually the upper interface is the default working channel:

```
GR 06> maint 83
[RX]
[RX] WORKING_CHANNEL: 1
[RX] PROTECTION_CHANNEL: 1
```

## Display TX combus statistics

The communications bus provides the 80-Mbs inter-card and IP switch control board communications path. Use **maint 106** to view Combust statistics on the transmit side:

```
GR 06> maint 106
[TX]
[TX] Combust Segment reassembly stats:
[TX] =====
[TX] Packets Re-assembled: 0
[TX] Segments Received: 0
[TX] Segments Dropped, no mem: 0
[TX] Packet Timeout: 0
[TX] Orphaned Segments: 0
[TX] Segments out of Sequence: 0
```

## List next hop data

Use **maint 145** to display the receive side hardware registers:

```
GR 06> maint 145
[TX] Location is: 0
[TX] Add: 0 Delete: 0 noNH: 0
[TX] Location is: 1
[TX] Add: 91 Delete: 6 noNH: 0
[TX] 0: 0.0.0.0 0:f8 6:00 UNREACH
[TX] 1: 0.0.0.0 0:f8 6:00 DROP
[TX] 2: 0.0.0.0 0:f8 6:00 BCAST
[TX] 3: 0.0.0.0 0:f8 6:00 RMS
[TX] 4: 0.0.0.0 0:f8 6:00 MCAST
[TX] 5: 0.0.0.0 0:00 6:00 DROP
[TX] 6: 0.0.0.0 0:fc 6:00 RMS
[TX] 7: 222.222.1.1 0:00 6:00 RMS
[TX] 8: 0.0.0.0 a:00 6:00 BCAST
[TX] 9: 0.0.0.0 a:00 6:00 LOCAL
[TX] 10: 0.0.0.0 a:00 6:00 FWD
[TX] 11: 0.0.0.0 b:01 6:00 BCAST
[TX] 12: 0.0.0.0 b:01 6:00 LOCAL
[TX] 13: 0.0.0.0 b:01 6:00 FWD
[TX] 14: 0.0.0.0 c:01 6:00 BCAST
[TX] 15: 0.0.0.0 c:01 6:00 LOCAL
[TX] 16: 0.0.0.0 c:01 6:00 FWD
.
.
.
```





# Ascend Tunnel Management Protocol

## 9

This chapter describes how to configure the GRF as an ATMP home agent in IP gateway mode.

This chapter contains:

|                                            |      |
|--------------------------------------------|------|
| Introduction to ATMP .....                 | 9-2  |
| GRF ATMP implementation .....              | 9-3  |
| Starting aitmd .....                       | 9-4  |
| Support for virtual private networks ..... | 9-5  |
| ATMP tunnel components .....               | 9-8  |
| Configuring the GRF as a home agent .....  | 9-17 |
| Extended ATMP configuration example .....  | 9-29 |
| Monitoring ATMP activity on the GRF .....  | 9-33 |
| Using grstat commands .....                | 9-37 |
| Using grfr commands .....                  | 9-39 |

## 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 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 router and the other unit is a TNT. Because the GRF receives packets through the tunnel which must then be routed, ATMP on the GRF applies only to IP networks.

## How ATMP connections work

Figure 9-1 shows a basic ATMP tunnel configuration:

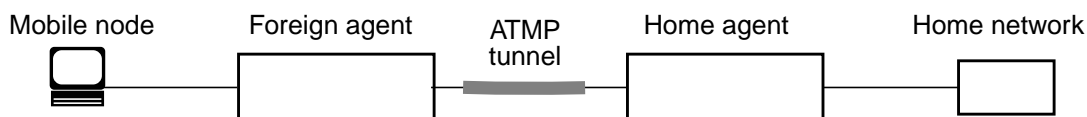


Figure 9-1. Components in a basic ATMP tunnel configuration

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 home network.
- **Foreign agent**  
The foreign agent is an Ascend unit dialed by the mobile node. It is the starting point of the ATMP tunnel.  
The foreign agent brings up an IP connection to the home agent over which to negotiate the tunnel. The foreign agent also authenticates the mobile node using a RADIUS profile that includes ATMP parameters. Typically, the foreign agent is a TNT unit.
- **Home agent**  
The home agent is the terminating part 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 the “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 number.

## ***GRF ATMP implementation***

GRF routers support a subset of the Ascend Tunnel Management Protocol (ATMP) that enables the GRF to function as an ATMP home agent in IP gateway mode

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 must be authenticated by the foreign agent.
- establishes a tunnel for a mobile node connected to a foreign agent
- encapsulates IP traffic received from the home network that is destined for the mobile node, and forwards the encapsulated IP traffic across the tunnel
- strips encapsulation from IP traffic received across the tunnel from the mobile node and forwards it as normal IP traffic to the home network  
This function defines the home agent gateway mode.
- supports HSSI and ATM OC-3c Frame Relay as WAN links to the 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.

Routing media:

- Ethernet from the foreign agent to the 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

## Starting *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. The file is removed normally with **rm** and the removal again saved with **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.

To check that **aitmd** is running, use the **ps** command:

```
ps ax | grep ai
```

If **aitmd** is running, it is reported in the second line of output. If there is no second line, **aitmd** is not running:

```
10199 p2 S+ 0:00.02 grep ai
 390 00- I 0:00.04 /usr/sbin/aitmd -F
```

## 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 may be from an ISP, enabling mobile nodes to dial-in to a corporate network, or between two corporate networks that 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.

### 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 dedicated WAN connection to the home network. In the case of the GRF home agent, this is a Frame Relay permanent virtual circuit (PVC) configured on a HSSI card, or an ATM virtual circuit on an ATM OC-3c card. The connection to the home network must be a dedicated link because the home agent does not route packets it receives through the tunnel. In gateway mode, the home agent will not bring up a connection based on the IP addresses of packets coming from a tunnel. Normal routed traffic must go on a different circuit.

### Scalability

The GRF home agent supports connections for up to 10,000 mobile nodes, up to 10,000 tunnels can operate simultaneously. Up to 500 home networks can be configured using any combination of HSSI Frame Relay and ATM PVC interfaces.

Each home network connected to the GRF home agent requires that a unique IP address be locally configured on the GRF. The foreign agent sees this IP address and associates it with a single home network. To the foreign agent, it appears that each home network is connected to a different GRF home agent and that there is one GRF per home network. Actually, hundreds of

home networks can connect to a single GRF home agent. This is supported by assigning the GRF a unique IP address for each connecting home network.

Unlike most IP addresses, the home network-associated IP address is not configured on a media card interface using the `/etc/grifconfig.conf` file. Instead, the administrator assigns each connecting home network a unique IP address in the `/etc/aitmd.conf` file on the GRF and in the RADIUS profile on the foreign agent. As tunnels are created, **aitmd**, the GRF ATMP daemon, automatically assigns these IP addresses to `atmp0`. `atmp0` is a pseudo interface managed by **aitmd**. All packets coming from a tunnel “arrive” at `atmp0` where they are decapsulated and forwarded to the correct card. The **netstat -i** command returns the list of `atmp0` IP addresses configured on a GRF system.

If a customer requires hundreds or thousands of home agents, GRF home agent configuration must be carefully organized. Connections to the 500 home networks should be configured according to how much bandwidth the mobile nodes may require. One interface (`gs030` or `ga028`) is consumed per home network. Given the ATM and HSSI per-card interface limits, you cannot configure 500 home networks on a single media card.

### *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.

## 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 implementation only supports encapsulated IP packets.

The GRE header includes the home agent `atmp0` IP address, the foreign agent IP address, and the tunnel ID assigned by the home agent. IP and GRE packet header contents depend upon the direction of the packet. Figure 9-2 shows how the agents manage IP packets so that only encapsulated packets traverse the tunnel:

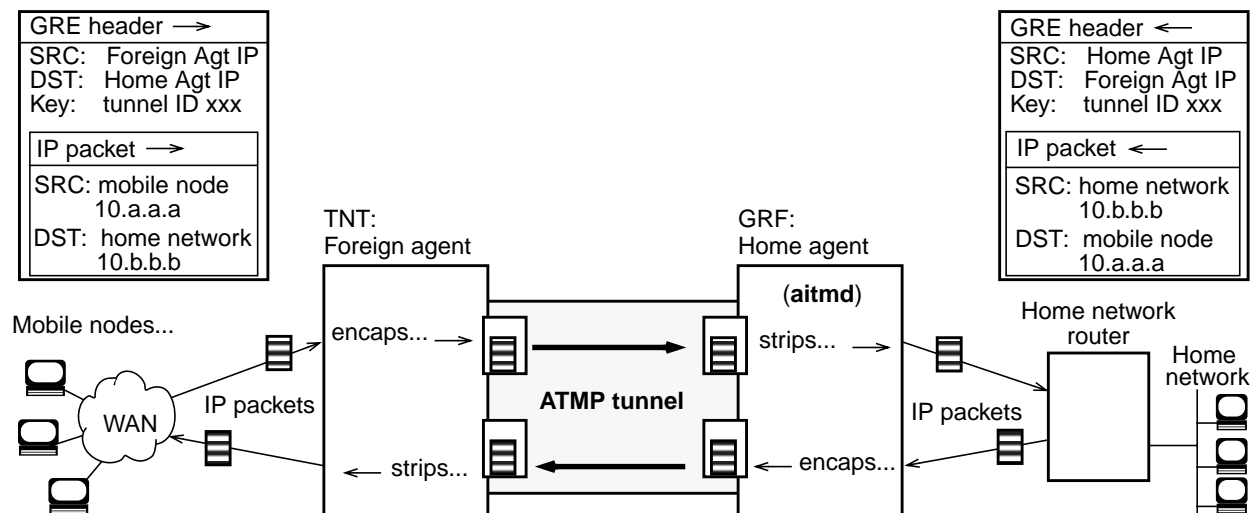


Figure 9-2. Movement of encapsulated packets through the tunnel

After a mobile node establishes a dial-up connection to the foreign agent, the foreign agent uses the node's RADIUS profile to obtain the routable IP address of the home agent assigned to this mobile node. Over a normal routed IP connection, the foreign agent negotiates the tunnel with the home agent. If the tunnel is established, the mobile node sends its packets destined for its home network over the dial-up connection. The foreign agent encapsulates the packets with a routable GRE header that uses the home agent's `atmp0` IP address as the destination address. Now "in the tunnel," these packets are routed across the Internet (or other WAN) to the GRF. The GRF strips the encapsulation and retrieves the tunnel ID. **aitmd** translates the tunnel ID to determine the target home network and dedicated link. The GRF forwards the packets across the link to the home network's router. The home network router completes the routing within the private network.

If the mobile node requests data, the home network router sends those data packets bearing the mobile node's private network address to the GRF home agent. The GRF encapsulates those non-routable packets and they transit the tunnel to the foreign agent. The packets continue through the dial-up connection to the mobile node. When the mobile node terminates the dial-up, the tunnel is also terminated. If the dedicated link to the home network is lost, the tunnel may be terminated.

## ATMP tunnel components

This section provides more information about tunnel components and operations.

### Tunnel ID

**aitmd** creates and maintains a mobile node lookup table that contains the following for each mobile node:

- mobile node IP address
- home network IP address
- tunnel ID
- foreign agent IP address
- home agent IP address

The home agent ATMP daemon, **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 home networks to the appropriate foreign agent. The mobile node's private network IP address is not used by the home agent because different mobile nodes can be using the same 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 Rx/BRx/Tx/BTx
[RX] - - - - - - - - - - - - - - - - - - - - - - - -
[RX] 6 02:01:888:0000 CirHomAg 10.9.9.9 0/0/0/0
```

The columns are 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
- **S:P:s0:s1 on ATM:**  
S = slot, P = port, s0 = ATM VPI, s1 ATM VCI
- state of link, either HomeAgent (the Ethernet card) or CirHomAg, the HSSI or ATM card owning the link to the home network



- home agent's IP address
- packets and bytes received, transmitted

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
- **S:P:s0:s1 on ATM:**  
S = slot, P = port, s0 = ATM VPI, s1 ATM VCI

## Tunnel negotiation

Tunnel negotiation is started by the foreign agent when the agent receives a request from a mobile node to connect to a target home network. From the RADIUS profile, the foreign agent determines which home agent is the gateway to the target home network and sends a tunnel request message to that home agent.

When the home agent is a GRF, the GRF requests a password. If the password is verified in the GRF's `/etc/aitmd.conf` file, the tunnel is established between the GRF home agent and the foreign agent. The GRF ATMP daemon, **aitmd**, creates a unique tunnel ID and sends it to the foreign agent. The foreign agent adds the tunnel ID to the GRE headers it adds to the front of the IP packets coming from the requesting mobile node. The GRF accepts the encapsulated packets forwarded through the tunnel from the mobile node. The GRF strips off the GRE encapsulation and forwards the IP packet to the target home network.

The home network can forward packets to the mobile node across the dedicated connection to the home agent. An example of packets coming from the home network is the mobile node downloading accumulated e-mail files. The GRF encapsulates all packets it receives from this connection and automatically forwards them to the associated foreign agent.

The components of an established tunnel are shown in Figure 9-3.

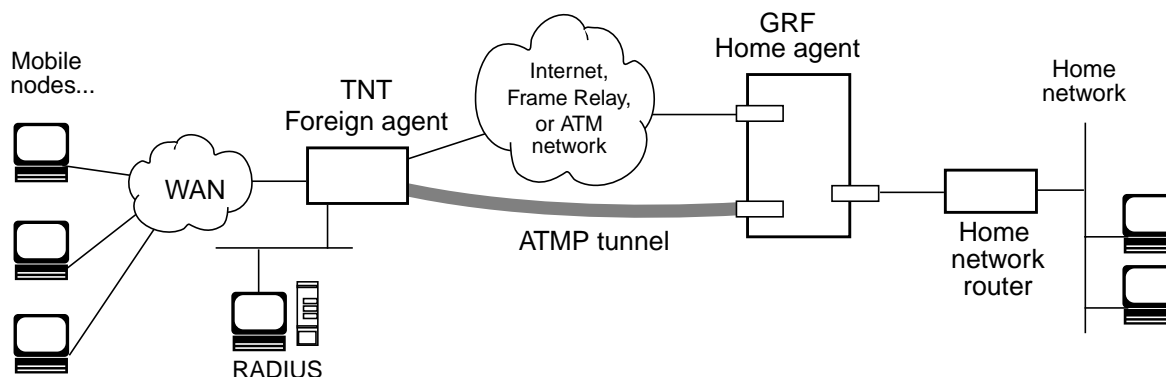


Figure 9-3. ATMP tunnel negotiation

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. If the circuit between the home agent and the home network is dropped for any reason, the GRF detects the lost link, notifies the foreign agent, and the tunnel is also dropped.

## Life-cycle of a tunnel

This is a typical scenario describing how an ATMP tunnel is requested, established, used, and torn down between a TNT foreign agent and a GRF home agent.

### *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.
- The TNT foreign agent locates a connection profile or RADIUS profile for the GRF home agent based on the Ascend-Home-Agent-IP-Addr attribute in the mobile node's RADIUS profile.

### *Foreign agent tunnel negotiation:*

- The TNT foreign agent connects to the GRF home agent over a regular, routed IP connection using the IP address from the mobile node's RADIUS profile. The foreign agent informs the GRF home agent that the mobile node is connected and requests a tunnel.
- 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, GRF tunnel software (**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:*

- 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 GRF link to the home network goes down, the GRF notifies the foreign agent and terminates the tunnel.

## Mobile node IP addresses

The ATMP tunnel and home agent gateway circuit operate independently of the IP 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, or virtual private network, address spaces do not mix with each other or with the normal, publicly-routed IP address space. Two or more mobile nodes can use the same unregistered IP address if they belong to different private home networks.

Even when mobile nodes in different private networks have the exact same IP address, the home agent sends their packets to the correct home network. This is because of configuration information in the mobile node's RADIUS profile maintained on the foreign agent.

### *RADIUS profile*

The mobile node's RADIUS profile contains the routable atmp0 IP address for the node's assigned home agent in the Ascend-Primary-Home-Agent parameter. 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. Refer to the *MAX TNT RADIUS Configuration Guide* for more information.

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,
 Ascend-IP-Route=Route-IP-Yes,
 Framed-Address=10.1.1.2,
 Framed-Netmask=255.255.255.0,
 Ascend-Primary-Home-Agent=aaa.aaa.aaa.aaa,
 Ascend-Home-Agent-Password="TntCodexyz"
 Ascend-Home-Agent-UDP-Port = 5150
 Ascend-Home-Network-Name = minnesota
```

The atmp0 IP address of the home agent assigned to mobile node XYZ is specified in the Ascend-Primary-Home-Agent parameter. In this example the atmp0 address is aaa.aaa.aaa.aaa. When the foreign agent encapsulates packets coming from the mobile node, the agent uses that IP address as the destination address in the GRE header.

## Home agent atmp0 IP addresses

A GRF can be configured as home agent for thousands of mobile nodes but there will be no indication (i.e., an IP address) of this activity found in the GRF's `/etc/grifconfig.conf` file. This is because the IP address assigned for each home agent is configured in the `/etc/aitmd.conf` file.

You enter the home agent `atmp0` IP address in two files:

- in `/etc/aitmd.conf` on the GRF
- as the Ascend-Primary-Home-Agent parameter in the mobile node's RADIUS profile (maintained on the TNT foreign agent)

The foreign agent uses the home agent `atmp0` IP address it finds in the RADIUS profile to contact the home agent when it receives a tunnel request from the mobile node.

As shown in Figure 9-4, the foreign agent continues to use the `atmp0` IP address as the destination address in the GRE header appended to packets coming from the mobile node.

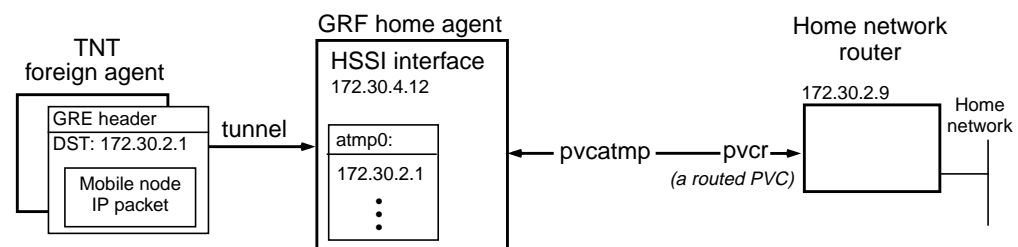


Figure 9-4. How the `atmp0` IP address is used

The `atmp0` IP address is not important from the perspective of the home network router. To enable hosts and routers on the home network to reach the mobile node, you must configure a static route in the home network router. The static route uses a normal IP address assigned to that HSSI interface with the dedicated link to the network router. The 172.30.4.12 IP address is statically configured on the home network router to be used only for setting up routes in the direction of the home agent. The static route must specify 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.

You can use the `netstat -i` command to display the `atmp0` home agent addresses associated with a specific GRF system:

```
super> netstat -i
```

| Name | MTU  | Network    | Address           | Ipkts  | Ierrs | Opkts | Oerrs | Col |
|------|------|------------|-------------------|--------|-------|-------|-------|-----|
| de0  | 1500 | <link1>    | 00:c0:80:86:16:e2 | 63100  | 0     | 25949 | 0     | 70  |
| de0  | 1500 | 281.142.10 | carol             | 62362  | 0     | 25747 | 0     | 70  |
| rmb0 | 616  | <link2>    | 00:00:00:00:00:00 | 162694 | 10    | 93803 | 0     |     |

## Ascend Tunnel Management Protocol

### ATMP tunnel components

---

|       |      |           |           |        |    |       |   |   |
|-------|------|-----------|-----------|--------|----|-------|---|---|
| rmb0  | 616  | <GRIT>    | 0:0x40:0  | 162694 | 10 | 93803 | 0 | 0 |
| lo0   | 1536 | <link3>   |           | 3230   | 0  | 3230  | 0 | 0 |
| lo0   | 1536 | <GRIT>    | 0:0x48:0  | 3230   | 0  | 3230  | 0 | 0 |
| atmp0 | 1536 | <link4>   |           | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.1.1 | 2,0,100,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.1.1 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.1.2 | 2,1,101,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.1.2 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 203.1.3.1 | 0,1,401,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 203.1.3.1 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.2.1 | 2,0,201,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.2.1 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.2.2 | 2,0,202,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.2.2 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.2.3 | 2,0,203,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.2.3 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.2.4 | 2,0,204,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.2.4 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 218.1.2.5 | 2,0,205,0 | 0      | 0  | 0     | 0 | 0 |
| atmp0 | 1536 | 0/32      | 218.1.2.6 | 0      | 0  | 0     | 0 | 0 |

## ATMP gateway circuit (pvcatmp)

The circuit (PVC) from the GRF home agent to the home network is configured as an ATMP gateway circuit. On a HSSI card this circuit is called a PVCATMP (in `/etc/grfr.conf`). On an ATM card, it is called 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 dedicated circuit. Traffic the GRF receives from a dedicated circuit is tunneled to the foreign agents (mobile nodes) associated with that circuit.

When an encapsulated packet from the mobile node is received by the GRF via the foreign agent, the GRF strips off the encapsulation and forwards the decapsulated packet across the dedicated circuit to the home network associated with that mobile node.

## Number of home networks

The GRF supports up to 500 home networks per system. Distributing the load across multiple HSSI and ATM media cards ensures efficient throughput. Based on the amount of bandwidth you estimate mobile node communications may require, you may prefer to allot the home networks among several media cards. Also, one interface (a `gs0yz` or a `ga0yz`) is consumed per home network. Given per-card interface limits, 500 home networks cannot be configured on a single card

## Connection from home network

You must establish a static Frame Relay link from the home network to the GRF home agent, a PVCRR (routed PVC). Assign the PVCRR the same DLCI value specified for the dedicated HSSI PVCATMP or the ATM OC-3c PVC circuit.

For example, if a HSSI card in slot 3 is configured for frame relay, then the following interfaces need to exist in the card configuration:

|                     |                                           |
|---------------------|-------------------------------------------|
| <code>gs030</code>  | - slot 3, physical link 0, logical port 0 |
| <code>gs0380</code> | - slot 3, physical link 1, logical port 0 |

**Note:**

- `gs030` is the first logical interface on the upper HSSI port, physical interface 0,
- `gs0380` is the first logical interface on the lower HSSI port, physical interface 1.

## OSPF advertises home network addresses

The home network addresses configured in the `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.

Here is an example of `gated.conf` entries to configure an `atmp0` interface in OSPF:

```
ospf yes { ## START ospf
traceoptions "/var/tmp/gated.ospf" replace size 1000k files 2 state
all ;
backbone{
 stubhosts {10.0.0.175 cost 10;}; ## routerid alias

INTERFACES
interface all cost 10 { disable; priority 1; }; ## default intf config
interface atmp0 cost 10 { passive; }; ## atmp home network

interface 10.2.2.175 nonbroadcast cost 30 { enable ; ## FDDI to base
 priority 1;
 routers {10.2.2.80; }; ## FDDI to base
};
```



## Configuring the GRF as a home agent

Per TNT-connected mobile node, there are three areas to configure in the GRF home agent:

- connection to a home network (via HSSI Frame Relay or via ATM OC-3c PVC)
- ATMP parameters in `/etc/aitmd.conf`
- configuration information to exchange / provide for the TNT foreign agent

The next sections briefly describe the tasks for each configuration. Illustrated examples follow the configuration descriptions.

### A. Connection to a home network

Individual home network configuration steps for both HSSI and ATM OC-3c cards are discussed in this section.

#### HSSI Frame Relay configuration

##### Overview

The connection to a home network is made across a Frame Relay link from a HSSI card. There are four steps:

1. Configure the Frame Relay link in the *Link* section of `/etc/grfr.conf`.
2. Configure the dedicated link in the *PVCATMP* section of `/etc/grfr.conf`.
3. Configure that dedicated link as a blank interface in `/etc/grifconfig.conf`.
4. Because link management does not initialize correctly on HSSI cards unless the first logical interface on each physical interface is configured as a blank interface in `/etc/grifconfig.conf`, you must also configure the first interface in `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**).

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 GRF chassis slot, HSSI physical port 0 or 1, and any optional link management parameters:

```
Slot Port Optional Parameters
==== ==== =====
link 2 1 Name="minnesota" Linktype=UNI-DCE
```

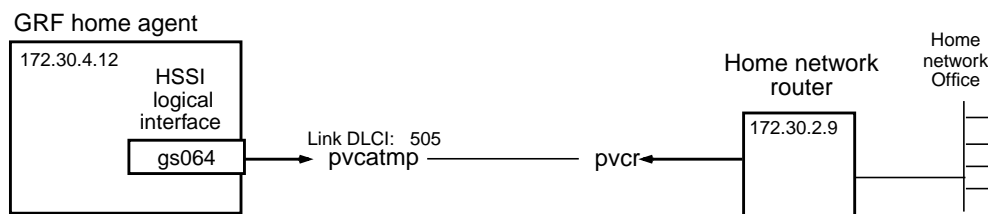
## Optional link parameters

These definitions are from the `/etc/grfr.conf.template` file included at the end of this guide.

- 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-TDE|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 dedicated 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 GRF) 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 gs064 505 172.30.2.9 Name="office"
```



- The `lif` is the logical interface name of the GRF home agent interface to which the PVCATMP is being configured.
- The DLCI is the value assigned to the PVCATMP link going to the home network.
- The peer IP address is that of the station at the other end of the PVCATMP link. 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.

## 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 = 55000000
- Bc value = 55000000
- 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:

```
name address netmask broad_dest arguments
gs064 - - - up
```

- 4 On HSSI media cards, Frame Relay link management does not initialize correctly unless the first interface on the card is configured and appears in `/etc/grifconfig.conf`. Based on the example above with the HSSI card installed in slot 6, here is the required entry in `/etc/grifconfig.conf`:

```
name address netmask broad_dest arguments
gs060 - - - up
```

When you finish, there will be two entries in `/etc/grifconfig.conf`:

```
name address netmask broad_dest arguments
gs064 - - - up
gs060 - - - up
```

## 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 excerpt from a *PVCATMP* section, 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 gs0280 888 192.0.2.50 Name="minnesota" Cir=55000000
Bc=55000000 Be=0
```

## 1. ATM OC-3c configuration

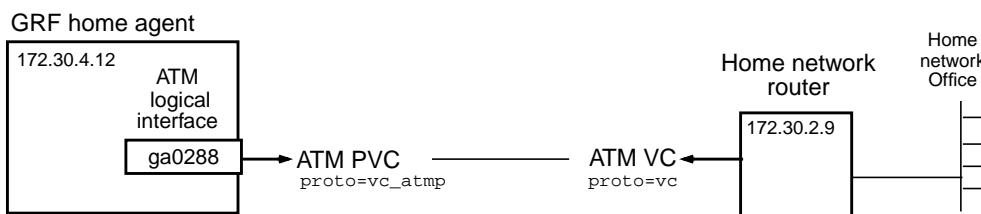
### Overview

The GRF ATM connection to a home network is made across a PVC from an ATM OC-3c card.

There are four steps:

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 dedicated link in the *PVC* section of `/etc/gratm.conf`.
4. Configure that dedicated link as a blank interface in `/etc/grifconfig.conf`.

The home network router connects to the GRF ATM PVC through an ATM VC.



A copy of the `/etc/gratm.conf` file is in the *GRF Reference Guide*, you can also refer to its man page (**man gratm.conf**).

From the CLI, establish a UNIX shell to edit the configuration files:

```
super> sh
#
```

There are four configuration tasks in the `/etc/gratm.conf` file. Repeat these tasks for each home network connected to the GRF.

- 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 In the *Interfaces* section of the `/etc/gratm.conf` file, specify the logical interface name of the circuit dedicated to the home network and assign it the `traffic_shape` name you defined in step 1:

```
Interfaces
Interface ga0288 traffic_shape=high_speed_high_quality
```

- 3 In the *PVC* section of the `/etc/gratm.conf` file, specify the logical interface name of the circuit dedicated to the home network , 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=high_speed_high_quality
```

- 4 ATMP requires that a Frame Relay PVC interface be configured as blank or inactive in `/etc/grifconfig.conf`. Based on the example above, here is the required entry:

```
name address netmask broad_dest arguments
ga0288 - - - up
```

## 2. Configure GRF ATMP parameters in 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 in which to edit the configuration files:

```
super> sh
#
```

In the shell, use a UNIX editor to edit `/etc/aitmd.conf` and other 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 `aitmd.conf` file has two types of 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, dependent upon DNS, they are less reliable than IP addresses.

Notice that you specify the IP address with the `addr` keyword. The password can be up to 20 alphanumeric characters. This password is a “shared secret” that must also be entered in the mobile node’s RADIUS profile maintained on the TNT. This `/etc/aitmd.conf` excerpt shows the required entries:

```
foreign_agent {
 addr yyy.yyy.yyy.yyy; # IP address of the foreign agent
 password TntCodexyz; # shared secret
}
```

- 2 Home networks must be identified in more detail.

In Frame Relay if you specify a text string name for the network, the name will also appear in **grfr** commands such as **grfr -c dlc**. If you do not specify a name, the **grfr** command displays a variation of the slot:port:s0:s1 identification.

This netmask is the netmask of the network for all of the mobile nodes attached to this home network. Netmask size is typically either 16 or 24 bits, but any other normal netmask length can be used. A 16-bit netmask usually means a mobile node tunneled to a 10.4.x.x network. A 32-bit netmask usually indicates a single node on the mobile network. This `/etc/aitmd.conf` excerpt shows the required entries:

```
home_network {
 name minnesota; # text string name. no more than 31 characters
 netmask_size 16; # number of bits in the home network netmask
}
```

```
home_agent_addr aaa.aaa.aaa.aaa;
 # IP address (atmp0) of the home agent on the grf
 # Only one home_network may use this address
circuit {
 card 2; # HSSI or ATM card in slot 2
 port 0; # Port (or link) 0
 s0 888; # DLCI 888 for HSSI card, VPI value for ATM OC-3c card
 s1 0; # Zero for HSSI card, VCI value for ATM OC-3c card
}
}
```

### 3. Configuration links to the TNT foreign agent

The initial (pre-tunnel) communication from the TNT foreign agent to the GRF home agent is through a normally-routed IP connection through 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 using the home agent `atmp0` IP address retrieved from the mobile node's RADIUS profile maintained on the TNT. Routed messages are exchanged to negotiate the tunnel. No special ATMP-related configuration is needed at this time.

Because a unique IP address (`atmp0`) 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.

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="top-secret"
 Ascend-Metric=2,
 Framed-Protocol=PPP,
 Ascend-IP-Route=Route-IP-Yes,
 Framed-Address=200.1.1.2,
 Framed-Netmask=255.255.255.0,
 Ascend-Primary-Home-Agent=aaa.aaa.aaa.aaa,
 Ascend-Home-Agent-Password="TntCodexyz"
 Ascend-Home-Agent-UDP-Port = 5150
 Ascend-Home-Network-Name = minnesota
```

Four parameters in the RADIUS profile relate to the GRF home agent configuration:

- Ascend-Primary-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 relate to the GRF home agent.

**TNT point-of-view:** This is the *hostname* or IP address of the first home agent the foreign agent tries to reach when setting up an ATMP tunnel for this mobile node, it is the atmp0 address linked to the target home network.

```
Ascend-Primary-Home-Agent=aaa.aaa.aaa.aaa,
```

This is the corresponding hostname or IP address entry in the GRF's /etc/aitmd.conf configuration file for the home network named "minnesota":

```
home_network {
 name "minnesota" ; # text string name. no more than 31 characters
 netmask_size 16; # number of bits in the home network netmask
 home_agent_addr aaa.aaa.aaa.aaa;
 # IP address (atmp0) of the home agent on the grf
```

Specify the home agent's IP 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 atmp0 psuedo interface 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 IP address for each home network. These IP addresses allow the GRF to connect to multiple home networks.

### atmp0 psuedo interface

The atmp0 interface is used by the operating system and must not be referenced (entered by a user) in the /etc/grifconfig.conf file.

To check which IP addresses are assigned to atmp0 after the **atmp** daemon loads a home network configuration, use the **netstat -i** command. Enter:

```
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 case, 172.30.1.9 and 221.1.1.2 are IP addresses for two different home agents. When the GRF is connected to multiple home networks, you see a corresponding number of IP addresses reported by **netstat -i**.

### *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 GRF's `/etc/aitmd.conf` configuration file. Use a text string of up to 20 characters, the default value is null.

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 TntCodexyz; # shared secret
```

### *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.

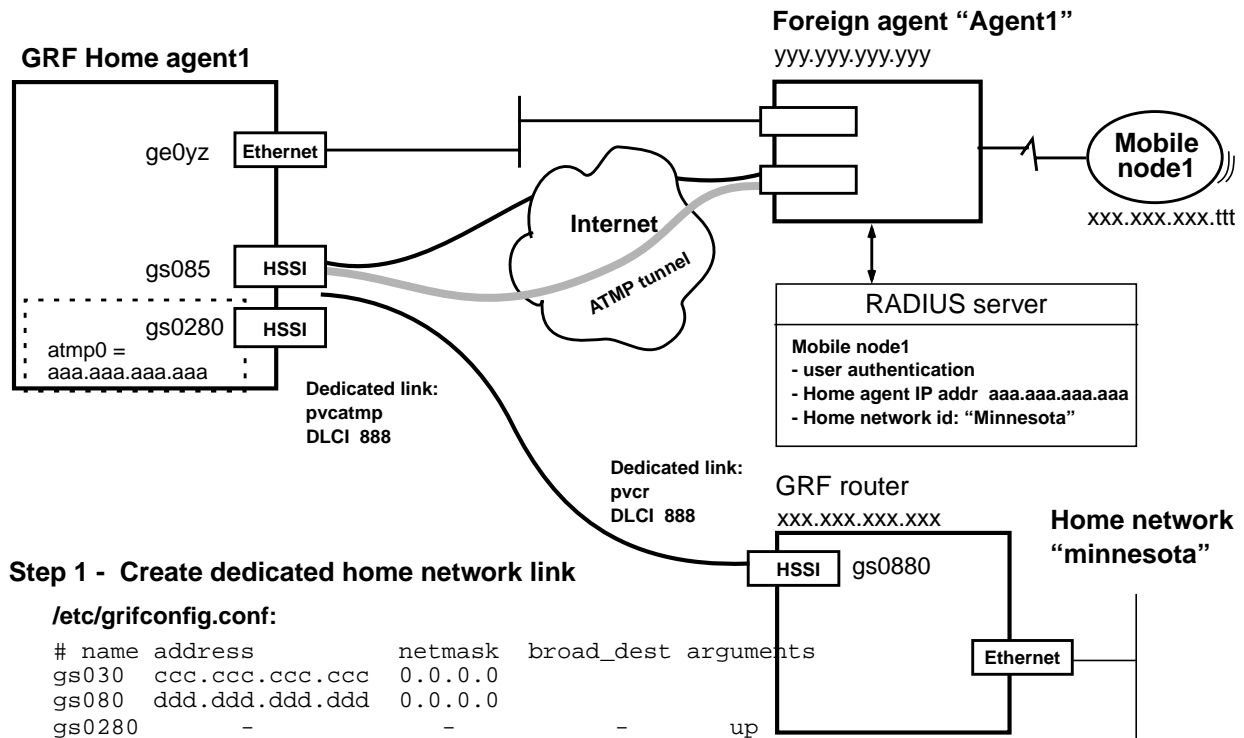
```
Ascend-Home-Network-Name = minnesota
```

This is the corresponding entry in the GRF's `/etc/aitmd.conf` configuration file:

```
home_network {
 name minnesota; # text string name. no more than 31 characters
```

## GRF ATMP tunnel gateway - example 1

Figure 9-5 illustrates the GRF configuration file entries needed to set up an ATMP tunnel to “Agent1”. Note that aaa.aaa.aaa.aaa is in the public address space, xxx.xxx.xxx.xxx is in the VPN address space.



**/etc/grifconfig.conf:**

```
name address netmask broad_dest arguments
gs030 ccc.ccc.ccc.ccc 0.0.0.0
gs080 ddd.ddd.ddd.ddd 0.0.0.0
gs0280 - - - up
```

**/etc/grfr.conf:**

```
#Link section
Slot Port Optional Parameters
=====
link 2 1 name="minnesota" LMType = AnnexD

#PVCATMP section
lif DLCI Peer IP Address Optional Parameters
=====
pvcatmp gs0280 888 xxx.xxx.xxx.xxx Name="minnesota" CIR=55000000 Bc=55000000 Be=0
```

### Step 2 - Configure ATMP

**/etc/aitmd.conf:**

```
foreign_agent
 addr yyy.yyy.yyy.yy
 password TntCodexyz
home_network
 name minnesota
 netmask_size 16
 home_agent_addr aaa.aaa.aaa.aaa
 circuit
 card 2
 port 1
 s0 888
```

### Step 3 - Configure GRF home network router

**/etc/grfr.conf:**

```
#Link section
Slot Port Optional Parameters
=====
link 8 1 name="agent1" LMType = AnnexD

#PVC section
lif DLCI Peer IP Address Optional Parameters
=====
pvcr gs0880 888 xx.xx.xx.zz
```

**/etc/grifconfig.conf:**

```
name address netmask
gs0880 eee.eee.eee.eee 0.0.0.0
```

(a non-routable address)

Figure 9-5. GRF configuration for an ATMP tunnel

## GRF ATMP tunnel gateway - example 2

Figure 9-6 shows ATMP configuration when more than one home network is connected to a GRF home agent. Note that the GRF has two IP addresses assigned as atmp0 values, and that these IP addresses are not entered in `/etc/grifconfig.conf`. Here are file entries to set up an ATMP tunnel and the dedicated connections to home networks "minnesota" and "alameda":

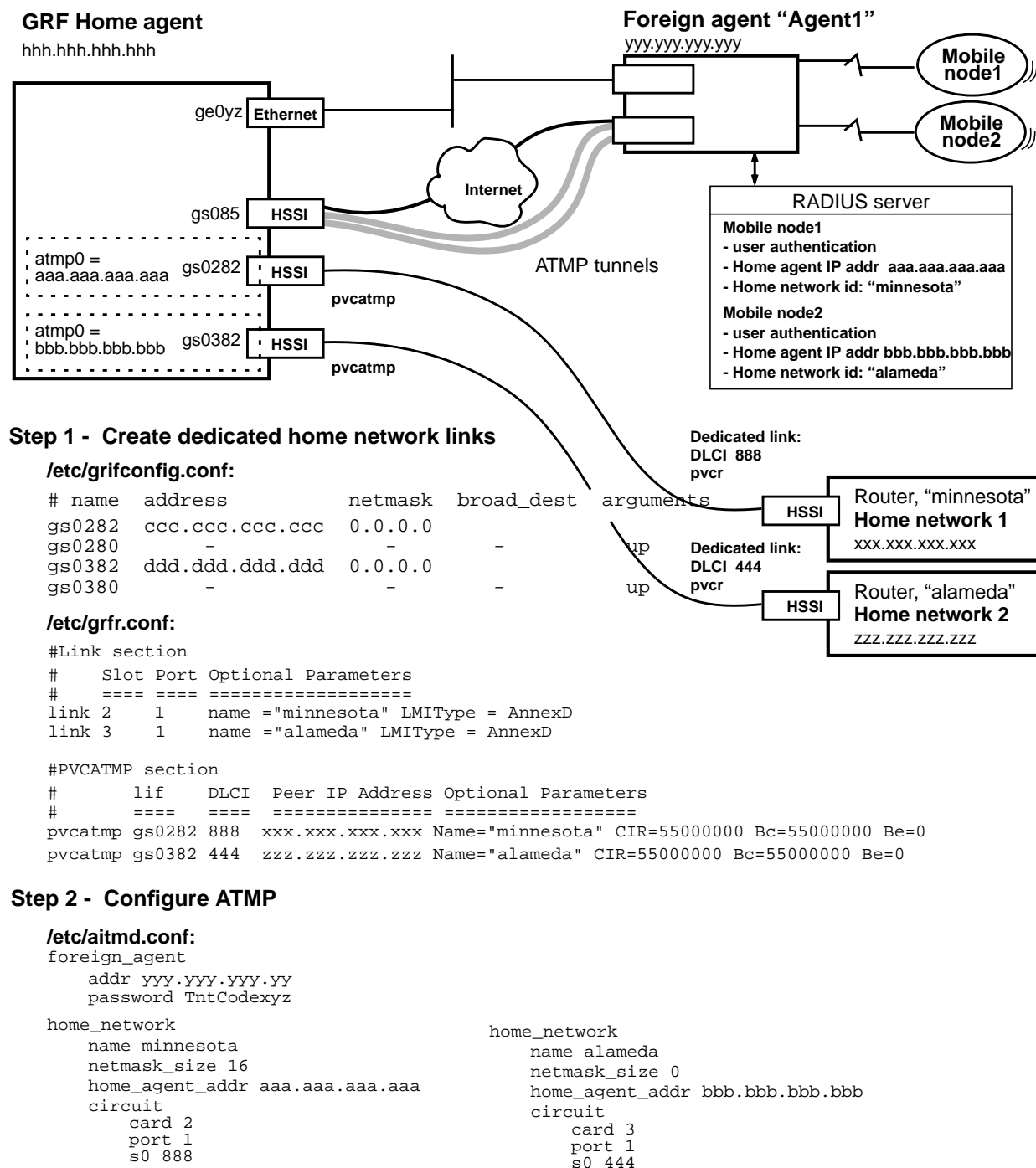


Figure 9-6. GRF ATMP configuration supporting multiple home networks

## Extended ATMP configuration example

This example provides a start to finish procedure to establish and check out a tunnel configuration.

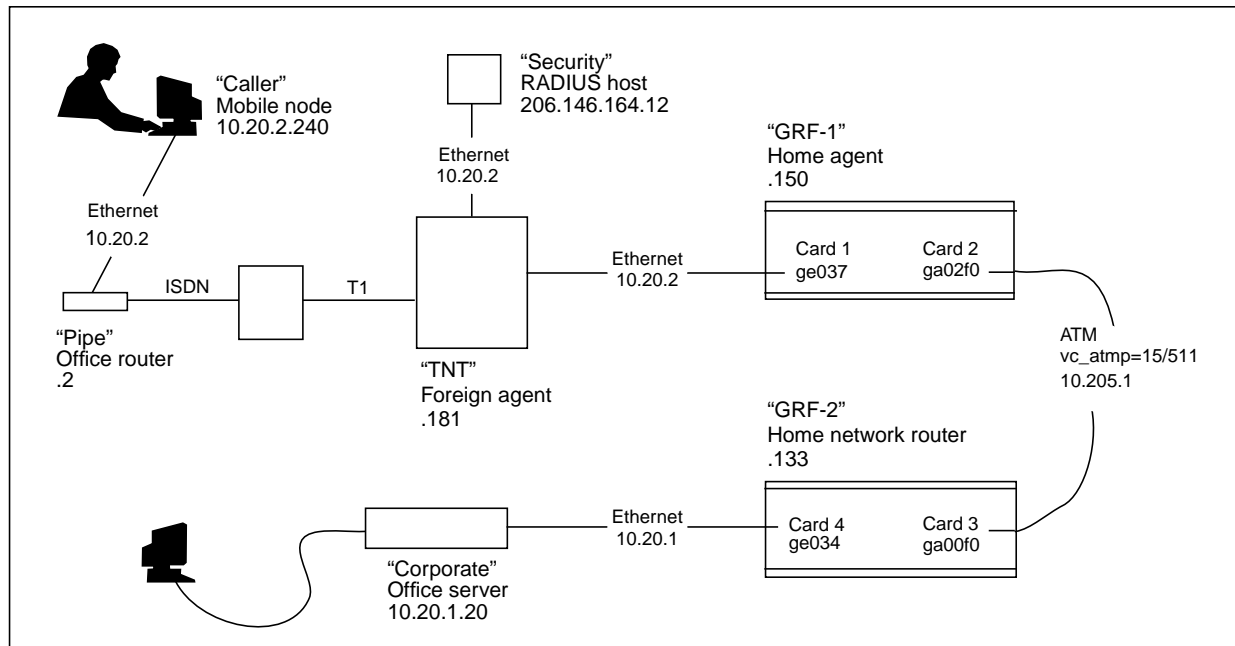


Figure 9-7. Diagram of extended

### Configure mobile node and office server

This table shows the chronological configuration for the mobile node and the office server:

| Action                                      | Mobile node "Caller"                                                  | Office server "Corporate"                                            |
|---------------------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------------|
| Associate host name with IP address:        | /etc/hosts:<br>10.20.2.240 caller-ether1                              | /etc/hosts:<br>10.20.1.20 corporate-ether1                           |
| Associate network interface with host name: | /etc/hostname.qel:<br>caller-ether1                                   | /etc/hostname.qel:<br>corporate-ether1                               |
| Configure interface:                        | \$ ifconfig qel plumb                                                 | \$ ifconfig qel plumb                                                |
| Initialized interface:                      | \$ ifconfig qel inet 10.20.2.240<br>netmask 255.255.255.0 -trailersup | \$ ifconfig qel inet 10.20.1.20<br>netmask 255.255.255.0 -trailersup |
| Verify interface:                           | \$ ifconfig qel<br>\$ ping 10.20.2.240                                | \$ ifconfig qel<br>\$ ping 10.20.1.20                                |
| Add route:                                  | \$ route add 10.20.1.0 10.20.2.2 1                                    | \$ route add 10.20.2.0 10.20.1.133 1                                 |
| Verify route:                               | \$ netstat -rn   grep 10.20.1.0                                       | \$ netstat -rn   grep 10.20.2.0                                      |

### *Configure mobile host and RADIUS client*

This table lists the configuration to register “Pipe” as a legal mobile host and the “TNT” as a client on the “Security” server:

| Action                                    | RADIUS authentication host “Security”                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Add mobile host’s Authentication Profile: | <pre>/etc/raddb/users  pipe Password = "dialup"                # Caller's Pipeline Ascend-Metric = 2, Framed-Protocol = PPP, Framed-Address = 10.20.2.2,              # Pipe address Framed-Netmask = 255.255.255.0, Ascend-Primary-Home-Agent = 221.1.1.3    # IP of home agent Ascend-Home-Agent-Password = "vpnethpw", # atmp0 ATMP tunnel (pw) Ascend-Home-Network-Name = "vpnatm",     # atm ATMP circuit to HN Ascend-Home-Agent-UDP-Port = 5150, Ascend-Idle-Limit = 60</pre> |
| Register TNT as an authentication client: | <pre>/etc/raddb/clients  tnt.somesite.com secretxyz #hostname encryption key</pre>                                                                                                                                                                                                                                                                                                                                                                                                   |

## Configure the home agent GRF

This table shows the GRF-1 configuration and the `/etc/aitmd.conf` entries that set Caller up as a mobile node (via Pipe):

| Action                                                                       | Configure the mobile node on the GRF-1 home agent                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Configure GRF-1 ATMP parameters:                                             | <pre> /etc/aitmd.conf  foreign_agent {     addr 206.146.160.181; # for Caller                         # IP address of the foreign agent     password vpnethpw;   # ethernet tunnel }  home_network {     name vpnatm;         # for Caller                         # ATM circuit     netmask_size 24;     # number of bits in the home network netmask     home_agent_addr 221.1.1.3 # IP addr of the home agent on the GRF.                         # Only 1 home network may use this address      circuit {         card 2;          # define the PVMATMP associated with the home network                         # ATM card slot         port 1;          # Port (or link) first is 0         S0 15;           # VPI         s1 5111;         # VCI     } } </pre> |
| Load the configuration, start the ATMP daemon, and log the daemon's message: | <p>If <b>aitmd</b> is not running, enable ATMP support by creating the file <code>/etc/aitmd.run</code> with the command:</p> <pre>\$ touch /etc/aitmd.run</pre> <p>If <b>aitmd</b> is running, send a HUP signal to aitmd and log the daemon's message:</p> <pre>\$ kill 1 &lt;aitmd process_id&gt;; aitmd -L TsTFLA -l /var/log/aitmd.log</pre>                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Verify ATMP configuration of GRF home agent addresses                        | <pre>\$ netstat -i   grep atmp0</pre> <p>Check that the 221.1.1.3 address is listed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Configure interfaces on GRF -1                                               | <pre> /etc/grifconfig.conf # Card 1 ge037 204.101.7.150      # connection to TNT # Card 2 ga02f0 - - - up         # dedicated for vc_atmp </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Configure ATM interfaces                                                     | <pre> /etc/gratm.conf Traffic_Shape=hshq peak 155000 sustain=155000 burst=2048 qos=high Traffic_Shape=hslq peak 155000 sustain=155000 burst=2048 qos=low Traffic_Shape=lshq peak 14000 sustain=14000 burst=32 qos=high Traffic_Shape=lsqp peak 14000 sustain=14000 burst=32 qos=low : Interface ga02f0 traffic_shape=hshq : PVC ga02f0 15/511      # to GRF-2 home network router </pre>                                                                                                                                                                                                                                                                                                                                                                                |
| Save configuration                                                           | <pre>\$ grwrite</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Reload configuration                                                         | <pre> \$ grrreset 2; grrreset 3; \$ kill &lt;fred_process_id&gt; </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Verify interfaces                                                            | <pre>\$ ping 204.101.7.150</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Configure the home network router GRF

This table lists the configuration steps for GRF-2 as a home network router.

| Action                                  | Configure the home network router GRF-2                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Configure interfaces on GRF-2:          | <pre>/etc/grifconfig.conf  # Card 3 ga00f0 10.205.1.133          # pvcr of ATMP circuit from home network router # Card 4 ge034 10.20.1.133          # as home network router</pre>                                                                                                                                                                                                         |
| Configure ATM interfaces:               | <pre>/etc/gratm.conf  Traffic_Shape=hshq peak 155000 sustain=155000 burst=2048 qos=high Traffic_Shape=hslq peak 155000 sustain=155000 burst=2048 qos=low Traffic_Shape=lshq peak 14000 sustain=14000 burst=32 qos=high Traffic_Shape=lspq peak 14000 sustain=14000 burst=32 qos=low : Interface ga00f0 traffic_shape=hshq # for pvcr : PVC ga00f0 15/511 proto=ipnl1c      # for pvcr</pre> |
| Enter ARP entry for ATM/PVCR interface: | <pre>/etc/grarp.conf  ga00f0 10.205.1.150 15/511</pre>                                                                                                                                                                                                                                                                                                                                      |
| Save configuration:                     | <pre>\$ grwrite</pre>                                                                                                                                                                                                                                                                                                                                                                       |
| Reload the configuration:               | <pre>\$ grreset 3; grreset 4;</pre>                                                                                                                                                                                                                                                                                                                                                         |
| Verify interfaces:                      | <pre>\$ ping 10.20.1.133 \$ ping 10.205.1.133</pre>                                                                                                                                                                                                                                                                                                                                         |
| Add routes in /etc/gated.conf:          | <pre>static{ : 10.20.2.0 gateway 10.205.1.150 retain; }</pre>                                                                                                                                                                                                                                                                                                                               |
| Save configuration:                     | <pre>\$ grwrite</pre>                                                                                                                                                                                                                                                                                                                                                                       |
| Add routes:                             | <pre>\$ gdc reconfig</pre>                                                                                                                                                                                                                                                                                                                                                                  |
| Verify routes:                          | <pre>\$ netstat -rn   grep 10.20.2.0</pre>                                                                                                                                                                                                                                                                                                                                                  |

At this point you can perform a **traceroute** on the mobile node Caller:  
\$ traceroute 10.20.1.20



## Monitoring ATMP activity on the GRF

This section shows **maint**, **netstat -i**, **grstat**, and **grfr** display and configuration commands.

### maint commands

Two **maint** commands, **maint 70** and **maint 73**, provide useful information about ATMP.

To use **maint** commands, first you start the **grmb** program on the target media card that you want to run the **maint** command on. From the CLI prompt or the UNIX shell, enter **grmb**:

```
grmb
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:

```
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 links associated with a media card.

The **State** column also reports configuration status per home agent and home network link.

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/DLCI** is the slot, port, and DLCI number of the dedicated link to the home network, the **pvcatmp**
- **State** indicates configuration status depending upon the type of card:
  - a card acting as the home agent (usually Ethernet)  
On the home agent card, a state of **HomeAgent** indicates the Ethernet card is properly configured in the `/etc/aitmd.conf` file as a home agent.
  - a card providing the frame relay link (**pvcatmp**) to a home network (HSSI)  
On the HSSI card linked to the home network, **State** can have three values:  
**Circuit**=the **pvcatmp** link is properly configured in `/etc/grfr.conf`, but the home agent is not set up in `/etc/aitmd.conf`.  
**HomeAgent** = the home agent is properly configured in `/etc/aitmd.conf`, but the **pvcatmp** link is not configured in `/etc/grfr.conf`.  
**Cir, HomAg** = the **pvcatmp** circuit and the home agent are both configured in their respective files
- **Address** is the IP address of the associated home agent (**atmp0**)

### *maint 70 examples*

In the first example, the state column shows two pieces of information:

- the HSSI card has a circuit to the home network, a pvcatmp
- the home agent is configured in the `/etc/aitmd.conf` file

```
GR 2> maint 70
[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/DLCI State Address Rx/BRx/Tx/BTx
[RX] ----- -
[RX] 0 02:0:160 Cir,HomAg 10.9.9.9 0/0/0/0
```

The next example shows that the card only has a home agent configured. The circuit is shown configured on the card that actually has that circuit. So, in a correctly operating system, "HomeAgent" is all you will see on the Ethernet card, while "Cir,HomAg" is what you will see on the HSSI or ATM card with the link to the home network.

```
[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/DLCI State Address Rx/BRx/Tx/BTx
[RX] ----- -
[RX] 0 06:0:160 HomeAgent 10.9.9.9 0/0/0/0
```

This example shows how the table looks when the circuit is configured in `/etc/grfr.conf`, but the home agent is not set up in `/etc/aitmd.conf` or if the tunnel manager is not running.

```
[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/DLCI State Address Rx/BRx/Tx/BTx
[RX] ----- -
[RX] 0 02:0:160 Circuit 10.9.9.9 0/0/0/0
```

## Display tunnel information

The **maint 73 tunnel\_number** command shows tunnel information.

Obtain the tunnel number using the **maint 70** command shown above. The tunnel number is the entry under FRT-Index, in this case, 1.

```
GR 2> maint 73 1
GR 2> [RX]
[RX] Mobile node tree list for home network index 1
[RX] Mobile Node /Mask Flags Foreign Agent Tunnel Id Slot:Port:DLCI
[RX] 10.4.0.0 /16 0 => 206.146.164.5 0x00000503 2: 0:160
```

This **maint 73 1** command shows a tunnel for the mobile node using address 10.4.0.0, 16 netmask bits, connecting to a foreign agent at address 206.146.164.5. The tunnel ID is 0x503. The ATMP gateway circuit (pvcatmp) is on slot 2, port 0, DLCI 160.

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, in this case, 0x503
- slot, port, and DLCI number of the dedicated link to the home network (pvcatmp)

## netstat -i command

To verify ATMP configuration of GRF home agent addresses, use the **netstat -i** command. In this sample, 221.1.1.1 is the first atmp0 IP address shown.

```
super> netstat -i
```

| Name  | MTU  | Network     | Address           | Ipkts  | Ierrs | Opkts  | Oerrs |
|-------|------|-------------|-------------------|--------|-------|--------|-------|
| de0   | 1500 | <link1>     | 00:c0:80:86:16:e2 | 77838  | 0     | 31725  | 0 209 |
| de0   | 1500 | 206.146.160 | carol             | 77838  | 0     | 31725  | 0 209 |
| rmb0  | 616  | <link2>     | 00:00:00:00:00:00 | 219891 | 13    | 128976 | 0 0   |
| rmb0  | 616  | <GRIT>      | 0:0x40:0          | 219891 | 13    | 128976 | 0 0   |
| lo0   | 1536 | <link3>     |                   | 3406   | 0     | 3406   | 0 0   |
| lo0   | 1536 | <GRIT>      | 0:0x48:0          | 3406   | 0     | 3406   | 0 0   |
| atmp0 | 1536 | <link4>     |                   | 0      | 0     | 0      | 0 0   |
| atmp0 | 1536 | 221.1.1.1   | 2,0,100,0         | 0      | 0     | 0      | 0 0   |
| atmp0 | 1536 | 0/32        | 221.1.1.1         | 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   |
| atmp0 | 1536 | 203.1.3.1   | 0,1,401,0         | 7861   | 0     | 7861   | 0 0   |
| atmp0 | 1536 | 0/32        | 203.1.3.1         | 7861   | 0     | 7861   | 0 0   |
| atmp0 | 1536 | 221.1.2.1   | 2,0,201,0         | 0      | 0     | 0      | 0 0   |
| atmp0 | 1536 | 0/32        | 221.1.2.1         | 0      | 0     | 0      | 0 0   |
| atmp0 | 1536 | 221.1.2.2   | 2,0,202,0         | 24334  | 0     | 24334  | 0 0   |
| atmp0 | 1536 | 0/32        | 221.1.2.2         | 24334  | 0     | 24334  | 0 0   |
| atmp0 | 1536 | 221.1.2.3   | 2,0,203,0         | 4819   | 0     | 4819   | 0 0   |
| atmp0 | 1536 | 0/32        | 221.1.2.3         | 4819   | 0     | 4819   | 0 0   |
| atmp0 | 1536 | 221.1.2.4   | 2,0,204,0         | 0      | 0     | 0      | 0 0   |
| atmp0 | 1536 | 0/32        | 221.1.2.4         | 0      | 0     | 0      | 0 0   |

For each home network, two lines are displayed. From the sample above, here is the first line of the pair for atmp0 221.1.2.4:

```
atmp0 1536 221.1.2.4 2,0,204,0 0 0 0 0
0
```

For atmp0 entries, the fourth column includes the set of numbers from /etc/aitmd.conf that describe the circuit for the home agent interface: card in slot 2, port 0, DLCI (s0) 204. The last number is for s1, currently not used and always 0.

Another way to determine the slot and port number for an address is to use **maint 70** on the HSSI card:

```
GR 2> maint 70
[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/DLCI State Address Rx/BRx/Tx/BTx
[RX] -----
[RX] 0 02:0:160 Cir,HomAg 10.9.9.9 0/0/0/0
```

## Using grstat commands

The **grstat** command has display commands that return useful information about ATMP links on HSSI and ATM OC-3c media cards.

*Look at IP packet counts per logical interface:*

```
grstat ipstat ga00f0
ga00f0
 ipstat
 count description
 7853 total packets received
 7853 packets forwarded normally
```

*Look at IP packet counts per card:*

```
grstat ipstat 3
card 3 (8 interfaces found)
 ipstat totals
 count description
 622207409 total packets received
 311099718 packets dropped
 7969 packets forwarded normally
 4 packets forwarded to the RMS
 311099718 packets ATMP decapsulated
```

*Look at packets dropped per media card*

```
grstat ipdrop 3
card 3 (8 interfaces found)
 ipdrop totals
 count description
 311099718 ATMP err: bad GRE header
```

*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
grstat ipdrop ge031
ge031
 ipdrop
 count 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
```

Refer to the *GRF Reference Manual* or check the **grstat** man page for more information about the **grstat** command.

## Using grfr commands

The **grfr** command has display commands that return useful information about ATMP on HSSI Frame Relay links.

### 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 R-Packets 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
```

```

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: | S:       |
|--------------|------|---------|---------|-----------|-------|-------|-------|-------|-------|----------|
| 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 | 56K | 56K | 56K | Active | 0:0:0    |
| Headroom-pub | 0    | 0    | 200  | Route  | 56K | 56K | 56K | Inact  | NO-ISIS  |
| 0:1:0        | 0    | 1    | 0    | Switch | 56K | 56K | 56K | Active | 0:1:0    |
| wg-atmp      | 0    | 1    | 401  | ATMP   | 30M | 30M | 30M | Active |          |
| wg-route     | 0    | 1    | 402  | Route  | 22M | 22M | 56K | Active | NO-ISIS  |
| 2:0:0        | 2    | 0    | 0    | Switch | 56K | 56K | 56K | Active | 2:0:0    |
| HN1-eth-dumm | 2    | 0    | 32   | Route  | 56K | 56K | 56K | Active | NO-ISIS  |
| HN1-eth      | 2    | 0    | 100  | ATMP   | 56K | 56K | 56K | Active |          |
| stress:201   | 2    | 0    | 201  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:202   | 2    | 0    | 202  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:203   | 2    | 0    | 203  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:204   | 2    | 0    | 204  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:205   | 2    | 0    | 205  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:206   | 2    | 0    | 206  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:207   | 2    | 0    | 207  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:208   | 2    | 0    | 208  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:209   | 2    | 0    | 209  | ATMP   | 30M | 30M | 30M | Active |          |
| stress:210   | 2    | 0    | 210  | ATMP   | 30M | 30M | 30M | Active |          |
| 2:1:0        | 2    | 1    | 0    | Switch | 56K | 56K | 56K | 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   | 56K | 56K | 56K | 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
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 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 **grfr -c ccp** to add or **grfr -c crp** 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 DLCI, slot, and link must be specified:

```
grfr -c ccp -f /etc/grfr.conf -i 606 -s 13 -l 0
```

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** command is sufficient. Specify the target DLCI to be disabled:

```
grfr -c crp -i 600 -s 13 -l 0
```

Here is the response:

```
PVC slot 13, link 0, dlci 600 deleted
```

# ATM OC-12c Configuration

# 10

Chapter 10 provides information needed to configure the ATM OC-12c media card.

The ATM OC-12c media card provides one bi-directional interface that supports 220 logical interfaces. The GRF can be configured in point-to-point or point-to-multipoint ATM topologies with either switches or hosts.

This chapter contains:

|                                                      |       |
|------------------------------------------------------|-------|
| ATM OC-12c functions .....                           | 10-2  |
| Physical and logical ATM OC-12c interfaces .....     | 10-6  |
| Configuration file and profile overview .....        | 10-9  |
| Assign IP addresses - grifconfig.conf .....          | 10-10 |
| Specify ATM card parameters – Card profile .....     | 10-11 |
| Change run-time code (optional) – Load profile ..... | 10-14 |
| Change dump default (optional) – Dump profile .....  | 10-16 |
| Configuring PVCs .....                               | 10-18 |
| PVC example .....                                    | 10-19 |
| Traffic shaping .....                                | 10-20 |
| Traffic shape names .....                            | 10-24 |
| Setting output rates .....                           | 10-25 |
| maint commands for ATM OC-12c media cards .....      | 10-26 |

## ATM OC-12c functions

### Large route table support

ATM OC-12c media card software maintains route tables containing up to 150K entries, and hardware support for full table lookups.

### Selective packet discard

Selective packet discard on the ATM OC-12c card ensures 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 preference to dynamic routing packets. The ATM OC-12c card begins selective discard when the number of available buffers falls below 128.

### 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.

Edit the `/etc/gratm.conf` file and specify `proto=isis` or `proto=llc` fields in the logical interface’s PVC statement. Use the `proto=isis` field when IS-IS is the only protocol to run, specify `proto=llc` if IP and IS-IS both will run. The protocol field enables an interface to accept IS-IS packets. Here is an example:

```
pvc ga030 0/40 proto=isis traffic_shape=high_speed_high_quality
```

Refer to the *Introduction to IS-IS* chapter for more information.

### 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).

### MTU

The maximum transmit unit for an ATM OC-12c packet is 9180 bytes, it cannot be set to a higher value.

## 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 support

The ATM OC-12c card supports raw ATM mode to other ATM OC-12c cards in the same GRF chassis.

It does not support raw mode from ATM OC-12c cards across the GRF backplane to ATM OC-3c cards in the same chassis. This means that you cannot configure the GRF to operate as an 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.

It does support raw mode between an ATM OC-12c card and a ATM OC-3c card interconnected through an ATM switch.

## SDH and SONET modes

The physical layer can be set to either SONET or SDH mode, SONET is the default.

Mode is a parameter in the `gratm.conf` file and is specified in the “*Signaling*” line. The example below shows mode specified as SDH and then SONET:

### *gratm.conf*

```
Signaling parameters
Signaling card=5 connector=bottom protocol=UNI3.0 mode=SDH
```

or

```
Signaling card=5 connector=bottom protocol=UNI3.0 mode=SONET
```

Notice that mode and signaling are configured per physical interface (port), and not on a logical interface.

## Inverse ARP

The ATM OC-12c media card supports ATM inverse ARP.

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
```

The GRF takes the ARP entry learned via ATM inverse ARP as opposed to the one in the `grarp.conf` file. If no ARP entry exists for a given PVC when **grarp** is run, the ARP entry given in the `grarp.conf` file is accepted.

## Setting SUNI receive clock

The SUNI component has a receive and a transmit clock. The transmit clock is *always* at the SUNI's internal setting. However, the receive side clock setting can be toggled between internal (the SUNI's own clock) and external, to use the clock of the transmitting node.

## Broadcasting

The ATM media card does not use standard broadcast IP group address.

Broadcast addresses are entered in the *Service section* of the `gratm.conf` file, and the card's transmit interface routes broadcast datagrams using the `type=bcast` Service in `gratm.conf`. Here is an example:

```
Broadcast Service info
Service name=bcl type=bcast addr=204.221.157.121
 addr=
 addr=
#
```

## Selective packet discard

Selective packet discard can be enabled on the ATM OC-12c 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 percent, 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 percentage threshold is set to 100 percent, all buffers are reserved for dynamic routing packets, no buffers are available for data packets. Any intermediate value indicates the percentage of buffers reserved for dynamic routing packets.

The selective discard mechanism begins to drop non-dynamic routing packets when the percentage of free transmit buffers is less than the user-defined percentage 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 ATM OC-12c 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.

### *Example*

Given an ATM OC-12c card installed in slot 2, this example shows where to set the threshold in the CLI:

```
prompt> read card 2
CARD/2 read

prompt> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0

prompt> set tx-cong=7
prompt> write
prompt> grreset 2
```

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
```

## Physical and logical ATM OC-12c interfaces

This diagram shows the organization of physical and logical ATM interfaces on the ATM OC-12c media card:

### Media card:

| Physical interface 0<br>(center) | Logical interfaces | VPI / VCI |          | Total # of active VCs |
|----------------------------------|--------------------|-----------|----------|-----------------------|
|                                  | 0 – ff (range)     | 0         | 0 – 1023 | 1408                  |
|                                  |                    | 1 – 3     | 0 – 127  |                       |

Figure 10-1. ATM OC-12c physical and logical interfaces

## Physical 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

Logical interfaces provide a simple way of mapping many IP addresses onto a single ATM port. A logical interface serves as the connection between ATM and IP. Each logical interface is assigned a unique IP address in `grifconfig.conf`.

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

VPI/VCI are assigned to logical interfaces in `gratm.conf`, and provide the bridge between ATM and IP.

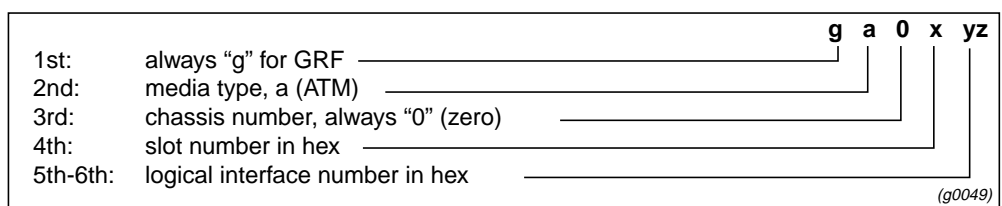


Figure 10-2. Interface name structure for an ATM logical interface

### Interface name *ga0yz*

The generic form of an ATM interface name is: `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



- the “z” logical interface number ranges from 0 through ff to identify each one of the 220 logical interfaces that can be configured per ATM card

Logical interfaces on connector 0 range from 0 to ff, you can assign 240 logical interfaces.

## Virtual circuits

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 *active* virtual circuits (VCs) as defined in the ATM Forum UNI 3.0 specification. Each virtual circuit has an associated IP address.

## Virtual paths

A virtual path connects two end stations which may be separated by one or more network devices such as a router or switch. A path consists of one or more virtual circuits.

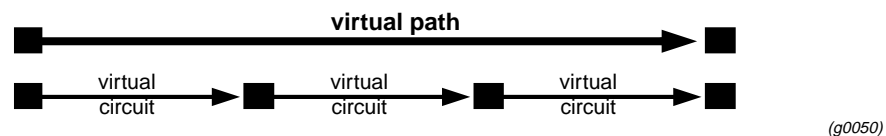


Figure 10-3. Components that form a virtual path

The ATM OC-12c media card supports four virtual path identifiers, VP 0–3.

Virtual path identifiers (VPIs) 0 through 3 are available for configuration use. VPIs are assigned in the `/etc/gratm.conf` file.

## VCIs

Virtual circuit identifiers (VCIs) name virtual circuits. VCIs are assigned in the `gratm.conf` file.

On VPI 0: VCI 0 through VCI 1023 can be used

On VPIs 1-3: VCI 0 through VCI 127 can be used

**Note:** Virtual circuits 0-31 on each VPI are reserved for use by UNI signaling.

## VPI/VCI

In the `/etc/gratm.conf` file, VPI/VCI specifies the (numeric) Virtual Path Identifier and Virtual Circuit Identifier of the virtual circuit, separated by a slash (/). For example, 0/126.

VPI values           = 0–3  
VCI values           = 0–127, 0–1023 depending upon the virtual path

## Permanent virtual circuits

Permanent virtual circuits (PVCs) are created statically. PVCs are configured in `gratm.conf`.

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`.

The VPI/VCI must only be unique per physical ATM port, not per card. The following VPI/VCI are supported:

VPI 0:VCI 0 . . . 1023  
VPIs 1–3:VCI 0 . . . 127  
VCI 0–31 on each VPI are reserved for UNI signaling.

On a given physical interface, 1408 VCs can be active at any one time.

## Switched virtual circuits

Switched virtual circuits (SVCs) are not supported on the ATM OC-12c card.

## Verifying an ATM configuration

**maint** commands enable you to verify an ATM configuration. They are described at the end of this chapter and some examples are provided.

Operations include:

- verify VC configuration
- examine active VCs and traffic on them
- display ARP servers registered with
- display known ARP entries

## 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:

```
greset <slot_number>
```

## ***Configuration file and profile overview***

These are the configuration tasks and options for ATM OC-12c media cards:

### ***1. Assign IP address to each logical interface***

Edit `grifconfig.conf` to identify each logical interface by assigning:

- an IP address
- the GRF interface name
- a netmask, as required
- a destination or broadcast address, as required
- an MTU, if needed

Create a separate entry in which to assign an ISO address if the logical interface will run IS-IS.

### ***2. Specify ATM card parameters in the Card profile:***

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard percentage in the `spd-tx-thresh` field
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

### ***3. Configure PVCs in `/etc/gratm.conf`***

Configure PVCs in the `/etc/gratm.conf` file.

### ***4. 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 ATM card.

If you want to change the run-time code in one ATM card (per physical interface), make the change in the Card profile, in the `load` field.

### ***5. 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 zero (0) which actually stores two dumps per day (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 (per interface), make the change in the Card profile, in the `dump` field.

## Assign IP addresses - *grifconfig.conf*

Edit the `grifconfig.conf` file to assign an IP address to each logical ATM interface. You also can provide other information about the logical IP network to which that interface is physically attached, or specify a different MTU in the `arguments` field, for example.

When you configure a logical interface on a point-to-point media such as ATM, enter the destination IP address in the `broad_dest` address field. An entry in the `broad_dest` address field for an ATM interface 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.

The optional `arguments` field is currently used to specify MTU values on a logical interface basis. 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.

Each logical interface is identified in `grifconfig.conf` as to its:

- interface name, `ga0yz` (always lower case)
- Internet address
- netmask
- broadcast/destination address
- `arguments` field (optional)

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

```
#name address netmask broad_dest arguments
ga030 xxx.xxx.xxx.xxx 255.255.255.0 - mtu 9100
ga0d2a yyy.yyy.yyy.yyy 255.255.255.0 zzz.zzz.zzz.zzz
```

This example sets an IP address for logical interface 0 on the ATM card in slot 3 (upper physical interface), and specifies a lower than default MTU value. A dash is used as a placeholder for the broadcast address.

The second entry sets an IP address for logical interface 42 on the ATM card in slot 13 (lower physical interface), and specifies a destination address.

## ***Specify ATM card parameters – Card profile***

Set specific ATM card configuration parameters at the Card profile. The fields to set are:

- OPTIONAL: specify ICMP throttling settings
- OPTIONAL: specify selective packet discard percentage in the `spd-tx-thresh` field
- OPTIONAL: change run-time binaries
- OPTIONAL: change dump variables

Fields you may want to set are in bold, default values are shown here.

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 }
```

### ***Specify selective packet discard percentage***

Here is where you enter the congestion threshold percentage:

```
super> list config
word = 0
ping = 1
reset = 1
init = 4
panic-reset = 0
spd-tx-thresh = 0
```

Change the threshold to 6 with this series of commands:

```
super> set spd-tx-thresh = 6
super> write
CARD8/read
```

### *Specify ICMP throttling*

You can specify ICMP throttling changes for this ATM card in these settings.

Refer to Chapter 1 for an explanation of each field or do a `set <field-name>?` for a brief description. Default values are shown:

```
super> list ic
echo-reply = 10
unreachable = 10
redirect = 2147483647
TTL-timeout = 10
param-problem = 10
time-stamp-reply = 10
```

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/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.

### *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 ATM 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_ATM12c
super> write
CARD/8 written
```

### *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 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
```

## Change run-time code (optional) – 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** ATM or ATM/Q cards.

Here is the path, default settings are shown:

```
super> read load
LOAD read

super> list
hippi = { " " N/A on 0 1 < { 1 /usr/libexec/portcards/xlxload.run N/A } { +
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/Q card settings:

```
super> list 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 = < >
```

To execute different run-time code on the receive side of the ATM/Q card, replace `/usr/libexec/portcards/atmq_rx.run` with the path to the new code.

```
super> set rx-path = /usr/libexec/portcards/testatm-12.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> list 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 = < >

super> set enable-boot-seq = on
super> write
LOAD written
```

## ***Change dump default (optional) – Dump profile***

Global 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 accommodates the default setting of zero (0) which actually stores two dumps per day (the current dump and the first dump of the day). Use caution if you change the recommended default.

Here is the path, default settings are shown:

```
super> read dump
DUMP read

super> list
hw-table = <{hippi 20 var 0} {fddi 20 /var/portcards/grdump 2} {rmb 20 /+
dump-vector-table = <{2 fddi "FDDI default dump vectors" <{1 "fddi core +
config-spontaneous = off
keep-count = 0
```

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/Q settings:

```
super> list hw-table
hippi = { hippy 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 cards.

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/Q 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 4096 } { 2 "ATM-12 +

super> list seg
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 }

super> list 1
index = 1
description = "ATM-12 SDRAM memory"
start = 16777216
length = 4096

super> list s 7
index = 7
description = "Rx CTRL Regs"
start = 50331648
length = 16
```

# Configuring PVCs

## Overview

To configure PVCs, use a UNIX editor to edit the `.conf` files. Open the UNIX shell:

```
prompt> sh
```

### *grifconfig.conf*

- assign IP address to the logical interface

### *grarp.conf*

- supply IP-to-physical address mapping information for ARP service  
Put an entry into `grarp.conf` ONLY if the remote destination does NOT support inverse ATM ARP. (The GRF supports inverse InATMARP.)

### *gratm.conf*

- *Traffic Shaping section*  
Set traffic shaping name and quality of service parameters, use any string, set a name for each type of service that will be assigned
- *Signaling section*  
Set **protocol=NONE** on the signaling entry for the appropriate card and connector combination
- *Interface section*  
Define traffic shaping profile for the logical interface to which the media card's PVCs are assigned
- *PVC section*  
Specify characteristics for each PVC, including:
  - assigned logical interface name
  - VPI/VCI
  - protocol supported
  - AAL used
  - assigned traffic shaping profile (only if it would be different from the profile given above to the logical interface in the *Interface section*)

## OC-12c PVC parameters

- `proto=vc` not legal for PVC configurations
- in `proto=raw` configurations, `dest_if` must be another ATM OC-12c card because raw connections between OC-12 and OC-3c cards are not supported
- only `input_aal=5` is supported

Templates of these configuration files are in the *GRF Reference Guide*.

Traffic shaping is discussed in detail later in this chapter.

## PVC example

This example configures a PVC with the following attributes:

- resident on single physical interface, ATM OC-12c card is in slot 3
- on logical interface 99 (hex=99)
- connects to an endpoint that does not support inverse ARP

The ATM OC-12c card supports inverse ATM ARP (InATMARP). This protocol is used to determine the IP address at the other end of a PVC.

- VPI/VCI, in this case 0/32
- runs IP protocol, AAL-5 (default)
- the local IP address is 192.0.130.1
- the remote IP address is 192.0.130.111

### *grifconfig.conf*

```
Interface Internet broadcast/destination
name address netmask address arguments
ga0399 192.0.130.1 255.255.255.0
```

Here is the ISO address entry and format if the PVC will run IS-IS:

```
<interface-name> <iso-address> <iso-area> - iso
ga030 49.0000.80.3260.3260.00 49.0000.80 - iso
```

### *grarp.conf*

```
[ifname]hostnamephys_addr [temp] [pub] [trail]
#
ga0399192.0.130.1110/32
```

### *gratm.conf*

```
#
Traffic shaping parameters
#
Traffic_Shape name=low_speed_high_quality \ peak=15000 qos=high
Traffic_Shape name=high_speed_low_quality \ peak=15000 qos=hlow

Interfaces
#
Interface ga0399 traffic_shape=low_speed_high_quality

PVC's
#
PVC ifname VPI/VCI proto=ip|raw|vc|ipnllc|isis|isis_ip
[input_aal=3|5|NONE] \
[dest_if=logical_if [dest_vc=VPI/VCI]] [traffic_shape=shape]
#
PVC ga0399 0/32 proto=ip traffic_shape=high_speed_low_quality
```

The *Traffic Shaping* section includes many different `Traffic_Shape name=` entries as shown above, one for each type of service that could be assigned.

## ***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 profiles are assigned PVCs, 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 GRF; 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.

The GRF ATM card expects to forward cells as fast as it can. However, the next node in the virtual circuit connection may not be able to handle this rate of traffic. Traffic shaping provides a measure of control over the rate at which cells are transmitted.

## **Profile parameters**

An ATM OC-12c traffic shaping profile involves four parameters. Three are described in UNI 3.0/3.1 to effectively manage the timing of the transmission of ATM cells over SONET OC-12c media.

The parameters are set in `gratm.conf` and include:

- peak cell rate, in kilobits/second (PCR)
- Sustained cell rate, in kilobits/second (SCR)
- maximum burst size, in cells (MBS)

The fourth parameter is not in the UNI specification but is also set in `gratm.conf`:

- priority, either high or low (PRI)

**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 times out while trying to move data at 19 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 them to go out. If the mismatch in speeds is large, packets on the faster incoming network eventually will be lost, and retransmission will be required.

## Rate queues and QOS

Each different peak rate you specify automatically creates a rate queue. The ATM OC-12c card allows sixteen rate queues per physical ATM interface. Eight queues are high-priority, eight are low-priority.

Each rate queue has an associated counter. If you specify a 17th peak rate, an error message reminds you of the limit. The error is generated when you try to set the ninth maximum bit rate in either the high or low QOS. If you have eight high and zero low QOS, and try to implement a ninth high QOS, you get the error. The limit is based on eight high and eight low QOS.

Quality of Service (QOS) has two options, high and low. You can have four high QOS and four low. If no QOS is defined, the default is high. The settings of high and low indicate the type of rate queue group to select. There are eight rate queues, 0–3 are high QOS, 4–7 are low QOS. The rate queues equate to priority. The first four queues defined for each high priority and each low priority become the assigned rate queues. You can then assign maximum, sustained, and burst rates to each of the rate queues. High QOS queues get serviced before low. Only four different peak cell rates can be assigned per QOS.

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 VCs are active, VCs assigned to low-priority queues may not get serviced.

### *Priority*

Priority is a characteristic of a rate queue. A high-priority (for access) queue means a Quality of Service (qos) equal to high. A low-priority (for access) queue equates to a Quality of Service (qos) = low. qos is specified in `gratm.conf` as part of the traffic shaping profile.

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.

### *Queueing*

Queueing among the VCs assigned to the same traffic shape is round-robin. Between VCs with different traffic shapes, a priority queueing scheme gives highest priority to those with the lower sustained or peak rates.

## 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 back off 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. On ATM OC-12c cards, the largest burst allowed is 256 cells.

Maximum burst size operates only as multiples of 32 cells. The smallest burst size allowed is 32 53-byte cells (13568 bits), the largest burst allowed is 2048 cells. Software rounds the requested burst size downward to the next such multiple, or 32, whichever is greater.

## 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.

Can the VC exceed 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  $(N \times \text{SCR})$  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.

## ***Traffic shape names***

Peak cell rate, sustained cell rate, maximum burst size, and priority (high/low qos) 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:

```
Traffic_Shape name=high_speed_high_quality \
 peak=15500 sustain=15500 burst=2048 qos=high
```

(In `gratm.conf`, a backslash (\) is used to divide a single long line of characters.)

The name given above specifies the best possible service and access to bandwidth resources. The following name specifies the minimum service possible:

```
Traffic_Shape name=lowest_speed_lowest_quality \
 peak=15500 burst=32 qos=low
```

Sustainable rate defaults to peak cell rate when it is not specified.

PVCs and logical interfaces are individually assigned a specific `Traffic_Shape` name.

You can create as many `Traffic_Shape` names as you need, but you can specify only eight different peak rate queues. The eight peak rates are combined with different sustained rates and burst sizes to create more than eight `Traffic_Shape` names.

Peak rate is the only required parameter in a `Traffic_Shape` name. If you do not specify sustained rate and burst size, they default to match the peak rate. Another optional parameter is Quality of Service (qos). Quality of Service defaults to high priority.

## **ATM OC-12c 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 256 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.

Traffic shaping parameters are specified in the *Traffic shaping section* of `gratm.conf`. Refer to the *GRF Reference Guide* for a copy of the `gratm.conf` file.

## Setting output rates

### Sending at a controlled rate

To ensure that the transmission of cells does not exceed a specific rate, create a traffic shape specifying that peak rate in `gratm.conf`.

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).

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. 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.

### Changing a rate queue

Values in a rate queue cannot be changed on the fly. Changes must be made in the `gratm.conf` file and the ATM media card rebooted.

## ***maint commands for ATM OC-12c media cards***

The ATM OC-12c **maint** commands display a range of information about the media card.

### **Preparing to use maint**

To use the **maint** commands, you must use the **grrmb** command and bring up the GR > prompt. First, switch to the **maint** prompt. Enter:

```
grrmb
```

The new prompt appears:

```
GR 66>
```

Then change the prompt port to the ATM media card you are working with. For example, if you are working with a card in slot 15, enter:

```
GR 66> port 15
```

This message is returned along with the changed prompt:

```
Current port card is 015
GR 15>
```

To leave the **maint** prompt, enter **quit**.

### **Lists of maint commands**

The set of ATM OC-12c **maint** commands has been revised and some have new numbers. In the list below, old numbers are in parenthesis. To see a list of **maint** commands, enter: **maint 1**

```
GR 15> maint 1
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 vcil]
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 card version numbers

The **maint 2** command displays g:

```
GR 0> maint 2
GR 4> Code Version: A1_4_3
 Compiled in: /volatile/jkr/A1_4_3/atm-12,
 on: Mon Nov 24 15:27:27 CST 1997.
 GRIPv4 Library Version: 1.4.3, Compiled on Sat Nov 22 14:58:49 CST 1997.
 Board rev: 3 FPGA rev: 1
 DIP switch settings: 1=OFF 2=OFF 3=OFF 4=OFF
```

## 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 0> maint 3
GR 0>
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
GR 0>
```

## 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 0> maint 4
GR 0>
 VC statistics
 output
 rq pkts errs aal pro input
 CID Vpi Vci rq pkts errs 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 ARP information

To look at ARP addresses per VPI/VCI, use **maint 8**:

```
GR 0> maint 8
GR 0>
 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 0> maint 9
GR 0>
 ARP-S NSAPA STATE

00: 0X47.0005.80ffe1000000f21513eb.0020481513eb.00 0/51

 Not Configured

 Not Configured
GR 0>
```

## Display memory usage

The **maint 10** command displays memory information about the 4-byte table memory (for media card tables) and the 616-byte table memory buffers (for Combus usage). The number of total and free buffers are provided.

```
GR 0> maint 10
GR 0>
 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
GR 0>
```

## 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:
0897645539 Automatically forwarded by hardware
0000192842 Handled by software as IS-IS
```

## 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 sustainable cell rate, and **FMBS** is the forwarding maximum burst size.

```
GR 0> maint 13
IF VPCI 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
00 0/045 pvc 622080 622080 000255 IP pt-pt
00 0/046 pvc 622080 622080 000255 IP pt-pt
00 0/047 pvc 622080 622080 000255 IP pt-pt
00 0/048 pvc 622080 622080 000255 IP pt-pt
00 0/049 pvc 622080 622080 000255 IP pt-pt
00 0/050 pvc 622080 622080 000255 IP pt-pt
00 0/051 pvc 622080 622080 000255 IP pt-pt
00 0/052 pvc 622080 622080 000255 IP pt-pt
00 0/053 pvc 622080 622080 000255 IP pt-pt
00 0/054 pvc 622080 622080 000255 IP pt-pt
00 0/055 pvc 622080 622080 000255 IP pt-pt
00 0/056 pvc 622080 622080 000255 IP pt-pt
00 0/057 pvc 622080 622080 000255 IP pt-pt
00 0/058 pvc 622080 622080 000255 IP pt-pt
00 0/059 pvc 622080 622080 000255 IP pt-pt
```

## ATM OC-12c Configuration

### *maint* commands for ATM OC-12c media cards

---

```
00 0/060 pvc 622080 622080 000255 IP pt-pt
00 0/061 pvc 622080 622080 000255 IP pt-pt
00 0/062 pvc 622080 622080 000255 IP pt-pt
00 0/063 pvc 622080 622080 000255 IP pt-pt
00 0/064 pvc 622080 622080 000255 IP pt-pt
00 0/065 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 0> 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**



# IP Packet Filtering

# 11

Chapter 11 describes how to configure IP packet filtering for GRF media interfaces.

This chapter contains these topics:

|                                                        |       |
|--------------------------------------------------------|-------|
| GRF filtering implementation .....                     | 11-2  |
| Filters .....                                          | 11-3  |
| Rules .....                                            | 11-3  |
| Filters for service ports .....                        | 11-5  |
| Bindings .....                                         | 11-6  |
| Packet header logging .....                            | 11-9  |
| Controlling access to the RMS .....                    | 11-12 |
| Filtering configuration process .....                  | 11-14 |
| Filtering configuration file – filterd.conf .....      | 11-15 |
| Filter grammar reference .....                         | 11-18 |
| Monitoring filtering operations – maint commands ..... | 11-21 |

## 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

## Configuration daemon

The **filterd** daemon controls filtering. Messages from **filterd** are sent to the `gr.console` log. Refer to the `filterd.conf` 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 a `filterd.conf` template is included in this chapter.

## CLI access to filterd.conf

The **gredit** command accesses the `filterd.conf` file from the command-line interface (CLI) and opens the file in the **vi** editor.

Enter:

```
super> gredit filterd
```

## maint commands

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.

## Filters

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 `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 `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:

|                 |                                  |
|-----------------|----------------------------------|
| <code>Gt</code> | to mean greater than             |
| <code>Lt</code> | to mean less than                |
| <code>Le</code> | to mean less than or equal to    |
| <code>Ge</code> | 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    | - <code>vlif &lt;number&gt;</code> ranges between 0 and 255, as in <code>vlif 0..18</code> , or <code>vlif 1,20,23..25</code> , or <code>vlif 222</code> . |
| HIPPI, SONET | - <code>vlif &lt;number&gt;</code> is always <code>vlif 0</code> since a HIPPI card has a single logical interface.                                        |
| HSSI         | - <code>vlif &lt;number&gt;</code> ranges between 0 and 255 when the framing protocol is Frame Relay, and between 0 and 1 when PPP or HDLC are running.    |
| FDDI         | - <code>vlif &lt;number&gt;</code> ranges between 0 and 3, depending upon the combination of SAS and DAS connections.                                      |
| 10/100Base-T | - <code>vlif &lt;number&gt;</code> 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.

## Media type

Use these names to specify media type:

```
hippi
fddi
atm (OC-3c and OC-12c)
hssi
ether
sonet
```

## 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.

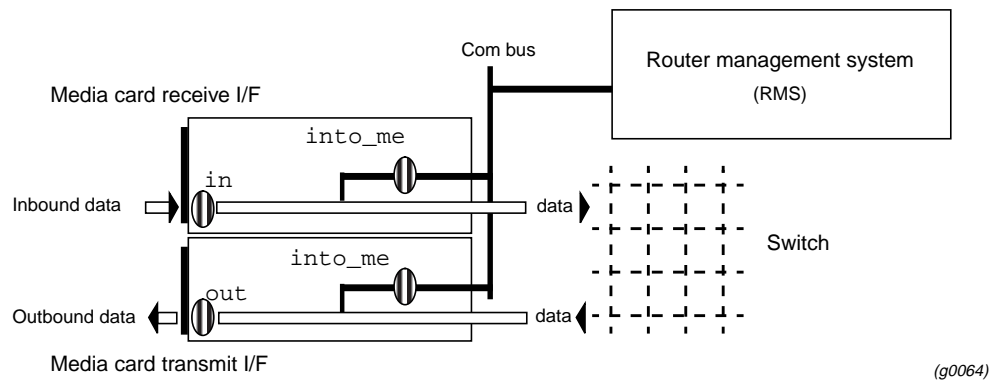


Figure 11-1. Placing filter with the direction option

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;
 }
}
```

## Actions

This statement describes the activity initiated by the filter.

Currently available actions include:

- `filter`, for filtering
- `log`, for logging of packet headers
- `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 a filtered packet is sent to a logging host. 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* - is the IP address of the target host you want to send  
logged information to

*port* - is 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* - 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

## Logging loops

A logging loop is created when a logged packet causes another logged packet to be generated, and the card processing activity is taken up with cycles of logging packets. 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.

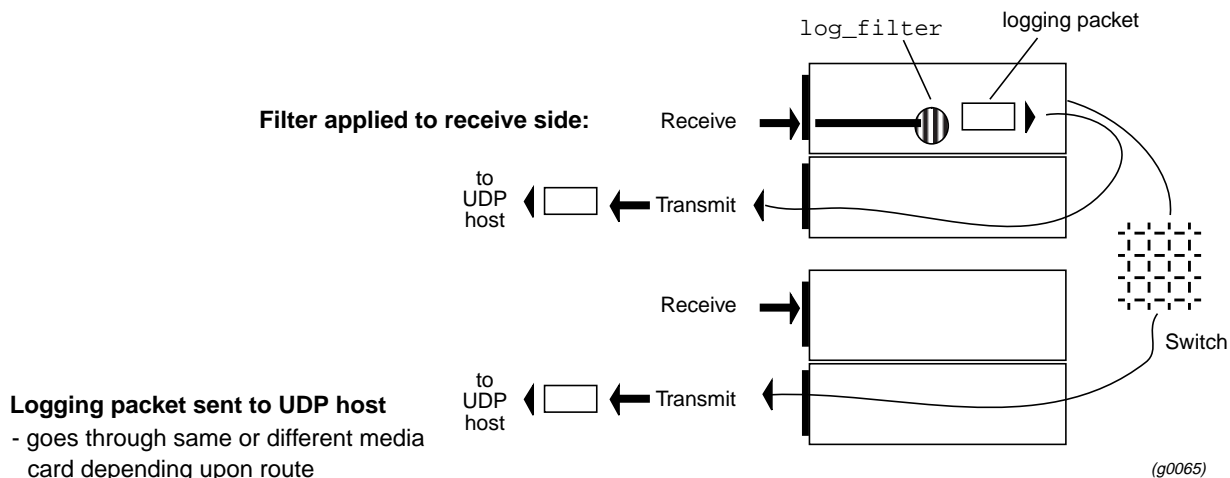


Figure 11-2. Receive-side logging filters do not loop

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.

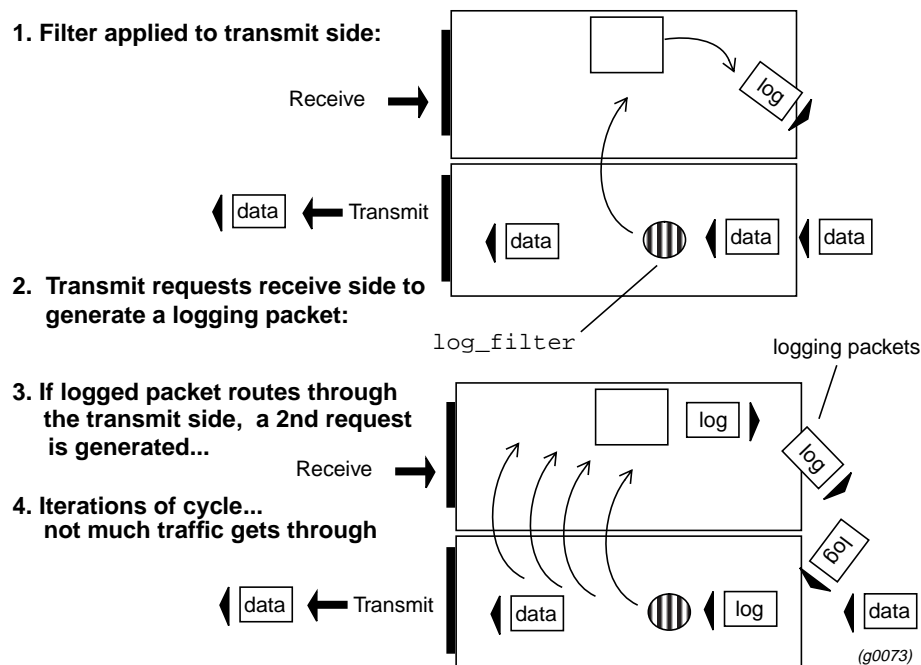


Figure 11-3. How logging loops can occur

## **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 (e<sub>0</sub>, 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 steps to create and start up GRF filtering:

- 1** Copy the `/etc/filterd.conf.template` file into `/etc/filterd.conf`
- 2** Edit `filterd.conf` to create and assign each filter:
  - create the filter(s) and specify rule(s)
  - specify the binding(s)

The template for the `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**

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.

## Filtering configuration file – filterd.conf

This is the filterd.conf file template:

Part 1 of 3

```
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 {
 to 222.222.222.1 0.0.0.0;
 ipv4protocol tcp {
 port 25;
 }
 }

 #
 # Let the consultant from somehost.somewhere.com into one of our
 # networks.
 #
 permit {
 from 221.222.222.32 0.0.0.0;
 to 222.222.222.0 0.0.0.255;
 }
}
```

(g0066)

Figure 11-4. Template for unedited filterd.conf file (part 1 of 3)

#### Part 2 of 3

```
The following is a second filter, which we will stick on our 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
 }
}
```

(g0067)

Figure 11-5. Template for unedited filterd.conf file (part 2 of 3)



Part 3 of 3

```
Once all the 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;
 }
}
```

*(goo68)*

*Figure 11-6. Template for unedited filterd.conf file (part 3 of 3)*

## ***Filter grammar reference***

This section provides the grammar of filter components. More information at this level of detail is available in the **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 ;

slot : NUMBER ;

direction_key : IN ;
 | INBOUND ;
 | OUT ;
 | OUTBOUND ;
 | ;
```

## Monitoring filtering operations – maint commands

**maint** commands are executed at the GR> prompt.

### Preparing to use maint

First, enable the GR> prompt. Enter:

```
grrmb
```

or

```
super> grrmb
```

The new prompt appears:

```
GR 66>
```

The number in the GR> prompt indicates the slot number of the card the **maint** command will act on, 66 indicates the IP switch control board. Use the **port** command to change the prompt to the media card you are working with. If you are working on a card in slot 3, enter:

```
GR 66> port 3
```

This message is returned along with the changed prompt:

```
Current port card is 03
GR 03>
```

To leave the GR> prompt, enter **quit**. You are returned to the Unix prompt.

### 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
```

## IP Packet Filtering

### Monitoring filtering operations – maint commands

---

```
[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:

```
grmb
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 **grmb** 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:

```
grmb
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
GR 05> [TX]
[TX] ActionID type status access
[TX] 00000039 filter (loaded) 0002

GR 05> port 2
Current port card is 2
GR 02> [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
GR 05> [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
GR 05> [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
GR 05> [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
```

# Introduction to GRF BGP

# 12

Chapter 12 describes the GRF-based additions to BGP and provides configuration information and examples. GateD configuration information is available in the *GRF Reference Guide*.

This chapter contains these topics:

|                                                    |       |
|----------------------------------------------------|-------|
| Introduction to BGP .....                          | 12-2  |
| BGP protocol features .....                        | 12-2  |
| Additions to BGP .....                             | 12-6  |
| BGP statement .....                                | 12-10 |
| Configuring multi-exit discriminators (MEDs) ..... | 12-12 |
| Configuring DPA attributes .....                   | 12-13 |
| Configuring route reflection .....                 | 12-14 |
| Route reflection example .....                     | 12-16 |
| Weighted route dampening .....                     | 12-19 |

# ***Introduction to BGP***

## **Exterior protocols**

Exterior protocols exchange routing information between autonomous systems (AS), and are only required when an AS must exchange routing information with another AS.

Routers within an AS run an interior routing protocol like RIP. Only those gateways that connect one AS to another AS need to run an exterior routing protocol.

## **EGP – BGP transition**

Exterior Gateway Protocol (EGP) reachability information is passed into “core” gateways (originally into ARPANET/MILNET) where the best routes are chosen and are passed back out to all connected autonomous systems. EGP assumes a hierarchical structure and is less effective as the networks move toward a less hierarchical architecture.

Border Gateway Protocol (BGP) supports non-hierarchical topologies and can be used to implement a network structure of equivalent ASs. BGP exchanges reachability information between ASs, and provides more capabilities than does EGP.

To select the best route, BGP uses path attributes that provide more information about possible routes. Path attributes can include administrative preferences based on political, organizational, or security (policy) considerations in the routing decision.

The current GRF BGP implementation supports versions 2, 3 and 4 of the BGP protocol. BGP version 3 only propagates classed network routes, and the AS path is a simple array of AS numbers. BGP 4 propagates fully general address-and-mask routes, and the AS path has some structure to represent the results of aggregating dissimilar routes.

## ***BGP protocol features***

The Border Gateway Protocol (BGP) is an exterior routing protocol used for exchanging routing information between autonomous systems. BGP exchanges routing information between multiple transit autonomous systems as well as between transit and stub autonomous systems. While BGP is related to EGP, it operates with more capability, greater flexibility, and less required bandwidth. BGP uses path attributes to provide more information about each route and, in particular, maintains an AS path. The AS path includes the AS number of each autonomous system the route transits, providing information sufficient to prevent routing loops in an arbitrary topology.

Path attributes establish administrative preferences among groups of routes, and allow flexibility in assigning route preference to achieve a variety of administrative ends.

The BGP protocol provides a high degree of control and flexibility for doing interdomain routing while enforcing policy and performance constraints and avoiding routing loops.

## **Autonomous systems**

The classic definition of an autonomous system (AS) is a set of routers under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs.

It is common for a single AS to use several interior gateway protocols and sometimes several sets of metrics within the AS. The use of the term autonomous system stresses the fact that even when multiple IGP and metrics are used, the administration of an AS appears to other ASs to have a single coherent interior routing plan and presents a consistent picture of which networks are reachable through the AS.

ASs are assumed to be administered by a single administrative entity, at least for the purposes of representation of routing information to systems outside of the AS.

A connection between two autonomous systems has two aspects:

- a physical connection

There is a shared network between the two ASs and, on this shared network, each AS has at least one border gateway belonging to that AS. In this way, the border gateway of each AS can forward packets to the border gateway of the other AS without resorting to Inter-AS or Intra-AS routing.

- a BGP connection

There is a BGP session between BGP speakers in each of the ASs, and this session communicates those routes that can be used for specific networks via the advertising AS. The BGP speakers that form the BGP connection must themselves share the same network that their border gateways share. Thus, a BGP session between adjacent ASs requires no support from either inter-AS or intra-AS routing.

At each connection, each AS has one or more BGP speakers and one or more border gateways, and these BGP speakers and border gateways are all located on a shared network. Paths announced by a BGP speaker of one AS on a given connection are taken to be reachable for each of the border gateways of the other ASs on the same shared network. Indirect neighbors are allowed.

Much of the traffic carried within an AS either originates or terminates at that AS. Either the source or destination IP address of the IP packet identifies a host on a network internal to that AS. Traffic that fits this description is called “local traffic”. Traffic that does not fit this description is called “transit traffic”. One goal of BGP usage is to control the flow of transit traffic.

Based on the way a particular AS deals with transit traffic, the AS fits one of the following categories:

- stub AS  
A stub AS has a single connection to one other AS and only carries local traffic.
- multihomed AS  
A multihomed AS has connections to more than one other AS, but refuses to carry transit traffic.
- transit AS  
A transit AS has connections to more than one other AS, and is designed (under certain policy restrictions) to carry both transit and local traffic.

Since a full AS path provides an efficient and straightforward way of suppressing routing loops and eliminates the “count-to-infinity” problem associated with some distance vector algorithms, BGP imposes no topological restrictions on the interconnection of ASs.

## Sessions

BGP supports two basic types of sessions between neighbors, internal (sometimes referred to as IBGP) and external. Internal sessions are run between routers in the same autonomous system; external sessions run between routers in different autonomous systems. When sending routes to an external peer, the local AS number is prepended to the AS path, hence routes received from an external peer are guaranteed to have the AS number of that peer at the start of the path. Routes received from an internal neighbor do not generally have the local AS number prepended to the AS path. These routes usually have the same AS path that the route had when the originating internal neighbor received the route from an external peer.

Routes with no AS numbers in the path may be legitimately received from internal neighbors; these indicate that the received route should be considered internal to your own AS.

## Route advertisements

BGP collapses routes with similar path attributes into a single update for advertisement. Routes that are received in a single update are re-advertised in a single update. The churn caused by the loss of a neighbor is minimized, and the initial advertisement sent during peer establishment is maximally compressed. BGP does not read information from the kernel message-by-message, but fills the input buffer and processes all complete messages in the buffer before reading again. BGP also does multiple reads to clear all incoming data queued on the socket. This feature can cause other protocols to be blocked for prolonged intervals by a busy peer connection.

The BGP implementation expects external peers to be directly attached to a shared subnet, and expects those peers to advertise next hops that are host addresses on that subnet (this constraint can be relaxed by configuration for testing).

For groups of internal peers, alternatives can be selected by specifying the group type:

- Type internal groups expect all peers to be directly attached to a shared subnet so that, like external peers, the next hops received in BGP advertisements can be used directly for forwarding.
- Type routing groups determine the immediate next hops for routes by using the next hop received with a route from a peer as a forwarding address, and use this to look up an immediate next hop in an internal gateway's routes. These groups support distant peers, and need to be informed of the IGP whose routes they are using to determine immediate next hops.

For internal BGP group types, where possible, a single outgoing message is built for all group peers based on the common policy. A copy of the message is sent to every peer in the group, with possible adjustments to the next hop field as appropriate to each peer. This message minimizes the computational load of running large numbers of peers in these types of groups. BGP allows unconfigured peers to connect if an appropriate group is configured with an allow clause.

## Enforcing policy

BGP provides the capability for enforcing policies based on various routing preferences and constraints. Policies are not directly encoded in the protocol. Rather, policies are provided to BGP in the form of configuration information. BGP enforces policies by affecting the selection of paths from multiple alternatives and by controlling the redistribution of routing information.

Policies are determined by the AS administrator. Routing policies are created according to political, security, or economic considerations. The following are examples of routing policies that can be enforced with the use of BGP:

- 1 A multihomed AS can refuse to act as a transit AS for other ASs. It does so by only advertising routes to networks internal to the AS.
- 2 A multihomed AS can become a transit AS for a restricted set of adjacent ASs. It does so by advertising its routing information only to this set of ASs.
- 3 An AS can favor or disfavor the use of certain ASs for carrying transit traffic from itself.

Performance-related criteria can also be controlled using BGP. For example, an AS can minimize the number of transit ASs by preferring shorter AS paths over longer paths.

## Path selection

A major task of a BGP speaker is to evaluate different paths to a destination network from its border gateways at that network, select the best path, apply appropriate policy constraints, and then advertise that path to all its BGP neighbors.

The key issue is how different paths are evaluated and compared. In traditional distance vector protocols (RIP), only one metric (hop count) is associated with a path. As such, comparison of different paths is reduced to simply comparing two numbers. A complication in inter-AS routing arises from the lack of a universally agreed upon metric among ASs that can be used to evaluate external paths. Therefore, each AS can have its own set of criteria for path evaluation.

A BGP speaker builds a routing database consisting of the set of all feasible paths and the list of networks reachable through each AS sequence.

## Additions to BGP

As a contributing member of the GateD Consortium, Ascend's High Performance Network Division is developing additions to GateD, more specifically, to BGP.

These additions include:

- BGP confederations
- GateD State Monitor tool
- multi exit discriminator (MED)
- communities
- destination preference attribute (DPA)
- route filtering
- route reflection
- routing arbiter interaction
- route preference biasing
- stability support for BGP-OSPF interaction
- a weighted route dampening statement

## Configurable Export-Best-BGP (CEBB)

This feature is an enhancement of the GRF BGP protocol implementation. It allows a customer to configure their export policy in a manner that is consistent with the default functionality of the Cisco implementation. It can be described as BGP autonomy as well.

Essentially, if a route prefix is known via a protocol other than BGP and which is more preferred than BGP (for example, an IGP), a customer will be able to set `export-best-bgp` in their policy and will have the most-preferred BGP route exported without having to redistribute the IGP into BGP.

Note that the prefix will be exported or distributed within the IGP as well. The purpose of this feature is to potentially allow for increased BGP stability (only if next hops do not change) both internally and externally (advertised to AS-external peers). Second, if an IGP is redistributed into BGP and there is some level of instability w/in the IGP, it would appear as BGP flap to external peers. It also allows for simpler policy configuration because protocol redistribution is not required.



## Asynchronous Multi-Level Next Hop Resolution (AMLNHR)

This feature is an enhancement of the GRF BGP selection algorithm. It allows a customer to configure their intra-AS policy such that a prefix known via BGP can have its next hop resolved by another BGP prefix and, in turn, the next hops are resolved until a resolving IGP (OSPF, ISIS, RIP) prefix is found. This is consistent with the default functionality of the Cisco IOS implementation.

In the past, we only were able to walk up the “Next-hop-resolution-tree” one level because we did not support the resolution of next hops with BGP. So, if a user attempted to resolve a prefix learned via BGP, it would only compare the resolving BGP prefix with other prefixes that could possibly resolve the next hop one level above.

In other words, if the network conditions reached this part of the BGP selection algorithm:

Shortest IGP distance - The route whose NEXT\_HOP is closer (with respect to the IGP distance) is preferred.

The proper comparison of the “ultimate” resolving IGP routes is used. The depth of the search is completely dependent on the resolution of a prefix and is not “hard-coded” to a fixed depth limit.

Old style:

BGP prefix NH = A -> NH resolution of A  
(actual depth required for “ultimate” resolution disregarded)

New style:

BGP prefix NH = A -> NH resolution of A -> B = Resolution of A's NH -> C =  
Resolution of B's NH -> D = OSPF prefix

The purpose of this feature is to allow customers to route completely based upon closest exit or administratively-configured IGP variables. Also, it should add stability to customer networks because if a “level” within the resolution tree is withdrawn by a neighbor or peer, there is a greater likelihood of immediate resolution within GateD instead of potential flood storms until convergence.

## CLI access to gated.conf

The **gredit** command can access the `gated.conf` file from the command-line interface (CLI). Use a UNIX editor to make and save changes. Enter:

```
super> gredit gated
```

## BGP confederations

In addition to improvements in routing policy control, current techniques for deploying BGP among speakers in the same autonomous system generally require a full mesh of TCP connections among all speakers for the purpose of exchanging exterior routing information.

In autonomous systems, the number of intra-domain connections that need to be maintained by each border router can become significant. While this is usually acceptable in small networks, it may lead to unacceptably large internal peer groups in large networks. To help address this problem, BGP supports confederations for internal peer groups (with BGP version 4 only). It may be useful to subdivide an autonomous system into smaller domains for purposes of

controlling routing policy via information contained in the BGP AS\_PATH attribute (similar to EBGp). For example, one may choose to consider all BGP speakers in a geographic region as a single entity (confederation).

Subdividing a large autonomous system allows a significant reduction in the total number of intra-domain BGP connections, as the connectivity requirements simplify to the model used for inter-domain connections.

There is usually no need to expose the internal topology of this divided autonomous system, which means it is possible to regard a collection of autonomous systems under a common administration as a single entity or autonomous system when viewed from outside the confines of the confederation of autonomous systems itself.

A member of a BGP confederation will use its routing domain identifier (the internally visible AS number) in all transactions with peers that are members of the same confederation as the given router.

A BGP speaker receiving an AS\_PATH attribute containing a confederation ID matching its own confederation treats the path in the same fashion as if it had received a path containing its own AS number. Thus, BGP peering sessions at the confederation border are analogous to external BGP transactions within an AS; and BGP peering sessions within the confederation are analogous to internal BGP transactions.

The confederation implementation conforms to RFC 1965, "Autonomous System Confederations for BGP".

## **GSM - monitoring tool for GateD**

The GateD State Monitor (GSM) provides an interactive interface to a running GateD daemon which can be used to query internal GateD variables. This interface is implemented the same as any other protocol in GateD, and accepts telnet connections to port 616. After user identification, GSM answers any query sent as ASCII commands.

The commands query memory, routing table, interface list and other internal parameters. As an example, the command set for BGP will:

- show bgp summary
- show bgp peer information
- show bgp group summary
- show bgp aspath information
- show number of routes sent to a peer

GateD State Monitor provides an interactive interface to a running GateD daemon which can be used to query internal GateD variables. In this release, GSM adds per peer statistics within the "show bgp detail" section and other improvements. The GSM interface is implemented like any other protocol in GateD. Users open a telnet connection to the machine running GateD. After user password identification, GSM answers any query sent as ASCII commands. Memory, routing table, interface list and other internal parameters are displayed for a range of protocols.

Users open a telnet connection to the machine running GateD on TCP port 616. To telnet to the GSM interface, one must use the password from the administrative 'netstar' account. The default password = "NetStar". If you have changed the password, use the new one.

If GateD is running on 192.22.22.22, here is the command:

```
telnet -a 192.22.22.22 616
Trying 192.22.22.22...
Connected to 192.22.22.22.
Escape character is '^]'.
Password? ----- (for example, NetStar)
1GRF Gated State Monitor. Version GateD R3_5Beta_3;
Path:
GateD-mike.netstar.com> show
1GRF HELP: The possible show subcommands are:
1GRF version : Show the current GateD version
1GRF kernel : Show the Kernel support
1GRF iso : Show the ISO support
1GRF interface [name|index]: Show interface status
1GRF memory : Show the memory allocation
1GRF ip : Show info about IP protocol
1GRF task : Show list of active tasks
1GRF ospf : Show info about OSPF protocol
1GRF timer : Show list of timers
1GRF bgp : Show info about BGP protocol
1GRF rip : Show info about rip protocol
1GRF isis : Show info about ISIS protocol
```

Note that the GSM prompt includes the machine's domain name. The top level command help or ? provides a list of available commands.

## MEDs

The Multi Exit Discriminator (MED) allows the administrator of a routing domain to choose among identical, equally-preferred routes received from a neighboring Autonomous System (AS).

This attribute is used only for decision making in choosing the best route to the AS. If all the other factors for a path to a given AS are equal, the path with the lower MED metric value takes preference over other paths. The implementation in this version of GateD treats a route with no declared MED as preferable to a route with any MED.

This attribute is not propagated to other neighboring ASs. However, this attribute may be propagated to other BGP speakers within the same AS. The value of the MED attribute is a four-octet unsigned integer.

## Communities

A community is a group of destinations that share a common property. For example, all destinations belong by default to the general Internet community. Each destination can belong to multiple communities. All prefixes with the community attribute belong to the communities listed in the attribute. An autonomous system administrator can define those communities to which a particular destination belongs.

A BGP speaker can use the communities attribute to control which routing information it accepts, prefers or distributes to other neighbors. Also, a BGP speaker receiving a route that does not have the communities path attribute can append this attribute to the route when propagating the route to its peers. A BGP speaker receiving a route with the communities path attribute can modify the attribute according to local policy.

The attribute consists of a set of four octet values, each of which specify a community. The community attribute values are encoded using an AS number in the first two octets, with the remaining two octets defined by the AS.

## Destination preference attribute (DPA)

The destination preference attribute (DPA) is intended to aid in the implementation of symmetric inter-domain routing in the multi-provider Internet. DPA is documented in <draft-ietf-idr-bgp-dpa-05.txt> and in <draft-ietf-idr-dpa-application-02.txt>.

The functionality provided by DPA can be provided more efficiently through the use of the community attribute, and use of that attribute is recommended instead of DPA. The GateD DPA implementation is provided only for interoperating with existing systems using DPA. Again, we suggest using communities.

## Route reflection

BGP route reflection for internal peer groups allows an administrator to configure a large autonomous system so that every border router in the autonomous system need not be “fully meshed” with every other border router (need not be an internal peer of every other border router), as is usually the case.

Similarly, with Route Reflection, every non-border router in the autonomous system need not be fully meshed with (be an internal peer of) every border router.

## Routing arbiter interaction

GateD can interact with the Routing Arbiter's Route Server, as deployed at various Internet backbone Network Access Points (NAPs). GateD properly handles BGP routing updates from the Route Server. Previously, GateD would drop the Route Server peer session.

The “ignorefirstashop” option is added and enables GateD to retain routes propagated by the Route Server.

## Route preference biasing

The administrator of a BGP routing domain can modify the length of an exported AS Path by including the AS number multiple times in the AS Path using the “ascount” option. This can influence the neighbor's choice of preferred route, and help prevent sub-optimal routing over redundant links between neighboring autonomous systems.

## Interface statement

The Interface statement should always specify “interface all passive” to prevent policy from getting lost when interfaces are considered inoperable. This also prevents protocol time-outs due to interface fluctuations.

## BGP Group statement

This version of GateD allows for multiple declarations of the BGP Group statement to specify special network topologies, for example, route reflection. The existing BGP **group** statement now includes the **aspath-opt** field. The current **aspath-opt** attribute is **community** (Communities)

## BGP statement

This is the current BGP statement and valid options:

```
bgp yes | no | on | off
{
 protocol-precedence precedence ;
 allow bad community ;
 defaultmetric metric ;
 traceoptions trace_options ;
 [clusterid host ;]
[group type
 (external peeras autonomous_system
 [ignorefirstashop] [subgroup integer]
 [med])

 | (internal peeras autonomous_system
 [ignorefirstashop]
 [lcladdr local_address]
 [outdelay time]
 [metricout metric]
 [reflector-client [no-client-reflect]]
 [subgroup integer]

 | (routing peeras autonomous_system proto proto_list
 interface interface_list
 [ignorefirstashop]
 [lcladdr local_address]
 [outdelay time]
 [metricout metric]
 [reflector-client [no-client-reflect]]
 [subgroup integer]

 | (confed peeras autonomous_system proto proto_list
 interface interface_list
 [ignorefirstashop]
 [lcladdr local_address]
 [outdelay time]
 [metricout metric]
 [reflector-client [no-client-reflect]]
 [subgroup integer]

 | (test peeras autonomous_system)]
 [aspath-opt]
 {
 [allow {
 network
 network mask mask
 network (masklen | /) number
```

```
 all
 host host]
} ;
peer host
[metricout metric]
[localas autonomous_system]
[ignorefirstashop]
[nogendefault]
[gateway gateway]
[nexthopself]
[protocol-precedence precedence]
[preference preference]
[lcladdr local_address]
[holdtime time]
[version number]
[passive]
[sendbuffer number]
[recvbuffer number]
[outdelay time]
[keep [all | none]]
[show-warnings]
[noaggregatoid]
[keepalivesalways]
[v3asloopokay]
[nov4asloop]
[ascount count]
[throttle count]
[allow bad routerid]
[logupdown]
[ttl t11]
[traceoptions trace_options]
;
} ;
} ;
```

The **bgp** statement enables or disables BGP. By default, BGP is disabled. The default metric for announcing routes via BGP is no metric.

## Configuring multi-exit discriminators (MEDs)

External BGP sessions may or may not include a single metric in the path attributes, which BGP calls the Multi-Exit Discriminator (MED).

For BGP versions 2 and 3, this metric is a 16-bit unsigned integer, and for BGP version 4, it is a 32-bit unsigned integer. In either case, smaller values of the metric are preferred. Currently, this metric is only used to break ties between routes with equal preference from the same neighbor AS.

Internal BGP sessions carry at least one metric in the path attributes, which BGP calls the **LocalPref**. The size of the metric is identical to the **MED**.

For BGP versions 2 and 3, this metric is considered better when its value is smaller. For version 4, the metric is better when its value is larger. BGP version 4 sessions can optionally carry a second metric on internal sessions, this is an internal version of the Multi-Exit Discriminator. The use of these metrics depends on the type of internal protocol processing that is specified.

### Originating a MED

**MED** is originated using the **metricout** field in three BGP statements:

- the **group** statement
- the **peer** statement
- the **export** statement

**MED** is imported using the “med” on the group keyword statement.

### BGP group statement

Packets sent to this group of BGP peers have the MED in the BGP packet modified to be the MED value from this AS.

To use MEDs in routing computations:

```
group type external peeras autonomous_system med
```

To send MEDs :

```
group type routing peeras host proto protocol
 interface interface_list metricout metric
```

### BGP peer statement

Packets received for the BGP peer and aspath are also checked for MED within the BGP packet of this value.

```
peer host metricout metric
```



## BGP export statement

Packets exported to a BGP neighbor can select routes to be sent by specifying MED on the source BGP protocol:

```
export proto bgp as 2750
{
 proto bgp as 2704
 metric 10
 {
 route-filter
 }
}
```

## Configuring DPA attributes

The DPA attribute is a pair: an AS#, DPA value. The DPA value is a 32-bit, non-negative integer. The AS# is the AS number of the originator of the DPA. The DPA may be used to set the degree of preference for a route. DPA can be used to set a route's preference using import policy. It can also be originated using export policy.

### Using import policy

DPA can be matched on import using the **aspath-opt** option. This is similar to how community import policy is written. The DPA value part of the attribute is always tested. The value of the AS# part of the DPA attribute can either be tested or left as a “don't care”. Policy can also explicitly state that the DPA attribute must be absent.

To match DPA value only:

```
aspath-opt {dpa <dpa-value>}
```

To match DPA value and AS#:

```
aspath-opt {dpa <dpa-value> as <AS#>}
```

To match updates that do not include DPA:

```
aspath-opt {dpa none}
```

For example, if we wanted to assign preference 180 to routes with DPA of 100, preference 190 to routes with DPA of 200, preference of 200 to routes with DPA of 200 from AS 1, and preference of 210 to all other routes, we could use the following import policy:

```
import proto bgp as 2 aspath-opt {dpa 100} preference 180 {all};;
import proto bgp as 2 aspath-opt {dpa 200 as 1} preference 200
{all};;
import proto bgp as 2 aspath-opt {dpa 200} preference 190 {all};;
import proto bgp as 2 preference 210 {all};;
```

Note that both community and DPA policy can be specified together, as in:

```
aspath-opt {dpa 10 comm-split 10 20}
```

### Export policy

DPA can be originated on export using the **mod-aspath** option. This is similar to how community export policy is written. Unlike communities, the DPA specified in export policy overrides any existing DPA. The syntax used is similar to that used for **import**. If the AS# is not specified, the local AS number will be used.

For example, suppose we want to send routes to peers in AS 2 with DPA of 100:

```
export proto bgp as 2 mod-aspath {dpa 100}
{proto bgp as 1 {all};};
```

If we wish to write a different originating AS# into the DPA (AS 3, for example):

```
export proto bgp as 2 mod-aspath {dpa 100 as 3}
{proto bgp as 1 {all};};
```

Such "AS spoofing" is generally discouraged.

## Configuring route reflection

Route reflection is a technique for efficiently communicating routes in an internal autonomous system (AS).

In autonomous systems, each BGP speaker peering with an external router is responsible for propagating reachability and path information to all other transit and border routers within that AS. This is typically done by establishing internal BGP connections to all transit and border routers in the local AS.

In autonomous systems with a large number of peers, this practice leads to a formidable mesh of TCP connections. Using internal BGP “route reflectors” reduces configuration, memory, and CPU requirements necessary to convey external route information to all other BGP peers in the AS. A router acting as a route reflector delivers (advertises) external routes between multiple client peers.

The internal peers of a route reflector are divided into client peers and non-client peers – a route reflector and its client peers form a cluster. Client peers need not be fully meshed and should not peer with internal speakers outside of their cluster. Non-client peers must be fully meshed.

Figure 12-1 shows how route reflection saves internal BGP (IBGP) advertising links by assigning a router reflector to deliver its external routes to peer routers:

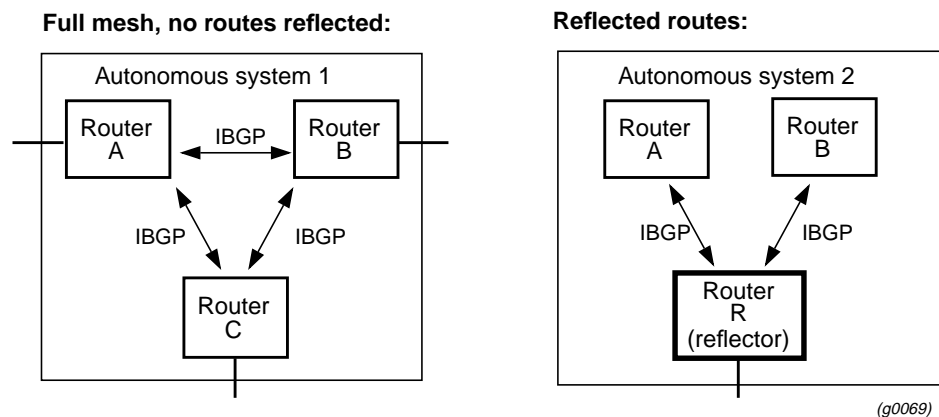


Figure 12-1. Full mesh and reflected route topologies

In autonomous system 1 above, all routers exchange external BGP information. In autonomous system 2, router R advertises external updates to client peer routers A and B as well as communicating A's updates to B, and B's to A. In both topologies, routers A and B have the same BGP tables.

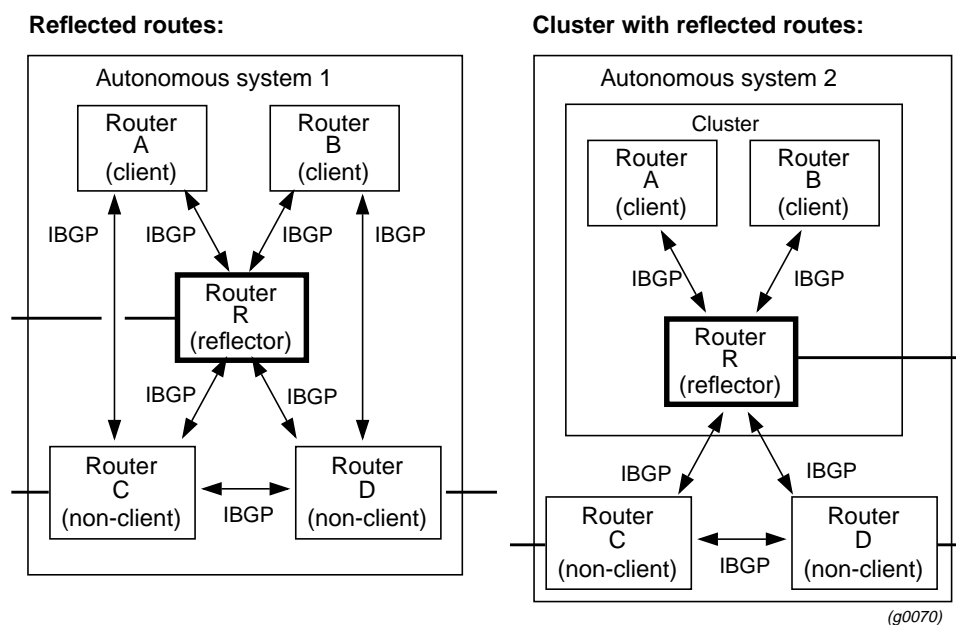


Figure 12-2. Client and non-client reflected route topologies

These examples add two non-client peers not participating in route reflection. Router R reflects routes between these groups. Updates from a client router are sent to all other client and non-client routers. Updates from non-client routers C and D are sent to client routers A and B. Updates from an external peer router are sent to all client and non-client routers.

## Route reflection example

This example demonstrates how to use `gated.conf` to establish route reflection for the set of routers illustrated here.

Routers A and B are clients of route reflector R. Together, routers A, B, and R form a route-reflection cluster. Routers C and D are in the same autonomous system as the other routers, but C and D are not part of the route reflection cluster.

No import or export entries are present in these `gated.conf` excerpts. With respect to route reflection, the configurations shown are sufficient. By default, routes will be imported from IBGP and exported to IBGP.

In a real-world configuration, there may be more importation and exportation. In particular, there are no static routes or EBGP peers here. The latter would limit the effectiveness of the IBGP mesh.

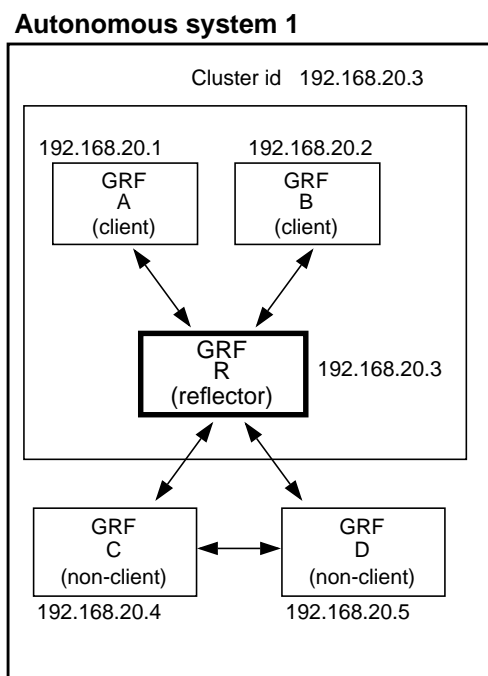


Figure 12-3. Route reflection configuration example

The next several pages provide the GateD entries for each of the routers included in the route reflection example illustrated in Figure 12-3.

### GateD entries for GRF A - a route reflector client

```
This is where the interfaces are defined
#
interfaces {
 interface all passive ;
} ;

routerid 192.168.20.1 ;
autonomous system 1 ;

#
```

```
start BGP section
#
This shows only route reflector portion of the parameters.
You can add other parameters as needed.

bgp on {
 group type routing peeras 1 proto rip interface all
 {
 peer 192.168.20.3;
 };
} ;

#
End BGP section
#

rip yes;

#
Import Policies
#
#End of File
#
```

### *GateD entries for GRF B - a route reflector client*

```
This is where the interfaces are defined
#
interfaces {
 interface all passive ;
} ;

routerid 192.168.20.2 ;
autonomoussystem 1 ;

#
start BGP section
#
This shows only route reflector portion of the parameters.
You can add other parameters as needed.

bgp on {
 group type routing peeras 1 proto rip interface all
 {
 peer 192.168.20.3;
 };
} ;

#
End BGP section
#

rip yes ;
#
Import Policies
#
#End of File
```

### *GateD entries for GRF R*

GRF R is a route reflector with clients A and B and non-clients D and E.

```
This is where the interfaces are defined
#
interfaces {
 interface all passive ;
} ;

routerid 192.168.20.3 ;
autonomoussystem 1 ;
#
start BGP section
#
This shows only route reflector portion of the parameters.
You can add other parameters as needed.
#
When acting as a route reflector, it is necessary to export
routes from the local AS into the local AS.

bgp on {
 clusterid 192.168.20.3 ;

 group type routing peeras 1 proto rip interface all reflector-client
 {
 peer 192.168.20.1;
 peer 192.168.20.2;
 };

 group type routing peeras 1 proto rip interface all
 {
 peer 192.168.20.4;
 peer 192.168.20.5;
 };
} ;

#
End BGP section
#

rip yes ;

#
Export Policies
#
export proto bgp as 1 { all; } ;

#End of File
```

### *GateD entries for GRF C - a non-client peer*

```
This is where the interfaces are defined
#
interfaces {
 interface all passive ;
} ;

routerid 192.168.20.4 ;
autonomoussystem 1 ;

#
start BGP section
#
bgp on {

 group type routing peeras 1 proto rip interface all
 {
 peer 192.168.20.3;
 peer 192.168.20.5;
 };

} ;

#
End BGP section
#
```

```
rip yes ;

#
Import Policies
#
#End of File

GateD entries for GRF D - a non-client peer
This is where the interfaces are defined
#

interfaces {
 interface all passive ;
} ;

routerid 192.168.20.5 ;
autonomous system 1 ;

#
start BGP section
#
bgp on {
 group type routing peeras 1 proto rip interface all
 {
 peer 192.168.20.3;
 peer 192.168.20.4;
 };
} ;

#
End BGP section
#

rip yes;

#
Import Policies
#
#End of File
```



## Weighted route dampening

The basic idea of weighted route dampening is to treat routes that are being announced and withdrawn (flapping) at a rapid rate as unreachable.

If a route flaps at a low rate, it should not be suppressed at all, or suppressed only for a brief period of time. With weighted route dampening, the suppression of a route or routes occurs in a manner that adapts to the frequency and duration that a particular route appears to be flapping. The more a route flaps during a period of time, the longer it will be suppressed. The adaptive characteristics of weighted route dampening are controlled by a few configurable parameters.

**Note:** Configuration of weighted route dampening is done via the weighted route dampening statement, not within the BGP statement.

### Weighted route dampening statement

Currently, only routes learned via BGP are subject to weighted route dampening, and no protocols announce routes suppressed by such dampening.

The syntax of the weighted route damping statement is:

```
dampen-flap {
 [suppress-above metric;
 reuse-below metric;
 max-flap metric;
 unreach-decay time;
 reach-decay time;
 keep-history time;]
};
```

A floating point number in units of flaps is assigned to the first three parameters described below. Each time a route becomes unreachable, 1 is added to the current instability metric.

#### **suppress-above** *metric*

This is the value of the instability metric at which route suppression will take place. A route will not be installed in the forwarding information base (FIB), or announced even if it is reachable during the period that it is suppressed.

#### **reuse-below** *metric*

This is the value of the instability metric at which a suppressed route will become unsuppressed if it is reachable but currently suppressed. The value assigned to **reuse-below** must be less than **suppress-above**.

#### **max-flap** *metric*

This is the upper limit of the instability metric. This value must be greater than the larger of 1 and **suppress\_above**.

#### **reach-decay** *time*

This value specifies the time desired for the instability metric value to reach one-half of its current value when the route is reachable. This half-life value determines the rate at which the metric value is decayed. A smaller half-life value will make a suppressed route reusable sooner than a larger value. Specify in seconds.

**unreach-decay** *time*

This value acts the same as **reach-decay** except that it specifies the rate at which the instability metric is decayed when a route is unreachable. It should have a value greater than or equal to **reach-decay**. Specify in seconds.

**keep-history** *time*

This value specifies the period over which the route flapping history is to be maintained for a given route. The size of the configuration arrays described below is directly affected by this value. Specify in seconds.

When only **dampen-flap {}**; is specified in the route dampening statement, then the following default values are currently used:

- **suppress-above** = 3.0;
- **reuse-below** = 2.0;
- **max-flap** = 16.0;
- **unreach-decay** = 900;
- **reach-decay** = 300;
- **keep-history** = 1800;

# Introduction to IS-IS

# 13

Chapter 13 provides a brief description of IS-IS configuration on the GRF and configuration examples using a variety of media cards and protocols.

The following media cards support IS-IS:

- ATM OC-3c
- FDDI
- Ethernet 10/100Base-T, 4- and 8-port
- HSSI (all protocols)
- SONET OC-3c

This chapter contains these topics:

|                                             |       |
|---------------------------------------------|-------|
| IS-IS Intra-Domain Protocol .....           | 13-2  |
| Assign configuration values .....           | 13-3  |
| Settings in the IS-IS statement .....       | 13-4  |
| Enable IS-IS on HSSI interfaces .....       | 13-6  |
| Enable IS-IS on ATM/Q interfaces .....      | 13-6  |
| Assign ISO address in grifconfig.conf ..... | 13-7  |
| Configuration example 1 .....               | 13-8  |
| Configuration example 2 .....               | 13-11 |
| GSM support for IS-IS .....                 | 13-15 |

Additions to IS-IS include:

- the number of allowable IS-IS circuits and adjacencies per circuit has increased from 20 to 126.
- Link-state packet (LSP) refresh interval is settable, invalid LSPs are now logged and dropped, but not processed.
- The adjacency change counter is logged through **syslogd**.

## IS-IS Intra-Domain Protocol

IS-IS is a link state interior gateway protocol (IGP) originally developed for routing ISO/CLNP (International Organization for Standardization/Connectionless Network Protocol) packets. The version distributed with GateD can route IP packets as well.

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.

Level 1 systems route directly to systems within their own area and route toward a Level 2 Intermediate System when the destination system is in a different area. Level 2 Intermediate Systems route between areas and keep track of the paths to destination areas. Systems in the Level 2 subdomain route towards a destination area, or another routing domain. As with any internet routing protocol, IS-IS support for large routing domains may also include many types of individual subnetworks. These subnetworks may include point-to-point links and broadcast subnetworks like ISO 8802 LANs.

In IS-IS, all subnetwork types are treated by the subnetwork independent functions as though they were connectionless subnetworks using subnetwork convergence functions where necessary. Like OSPF, IS-IS uses a “shortest-path first” algorithm to determine routes. GateD configuration syntax allows as much autoconfiguration as possible.

This integration also allows the ability to specify policy for exchanging routing information with other protocols running in GateD, such as BGP, EGP, and RIP. This version of GateD supports IP and ISO.

### IS-IS Statement

This statement enables the IS-IS protocol in GateD and configures the interfaces that are to run IS-IS. By default, IS-IS is disabled. The IS-IS statement consists of an initial description of the Intermediate System and a list of statements that determine the configuration of the specific circuits and networks to be managed.

```
isis no | ip {
 level 1|2 ;
 [traceoptions <isis_traceoptions> ;]
 [systemid <string> ;]
 [area <string> ;]
 [set <isis_parm> value ;]
 circuit|interface <interface-name>
 metric [level 1|2] metric
 priority [level 1|2] priority pointopoint ;
 [ipreachability level (1|2) (internal|external|summary)
 ipaddr netmask [metric metric:]]
} ;
```

## Assign configuration values

To start IS-IS configuration, organize values for systemid, area address, and ISO address.

### Assign a systemid

Each router must have a unique systemid. If you assign the same systemid to two routers, the results will be unpredictable. The systemid can be specified as a hex string of 12 characters or as an ASCII character string of six characters.

The systemid is entered in two files:

- in the IS-IS statement in `gated.conf` (in hex with 0x appended, or as string)
- as part of the ISO address in `grifconfig.conf`, in the format:  
`<interface-name> <iso-address> <iso-area> - iso`

The values of `iso-area` and `iso-address` must be given in ISO addressing form.

### Assign an area address

Each area must have a unique area address. If you assign the same address to two areas, the results will be unpredictable. The area address can be specified as a hex string or as an ASCII character string of any number of characters.

Area is entered in two files:

- in the IS-IS statement in `gated.conf` (in hex with 0x appended, or as string)
- as part of the ISO address in `grifconfig.conf`, in the format:  
`<interface-name> <iso-address> <iso-area> - iso`

The values of `iso-area` and `iso-address` must be given in ISO addressing form.

### Derive an ISO address

The ISO address for a specific interface is derived by first appending the systemid to the area address and then adding the 00 NSEL parameter at the end. This is the ISO address entered in the `grifconfig.conf` file.

For example, if the area address is given as an ASCII character string of `cccc`, you must convert `cccc` into the ISO address format. The character `c` is 63 in ISO:

```
c = 63
cccc = 63636363
iso-area = 63.6363.63
```

Similarly, if a systemid is given as an ASCII character string of `111111`, you must convert `111111` into the ISO address format. The character `1` is 31 in ISO:

```
1 = 31
111111 = 313131313131
iso-address = 3131.3131.3131.00
```

The last 00 appearing in the `iso-address` above is the NSEL parameter.

## Settings in the IS-IS statement

The IS-IS statement enables the IS-IS protocol and configures the interfaces and networks upon which IS-IS will run. This section briefly overviews the most-used parameters in the statement. Refer to the GateD section in the *GRF Reference Guide* for details.

```
isis no | ip {
 level 1|2 ;
 [traceoptions <isis_traceoptions> ;]
 [systemid <string> ;]
 [area <string> ;]
 [set <isis_parm> value ;]
 circuit|interface <interface-name>
 metric [level 1|2] metric
 priority [level 1|2] priority pointopoint ;
 [ipreachability level (1|2) (internal|external|summary)
 ipaddr netmask [metric metric;]]
} ;
```

### Enable IS-IS

Use `isis yes` or `isis ip` in `gated.conf`:

```
isis yes {
} ;
```

### Configure area

Two routers in Level 1 mode communicate with each other only if they belong to the same area. Two Level 2 routers can communicate across area boundaries.

Example 1, area specified in hex:

```
isis yes {
 area "0x49000080";
} ;
```

Example 2, area specified as a string:

```
isis yes {
 area "aaaa";
} ;
```

The string `aaaa` is printed in hex in the trace file as `61.6161.61`

### Configure systemid

Each router in the network should have a unique systemid. The systemid can be specified as a hex string of 12 characters or as a character string of six characters.

Example 1, systemid specified in hex:

```
isis yes {
 area "0x49000080";
 systemid "0x326032603260";
} ;
```

Example 2, systemid specified as a string:

```
isis yes {
 area "aaaa";
 systemid "aaaaaa";
} ;
```

A systemid character string is printed in hex format in trace files.

### *Configure interface*

Each interface must be configured in `gated.conf` to receive and transmit IS-IS packets. An interface can be enabled using either the `interface` or `circuit` keyword. Any of the optional circuit parameters are allowed on this statement.

Example showing how FDDI interface `gf030` can be configured:

```
isis yes {
 area "49000080";
 systemid "326032603260";
 interface "gf030" metric 10 priority 60;
} ;
```

## ***Enable IS-IS on HSSI interfaces***

If a HSSI interface is to run IS-IS, the Frame Relay and PPP configuration files must have enabling IS-IS entries.

### ***Frame Relay***

If the interface runs Frame Relay, edit the line in `grfr.conf` that configures the interface's PVC. Add `ISIS=Y` to enable the PVC to accept IS-IS packets:

```
pvc gs02c 101 200.170.22.2 Enabled=Y Name="RouterB2" ISIS=Y
```

### ***PPP***

When the HSSI card runs PPP, add the line `"enable osinlcp"` under the existing line `"enable ipcp"` in the `/etc/grppp.conf` file if it is not there already.

### ***HDLC***

HDLC interfaces need no special IS-IS configuration, configure as usual.

## ***Enable IS-IS on ATM/Q interfaces***

Edit the `/etc/gratm.conf` file and add `proto=isis` or `proto=isis_ip` fields in the interface's PVC statement. The field enables an interface to accept IS-IS packets.

Here are two examples:

```
pvc ga030 0/40 proto=isis traffic_shape=high_speed_high_quality
pvc ga030 0/41 proto=isis_ip traffic_shape=high_speed_high_quality
```

Use the `proto=isis` field when IS-IS is the only protocol to run.

Use the `proto=isis_ip` field when IP and IS-IS will both run.



## ***Assign ISO address in grifconfig.conf***

Each interface to run IS-IS must be specifically identified in the `etc/grifconfig.conf` file with an ISO address.

The ISO address entry is in addition to the interface's initial IP address entry in the file.

This is the IP address entry and format:

```
Internet broadcast/
name address netmask destination arguments

ga030 192.0.2.1 255.255.255.0 192.0.2.255
```

Here is the ISO address entry and format:

```
<interface-name> <iso-address> <iso-area> - iso
ga030 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

We would expect to see these adjacent entries for ATM interface `ga030` in `grifconfig.conf` :

```
ga030 192.0.2.1 255.255.255.0 192.0.2.255
ga030 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

## Configuration example 1

In example 1, three routers are configured as shown in Figure 13-1. All routers are in one area, area 49.0000.80. Router 3 routes toward another Level 2 router on interface ga030.

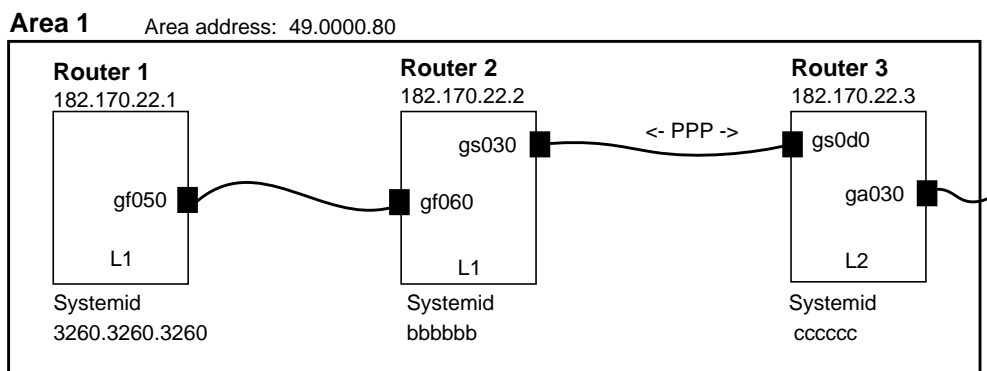


Figure 13-1. IS-IS configuration example - 1

### Router 1 configuration

Here are IS-IS configuration entries for Router 1:

*Line to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
gf050 49.0000.80.3260.3260.3260.00 49.0000.80 - iso
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 182.170.22.1;
rip no;
isis ip {
 level 1;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k files 5
 all;
 # Both systemid and area are in hex, hence 0x before the actual hex digits
 systemid "0x326032603260";
 area "0x49000080";
 interface "gf050" priority 20 metric 10;
};
static {
 200.222.1.0 masklen 24 gateway 198.174.11.48 preference 5 noinstall;
};
import proto isis {
 all;
};
export proto isis {
 proto static
 {all;};
};
```

```
};
```

## Router 2 configuration

Here are IS-IS configuration entries for Router 2:

*Lines to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
gs030 49.0000.80.6262.6262.00 49.0000.80 - iso
gf060 49.0000.80.6262.6262.00 49.0000.80 - iso
```

*Line to add in grppp.conf :*

```
enable osinlcp
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 182.170.22.2;
rip no;
isis ip {
 level 1;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k
 files 5 all;
 # systemid to be character string
 systemid "bbbbbb";
 # Area to be hex string.
 area "0x49000080";
 interface "gf060" priority 20 metric 10;
 # You want PPP operation, so set pointopoint on the interface statement.
 # Make sure you add "enable osinlcp" in /etc/grppp.conf for gs030.
 interface "gs030" priority 20 metric 10 pointopoint;
};
import proto isis {
 all;
};
```

## Router 3 configuration

Here are IS-IS configuration entries for Router 3: cccccc

*Lines to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
ga030 49.0000.80.6363.6363.00 49.0000.80 - iso
gs0d0 49.0000.80.6363.6363.00 49.0000.80 - iso
```

*Line to add in grppp.conf :*

```
enable osinlcp
```

*Lines to add in gratm.conf:*

```
pvc ga030 0/40 proto=isis traffic_shape=high_speed_high_quality
pvc ga030 0/41 proto=isis_ip traffic_shape=high_speed_high_quality
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 182.170.22.3;

rip no;
isis ip {
 level 2;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k files
 5
 all;
 systemid "cccccc";
 # Area is a hex address, so you need 0x
 area "0x49000080";

 # Make sure you set pvc proto=isis or proto=isis-ip in /etc/gratm.conf
 interface "ga030" priority 20 metric 10;
 interface "gs0d0" priority 20 metric 10 pointopoint;
};
import proto isis {
 all;
};
```

## Configuration example 2

In example 2, four routers are configured as shown in Figure 13-2. One Level 1 and one Level 2 router are in each area. Router B1 is the designated Level 2 router on Ethernet.

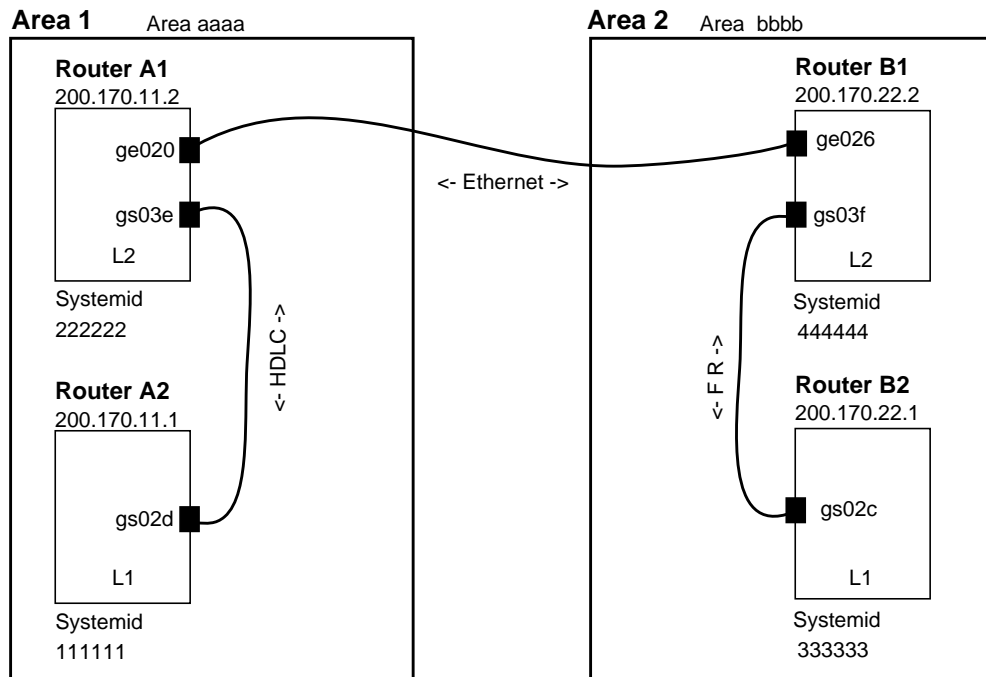


Figure 13-2. IS-IS configuration example - 2

### Router A1 configuration

Here are IS-IS configuration entries for Router A1:

*Lines to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
ge020 61.6161.61.3232.3232.3232.00 61.6161.61 - iso
gs03e 61.6161.61.3232.3232.3232.00 61.6161.61 - iso
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 200.170.11.2;

rip no;
isis ip {
 level 2;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k files
 5
 all;
 # Both systemid and area are strings.
```

```
 systemid "111111";
 area "aaaa";
 interface "gs03e" priority 20 metric 10;
 # Ethernet interface runs L2 only.
 interface "ge020" priority level 2 20 metric level 2 10;
 };
 import proto isis {
 all;
 };
};
```

## Router A2 configuration

Here are IS-IS configuration entries for Router A2:

*Line to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
gs02d 61.6161.61.3131.3131.3131.00 61.6161.61 - iso
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 200.170.11.1;

rip no;
isis ip {
 level 1;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k files
 5
 all;
 # Both systemid and area are strings.
 systemid "111111";
 area "aaaa";
 interface "gs02d" priority 20 metric 10;
};
import proto isis {
 all;
};
export proto isis {
 proto static
 {all;};
};
```

## Router B1 configuration

Here are IS-IS configuration entries for Router B1:

*Lines to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
ge026 62.6262.62.3434.3434.3434.00 62.6262.62 - iso
gs03f 62.6262.62.3434.3434.3434.00 62.6262.62 - iso
```

*Line to add in grfr.conf:*

```
pvc gs03f 102 200.170.22.1 Enabled=Y Name="RouterB1" ISIS=Y
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 200.170.22.2;
rip no;
isis ip {
 level 2;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k files 5
all;
 # Both systemid and area are strings.
 systemid "444444";
 area "bbbb";
 # Make sure you add ISIS=Y for the designated PVC in /etc/grfr.conf
 interface "gs03f" priority 20 metric 10;
 # Ethernet interface runs L2 only, this router wants to be DR, so priority 30 is set
 # which is higher than Area 1 A2 router
 interface "ge026" priority level 2 30 metric level 2 10;
};
import proto isis {
 all;
};
```

## Router B2 configuration

Here are IS-IS configuration entries for Router B2:

*Line to add in grifconfig.conf:*

```
<interface-name> <iso-address> <iso-area> - iso
gs02c 62.6262.62.3333.3333.3333.00 62.6262.62 - iso
```

*Line to add in grfr.conf:*

```
pvc gs02c 101 200.170.22.2 Enabled=Y Name="RouterB2" ISIS=Y
```

*Entries in gated.conf:*

```
interfaces {
 interface all passive;
};
routerid 200.170.22.1;
rip no;
isis ip {
 level 1;
 traceoptions "/var/tmp/gated_isis_trace" replace size 2000k
files 5 all;
 # Both systemid and area are strings.
 systemid "333333";
 area "bbbb";
```

```
 # This is a FR interface. Make sure you add ISIS=Y for the designated PVC
interface "gs02c" priority 20 metric 10;
};
import proto isis {
 all;
}
```



## GSM support for IS-IS

Gated State Monitor provides an interactive interface to a running GateD daemon which can be used to query internal GateD variables. The *GRF Reference Guide* provides more information about GSM. This section shows IS-IS options in GSM.

Users open a telnet connection to the machine running GateD on TCP port 616. To telnet to the GSM interface, one must use the password from the administrative 'netstar' account. The default password = "NetStar". If you have changed the password, use the new one.

If GateD is running on 192.22.22.22, here is the command:

```
telnet -a 192.22.22.22 616
Trying 192.22.22.22...
Connected to 192.22.22.22.
Escape character is '^]'.
Password? ----- (for example, NetStar)
1GRF Gated State Monitor. Version GateD R3_5Beta_3;
Path:
GateD-mike.netstar.com> show
1GRF HELP: The possible show subcommands are:
1GRF version : Show the current GateD version
1GRF kernel : Show the Kernel support
1GRF iso : Show the ISO support
1GRF interface [name|index]: Show interface status
1GRF memory : Show the memory allocation
1GRF ip : Show info about IP protocol
1GRF task : Show list of active tasks
1GRF ospf : Show info about OSPF protocol
1GRF timer : Show list of timers
1GRF bgp : Show info about BGP protocol
1GRF rip : Show info about rip protocol
1GRF isis : Show info about ISIS protocol
```

Note that the GSM prompt includes the machine's domain name. The top level command help or ? provides a list of available commands.

```
telnet -a 206.146.160.151 616
Trying 206.146.160.151...
Connected to 206.146.160.151.
Escape character is '^]'.
Password?
1GRF Gated State Monitor. Version GateD R3_5Beta_3; CVS
Branch:A1_4_0_2; Path:/nit/A1_4_0_2/gated/src/obj.BSD.OS-2.1-i386
GateD-mike.netstar.com> sh
1GRF HELP: The possible subcommands are:
1GRF version : Show the current GateD version
1GRF kernel : Show the Kernel support
1GRF iso : Show the ISO support
1GRF interface [name|index]: Show interface status
1GRF memory : Show the memory allocation
1GRF ip : Show info about IP protocol
```

```
1GRF task : Show list of active tasks
1GRF ospf : Show info about OSPF protocol
1GRF timer : Show list of timers
1GRF bgp : Show info about BGP protocol
1GRF rip : Show info about rip protocol
1GRF isis : Show info about ISIS protocol
```

```
GateD-mike.netstar.com> sh isis
1GRF HELP: The possible subcommands are:
1GRF summary: Show IS-IS summary
1GRF lspdb : Show isis lspdb level [1|2] [summary|detail]
```

```
GateD-mike.netstar.com> sh isis sum
```

```
1GRF ISIS DISPLAY
1GRF PDU Format Errors 0
1GRF Corrupt LSPs Detected 0
1GRF L1 LSP Overload 0
1GRF L2 LSP Overload 0
1GRF Manual Areas Dropped 0
1GRF Sequence Number Exceeded 0
1GRF Sequence Number Skipped 0
1GRF Purge Local LSP 0
```

```
1GRF Memory Usage
```

|                           |   |         |
|---------------------------|---|---------|
| 1GRF option blocks        | 0 | 0 bytes |
| 1GRF option data blocks   | 0 | 0 bytes |
| 1GRF IP prefix blocks     | 0 | 0 bytes |
| 1GRF ISO prefix blocks    | 0 | 0 bytes |
| 1GRF attribute blocks     | 0 | 0 bytes |
| 1GRF interior node blocks | 0 | 0 bytes |
| 1GRF IP leaf blocks       | 0 | 0 bytes |
| 1GRF ISO leaf blocks      | 0 | 0 bytes |
| 1GRF lsp desc blocks      | 0 | 0 bytes |
| 1GRF lsp entry blocks     | 0 | 0 bytes |
| 1GRF IP summary blocks    | 0 | 0 bytes |
| 1GRF Prefix change blocks | 0 | 0 bytes |
| 1GRF lsp buffers          | 0 | 0 bytes |
| 1GRF adj entries          | 0 | 0 bytes |
| 1GRF agelists             | 0 | 0 bytes |
| 1GRF circuit entries      | 0 | 0 bytes |
| 1GRF adj neighbor info    | 0 |         |

```
1GRF Option Breakdown
```

```
1GRF IP external prefix 0
1GRF IP internal prefix 0
1GRF IP summary prefix 0
1GRF ISO prefix 0
1GRF Leaf node 0
1GRF IDRP info 0
1GRF IS neighbors 0
```

|      |                   |       |         |                |       |
|------|-------------------|-------|---------|----------------|-------|
| 1GRF | ES neighbors      | 0     |         |                |       |
| 1GRF | Area Address      | 0     |         |                |       |
| 1GRF | IP interface      | 0     |         |                |       |
| 1GRF | Protos supported  | 0     |         |                |       |
| 1GRF | PD L2 IS          | 0     |         |                |       |
| 1GRF | IP authentication | 0     |         |                |       |
| 1GRF | Authentication    | 0     |         |                |       |
| 1GRF | Circuits          |       |         |                |       |
|      | Name              | State | Metrics | pdus (corrupt) |       |
| 1GRF | Adjacencies       |       |         |                |       |
|      | Neighbor          |       | Type    | ht             | State |



# Integrated Services: Controlled-Load

# 14

Chapter 14 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.

The chapter contains these topics:

|                                      |      |
|--------------------------------------|------|
| Overview .....                       | 14-2 |
| Controlled-Load implementation ..... | 14-2 |
| Filters .....                        | 14-3 |
| Filter examples .....                | 14-3 |

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.

## **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.

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;
```





# Transparent Bridging

# 15

Chapter 15 describes the GRF bridging implementation and provides configuration information.

The chapter includes these topics:

|                                                       |       |
|-------------------------------------------------------|-------|
| GRF bridging implementation .....                     | 15-2  |
| Bridging components .....                             | 15-5  |
| Management tools .....                                | 15-6  |
| Configuration file and profile overview .....         | 15-7  |
| 1. Starting bridged .....                             | 15-8  |
| Bridging example .....                                | 15-9  |
| 2. Creating bridge groups in bridged.conf .....       | 15-10 |
| 3. Assign IP addresses to bridge groups .....         | 15-11 |
| 4. Create an ATM PVC for an encapsulated bridge ..... | 15-12 |
| Sources of bridging data .....                        | 15-16 |
| Examining and debugging bridge configurations .....   | 15-19 |

## 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
- up to 16 bridge groups per GRF
- up to 32 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
- RFC1483, 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.

## 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 16 active 802.1 bridge groups, and will separate 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 interfaces 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 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 interfaces also “age” entries according to a site-specified time-out value. 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.

The time-out is specified in seconds in the `/etc/bridged.conf` file.

## 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 is when a bridging interface forwards a frame to all active interfaces in the bridge group. On the GRF, spamming is done when a broadcast address is received, or when a frame arrives whose destination address is not in the bridge filtering table.

## Bridging components

### Bridging daemon – bridged

The bridging daemon, **bridged**, is used to configure and manipulate 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 logical interfaces to a group
- assign priority, starting state, root path cost, and forwarding addresses to individual logical 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 – breedit

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 **:q** command.

At this point **bredit** runs a script in which you are asked if you want to make the changes permanent. The script also gives you 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 script options, **bredit** tells you that your changes were not committed.

If **bridged** is not running when **bredit** is used, the user is given the option of saving changes to the configuration in `bridged.conf` so that the next time **bridged** is started, the new changes take effect.

## 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 port information is needed.

```
super> brinfo bridge_group | all
```

**brinfo** prints the list of bridge groups and the bridge ports (underlying interfaces) that are members of the specified group(s). If no groups 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**.

## Configuration file and profile overview

When a new GRF system is installed or a site upgrades to a bridging software release, the `bridged.conf` file does not exist. The bridging daemon, **bridged**, will not start without this file. The **grstart** program periodically checks to see if the `bridged.conf` file exists; when it finds the file, **grstart** then starts **bridged**. The first configuration step is to create the `bridged.conf` file.

A template file for `bridged.conf` is provided in `/etc/bridged.conf.template`. Copy the template file `/etc/bridged.conf.template` into `/etc/bridged.conf`. Then run **bredit**.

These are the steps to configure bridging interfaces and parameters:

### 1. Start bridged

To start the bridging program, you copy the `bridged.conf` configuration file template and remove the `.template` ending.

### 2. Create bridge groups in bridged.conf

In the `/etc/bridged.conf` configuration file, you create and name the bridge groups, and assign bridging parameters to each.

### 3. Assign an IP address to each bridge group

Edit `/etc/grifconfig.conf` to identify each bridge group by assigning:

- an IP address
- the GRF interface name
- a netmask, required
- a destination or broadcast address, as required
- an MTU value, if needed

### 4. Create ATM OC-3c PVCs for encapsulated bridges

If you are going to configure an encapsulated bridge on an ATM circuit, edit the `gratm.conf` file to create a PVC on the ATM OC-3c logical interface.

### 5. Specify ARP service, if needed

Specify ARP service in the `/etc/grarp.conf` file.

## 1. Starting bridged

The **grstart** program is always checking for the presence of `/etc/bridged.conf` and, as shown below, immediately starts the **bridged** daemon when the file is found.

In the UNIX shell, copy the template for the bridging configuration file to `/etc/bridged.conf`:

```
cp /etc/bridged.conf.template /etc/bridged.conf
```

In a minute or two you will see this message:

```
Nov 11 21:43:26 testnode root: grstart: starting bridged.
```

Now you can use the **bredit** command to open `bridged.conf` and edit it:

```
brexit
#
NetStar $Id: bridged.conf,v 1.17.10.1 1997/11/18 23:27:21 pdm
#
Configuration file for Bridge Daemon (bridged).
#
Note: bridged will not start if it finds an error while
trying to parse this file. Use the "-d" option on the
command line with bridged to find proximity of the offending
line.
#
#
.
.
.
```

If you have not already copied `bridged.conf`, or if **grstart** cannot find the file, you see this message:

```
Could not find default config file : /etc/bridged.conf
This seems to be the first time bridged is being configured.
Do you want to use the template configuration file ? [y/n] [n]
```

If you respond “yes” (y) to the **bredit** query, the `bridged.conf` file is opened for you.

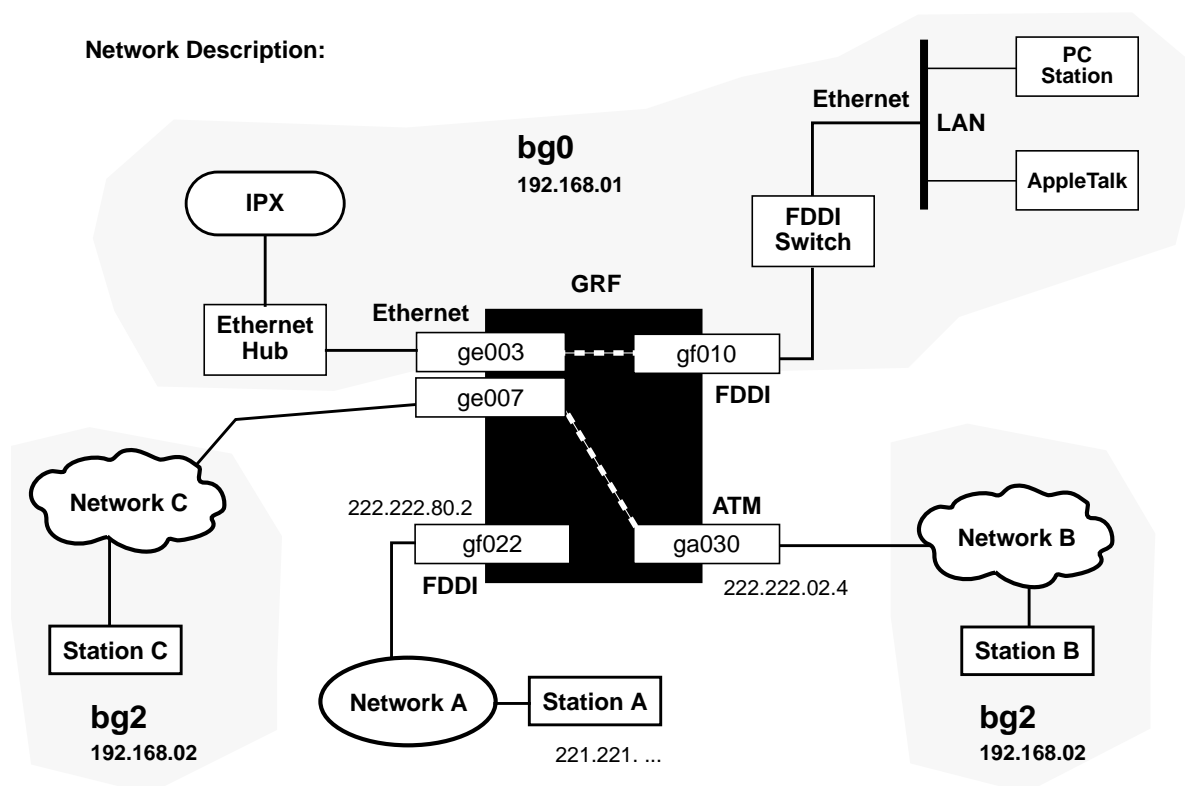
The `bridged.conf` file is opened in the UNIX **vi** editor. Comments in the file describe how to configure a group and its members, and define the bridging options and any defaults.

Exit the file with the **vi** command, **:q**, you do not need to write the file, **bredit** will do that.



## Bridging example

In this example, bridge group `bg0` is one 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.



### Configuration tasks:

1. Enter IP addresses in `grifconfig.conf`:
 

|                    |              |
|--------------------|--------------|
| <code>bg0</code>   | 192.168.01   |
| <code>bg2</code>   | 192.168.02   |
| <code>ge007</code> | 223.223. ... |
| <code>ga030</code> | 220.220. ... |
| <code>gf022</code> | 221.221. ... |
2. Define bridge groups in `bridged.conf`:
 

|                                |                                |
|--------------------------------|--------------------------------|
| <code>bridge_group bg0</code>  | <code>bridge_group bg2</code>  |
| <code>port ge003, gf010</code> | <code>port ge007, ga030</code> |
3. Set Network A for normal routed network.
4. Configure a PVC on `ga030` in `gratm.conf`.

Figure 15-1. Bridging example diagram

The GRF currently supports up to 16 bridge groups with as many as 32 logical GRF interfaces assigned to one group. A logical interface can be a member of only one bridge group.

From a GRF perspective, a bridge group equals a virtual LAN, each logical interface can support one virtual LAN.



## 3. Assign IP addresses to bridge groups

Assign an IP address to each bridge group in the `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 15-1:

| #     | 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 |            |          |
| ga030 |      | 222.222.02.4 | 255.255.255.0 |            |          |

A netmask entry is required for each bridge group.

## 4. 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 bridged traffic. The same PVC can also carry non-bridged traffic such as routed IP datagrams and IS-IS PDUs.

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 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
```

There are two types of bridging methods to specify:

- 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`.

- 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`. Media and transmission restrictions can

also be specified with this keyword, see the “LLC Encapsulation restrictions” section below.

### LLC Encapsulation restrictions

Media and transmission restrictions can be specified with LLC Encapsulation using these alternate `bridge_method,xxxx` keywords:

- `bridge_method=llc_encapsulated,broute_to_ether`  
IP datagrams are transmitted as Ethernet frames rather than using direct IP LLC/SNAP encapsulation.
- `bridge_method=llc_encapsulated,ether_only`  
All frames except BPDUs (routed datagrams and all bridged LAN frame types) are transmitted as Ethernet frames.
- `bridge_method=llc_encapsulated,broute_to_fddi`  
IP datagrams are transmitted as FDDI frames, rather than using direct IP LLC/SNAP encapsulation.
- `bridge_method=llc_encapsulated,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,xxxx`
- `proto=vcmux_bridge,yyyy`

### **proto=llc,bridging,xxxx**

This type of PVC is used for logical interfaces defined with `bridge_method=llc_encapsulated`, as well as logical interfaces not used for bridging. This PVC uses LLC encapsulation for each PDU.

The `,xxxx` represents a second protocol qualifier required for the `proto=` parameter. At this time, no second qualifiers are defined for LL encapsulation. Instead, LLC media and transmission restrictions are applied using the `bridge_method=` keywords in the Interfaces entry.

This PVC entry enables bridging on an LLC PVC:

```
PVCa0300/32proto=vcmux_bridge,bpdutraffic_shape=high_speed_high_quality
```

#### **proto=vcmux\_bridge,yyyy**

This type of PVC is used only for logical interfaces defined with `bridge_method=vc_muxplexed`. The PVC carries bridged traffic of a single type.

The ,yyyy represents a second protocol qualifier required for the `proto=` parameter. This qualifier defines the type of bridged traffic the PVC can carry. Traffic types include:

- `proto=vcmux_bridge,ether_fcs`  
Each PDU is an Ethernet frame, including a Frame Check Sequence.
- `proto=vcmux_bridge,ether_nofcs`  
Each PDU is an Ethernet frame, without a Frame Check Sequence.
- `proto=vcmux_bridge,fddi_fcs`  
Each PDU is an FDDI frame, including a Frame Check Sequence.
- `proto=vcmux_bridge,fddi_nofcs`  
Each PDU is an FDDI frame, without a Frame Check Sequence.
- `proto=vcmux_bridge,bpdu`  
Each PDU is an 802.1d Bridging Protocol Data Unit.

## PVC configuration examples

### LLC Encapsulated

Here is a sample LLC Encapsulated configuration. Note that the single PVC defined here can carry any kind of bridged frame, as well as routed IP traffic.

```
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_muxplexed
PVC
PVC ga030 0/32 proto=llc,bridging
```

### LLC Encapsulated, restricted to Ethernet

Here is a sample LLC Encapsulated configuration, restricted to Ethernet. Note that any IP 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_muxplexed,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 IP routed traffic is transmitted on its own PVC. If the separate IP PVC were not defined, then routed IP datagrams would be 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
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
PVC for IP (RFC 1577)
PVC ga030 0/35 proto=llc
```

## Configuration for ARP service - grarp.conf

Supply IP-to-physical address mapping information for ARP service.

Put an entry into `grarp.conf` ONLY if the remote destination does NOT support inverse ATM ARP. (The GRF supports inverse InATMARP.)

```
[ifname] hostname phys_addr [temp] [pub] [trail]
ga030 172.0.130.111 0/32
```

## 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 changes to take place. Enter:

```
greset <slot_number>
```

## 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/bridge.trace` log.

The output shown here reflects level 5, the default, and adequate for most debugging. Enter:

```
cd /var/tmp
vi bridge.trace
```

```
1997.11.25.11:39:46 NOTICE main.c:188 main() started
1997.11.25.11:39:46 NOTICE br_init.c:421 add_modify() adding group 'bg0' 1997.11.25.11:39:46 NOTICE
br_init.c:460 add_modify() adding 'ga030' to 'bg0'
1997.11.25.11:39:46 NOTICE br_init.c:460 add_modify() adding 'ge020' to 'bg0'
1997.11.25.11:39:46 NOTICE br_init.c:460 add_modify() adding 'ge021' to 'bg0'
1997.11.25.11:39:46 NOTICE br_init.c:460 add_modify() adding 'gf000' to 'bg0'
1997.11.25.11:39:46 NOTICE br_init.c:460 add_modify() adding 'gf002' to 'bg0'
1997.11.25.11:39:46 NOTICE standard.c:1103 std_initialisation() bg0 root [me]
1997.11.25.11:39:46 NOTICE standard.c:658 std_become_designated_port() ga030 desig bridge [me] port 1/1
1997.11.25.11:39:46 NOTICE standard.c:658 std_become_designated_port() ge020 desig bridge [me] port 128/2
1997.11.25.11:39:46 NOTICE standard.c:658 std_become_designated_port() ge021 desig bridge [me] port 128/3
1997.11.25.11:39:46 NOTICE standard.c:658 std_become_designated_port() gf000 desig bridge [me] port 2/4
1997.11.25.11:39:46 NOTICE standard.c:658 std_become_designated_port() gf002 desig bridge [me] port 128/5
1997.11.25.11:39:46 NOTICE standard.c:741 std_make_forwarding() bg0.ga030 Listening(1)
1997.11.25.11:39:46 NOTICE standard.c:741 std_make_forwarding() bg0.ge020 Listening(1)
1997.11.25.11:39:46 NOTICE standard.c:741 std_make_forwarding() bg0.ge021 Listening(1)
1997.11.25.11:39:46 NOTICE standard.c:741 std_make_forwarding() bg0.gf000 Listening(1)
1997.11.25.11:39:46 NOTICE standard.c:741 std_make_forwarding() bg0.gf002 Listening(1)
1997.11.25.11:40:01 NOTICE standard.c:997 std_forward_delay_timer_expiry() bg0.ga030 Learning(2)
1997.11.25.11:40:01 NOTICE standard.c:997 std_forward_delay_timer_expiry() bg0.ge020 Learning(2)
1997.11.25.11:40:01 NOTICE standard.c:997 std_forward_delay_timer_expiry() bg0.ge021 Learning(2)
1997.11.25.11:40:01 NOTICE standard.c:997 std_forward_delay_timer_expiry() bg0.gf000 Learning(2)
1997.11.25.11:40:01 NOTICE standard.c:997 std_forward_delay_timer_expiry() bg0.gf002 Learning(2)
1997.11.25.11:40:16 NOTICE standard.c:1000 std_forward_delay_timer_expiry() bg0.ga030 Forwarding(3)
1997.11.25.11:40:16 NOTICE standard.c:799 std_topology_change_detection() bg0 top_change ON
1997.11.25.11:40:16 NOTICE standard.c:1000 std_forward_delay_timer_expiry() bg0.ge020 Forwarding(3)
1997.11.25.11:40:16 NOTICE standard.c:1000 std_forward_delay_timer_expiry() bg0.ge021 Forwarding(3)
1997.11.25.11:40:16 NOTICE standard.c:1000 std_forward_delay_timer_expiry() bg0.gf000 Forwarding(3)
1997.11.25.11:40:16 NOTICE standard.c:1000 std_forward_delay_timer_expiry() bg0.gf002 Forwarding(3)
1997.11.25.11:40:51 NOTICE standard.c:1062 std_topology_change_timer_expiry() bg0 top_change OFF
```

*Figure 15-3. Output from bridging trace file*



## Bridge group information

**brinfo** returns configuration information about a bridge group and each of its member ports.  
Enter:

```
brinfo bridge_group
```

The number of ports in a group is stated in `brg_num_bports`:

```
brinfo bg0

Bridge Daemon: Running
Bridge_group: bg0
 brg_if_flags: 0x43 up,broadcast,running
 Ports: 2
 port ge003
 State: 0xf:Forwarding
 Flags: 0x9143 up,broadcast,running,promisc,link0,multicast
 Bridging media: ethernet
 MAC Address: 0:c0:80:00:55:d1

 port gf010
 State: 0xf:Forwarding
 Flags: 0x9143 up,broadcast,running,promisc,link0,multicast
 Bridging media: fddi
 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, Spanning Tree: Enabled
Log file: "/var/tmp/bridged.trace", Config file: "/etc/bridged.conf"
bridged started at: Thu April 11 21:43:28 1997
```

```
BRIDGE GROUP bg12
```

```
 Designated root : 00:a0:24:23:df:0d Priority: 105
 Bridge ID : 00:c0:80:01:46:90 Priority: 32768
```

```
 Root Path Cost : 10, Root Port : ga0680
 Root Max Age : 20, Hello Time: 2, Forward Delay : 5
 Bridge Max age: 20, Hello time: 2, Top Change Detected : 1
 Top Change: 1, Top Change Time: 25 sec, Hold Time: 1 sec
```

```
PORTS:
```

```
 gf080 Pri=128, Uni=1 State : 0 (Disabled): Not Connected, Cost : 10
 Desig Cost: 10, Top. Chng. Ack : 0 Config Pending : 0
 gf081 Pri=128, Uni=2 State : 0 (Disabled): Not Connected, Cost : 10
 Desig Cost: 10, Top. Chng. Ack : 0 Config Pending : 0
 gf082 Pri=128, Uni=3 State : 0 (Disabled): Not Connected, Cost : 10
 Desig Cost: 10, Top. Chng. Ack : 0 Config Pending : 0
 gf083 Pri=128, Uni=4 State : 0 (Disabled): Not Connected, Cost : 10
 Desig Cost: 10, Top. Chng. Ack : 0 Config Pending : 0
 ge065 Pri=128, Uni=5 State : 4 (Blocking): Connected, Cost : 10
```

```
Desig Cost: 10, Top. Chng. Ack : 0 Config Pending : 0
ge066 Pri=128, Uni=6 State : 3 (Forwarding): Connected, Cost : 10
Desig Cost: 0, Top. Chng. Ack : 0 Config Pending : 0
Dump snapshot finished at Sat April 13 22:22:22 1997
```

## Route trees and filtering table

The **netstat -rn** command returns the bridging filtering table of MAC addresses and other related information.

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 | Use       | Interface |
|-------------------|-------------------|-------|------|-----------|-----------|
| 00:c0:f2:00:1e:a0 | 00:c0:80:00:55:d2 | UHD   | 0    | 827173499 | gf081     |
| .                 | .                 | .     | .    | .         | .         |
| .                 | .                 | .     | .    | .         | .         |
| .                 | .                 | .     | .    | .         | .         |
| .                 | .                 | .     | .    | .         | .         |
| .                 | .                 | .     | .    | .         | .         |

*Source MAC address*

*Port 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       |

```
#
```

# 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`

## 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 **breedit**, 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
MAC address: 0:c0:80:0:55:d1

Port gf081 : State (0XF) FORWARDING
Flags : 0x9343 : up broadcast running promisc link0 multicast
MAC address: 0:c0:80:0:55:d3
```

All interfaces must be specified in `bridged.conf` to start 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:

```
Bridged Information:
=====
Debug level: 5, Trace mask: 0xffffffff, Spanning Tree: Enabled
Log file: "/var/tmp/bridged.trace", Config file: "/etc/bridged.conf"
/usr/sbin/bridged started at: Thu April 23 18:43:12 1997
BRIDGE GROUP bg0
Designated root : 08:00:2b:b6:38:80 Priority: 128
Bridge ID : 00:c0:80:00:55:d1 Priority: 129
Root Path Cost : 10, Root Port : gf081
Root Max Age : 15, Hello Time: 1, Forward Delay : 15
Bridge Max age: 20, Hello time: 2, Top Change Detected : 1
Top Change: 0, Top Change Time: 35 sec, Hold Time: 1 sec

PORTS:
gf080 Pri=128, Uni=1 State : 4 (Blocking): Connected, Cost : 10
 Desig Cost: 0, Top. Chng. Ack : 0 Config Pending : 0
gf081 Pri=128, Uni=2 State : 3 (Forwarding): Connected, Cost : 10
 Desig Cost: 0, Top. Chng. Ack : 0 Config Pending : 0

BRIDGE GROUP bg1
No ports defined
```

The configuration information starts at the **BRIDGE GROUP** section. The designated root shows the MAC address of the root bridge.

In the example above, the root bridge is transmitting BPDUs with a priority of 128. The GRF (bridge ID 00:c0:80:00:55:d1) is transmitting BPDUs at a priority of 129. If the priorities were equal, the MAC address would be used to determine the root bridge.

The MAC address of the GRF bridge 00:c0:80:00:55:d1 is also the address of the first FDDI interface in the first bridge group (in this example the first interface is `gf080`). In the case that all the ports (interfaces) in a bridge group are ATM interfaces without MAC addresses, the MAC address of the first Ethernet interface on the GRF is used to construct the bridge ID.

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.

The status of the interface corresponds to the result of the flags in the **brinfo** command described above.

## Bridge IDs via netstat -ni

Bridge IDs are listed as <Bridge> in the network column.

Use the **netstat -ni** command to check which interfaces are set as bridge IDs. An excerpt from **netstat** output is shown below:

```
netstat -ni
Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Coll
ef0 1500 <Link> 00:a0:24:23:e0:c6 135272 0 10474 1 4
ef0 1500 206.146.160 206.146.160.100 135272 0 10474 1 4
ef1 1500 <Link> 00:a0:24:23:f7:68 95866 0 95311 1 0
ef1 1500 <Bridge> 00:a0:24:23:f7:68 95866 0 95311 1 0
ef1 1500 <GRIT> 0:0x40:0 95866 0 95311 1 0
sl0* 296 <Link> 0 0 0 0 0
lo0- 1536 <Link> 6358 0 6358 0 0
lo0- 1536 127 127.0.0.1 6358 0 6358 0 0
lo0- 1536 <GRIT> 0:0x48:0 6358 0 6358 0 0
bg0 4352 <Link> 77 0 0 0 0
bg0 4352 198.174.59 198.174.59.100 77 0 0 0 0
gl000* 1524 <Link> 0 0 0 0 0
ga060 9180 <Link> 15 0 30 0 0
ga060 9180 <Bridge> 0:a0:24:23:f7:68 15 0 30 0 0
ga060 9180 222.222.20 222.222.20.2 15 0 30 0 0
ga0680 9180 <Link> 62 0 140 0 0
ga0680 9180 <Bridge> 0:a0:24:23:f7:68 62 0 140 0 0
ga0680 9180 222.222.30 222.222.30.1 62 0 140 0 0
gh070 65280 <Link> 0 0 0 0 0
gh070 65280 222.222.10 222.222.10.2 0 0 0 0 0
gf080 4352 <Link> 00:c0:80:01:46:90 1 0 5 0 0
gf080 4352 <Bridge> 00:c0:80:01:46:90 1 0 5 0 0
gf080 4352 222.222.60 222.222.60.2 1 0 5 0 0
gf081 4352 <Link> 00:c0:80:01:46:91 1 0 5 0 0
gf081 4352 <Bridge> 00:c0:80:01:46:91 1 0 5 0 0
gf081 4352 222.222.50 222.222.50.2 1 0 5 0 0
```

## Restarting bridged during debug

The **brsig** command provides a way to signal **bridged**. **brsig** takes the following parameters: **USR1**, **USR2**, and **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.
```

**SIGUSR1** causes **bridged** to write a dump file, `/var/tmp/bridged.dump` containing detailed information about the state of internal timers and bridging configuration. **bridged** rewrites the dump file each time it receives **SIGUSR1**.

**SIGUSR2** causes **bridged** to check the state of a bridging interface from the kernel's perspective.

**SIGHUP** causes **bridged** to reread the `bridged.conf` configuration file or an alternate file specified in the command line (usually reserved for debugging purposes). The **brexit** command sends a **SIGHUP** to **bridged**.



# Introduction to Subnetting

# A

The GRF supports variable-length subnet masking. This appendix describes these masks, and how they are used to improve the efficiency of routing.

Appendix A contains these topics:

|                                                          |      |
|----------------------------------------------------------|------|
| What is subnetting? .....                                | A-2  |
| Early implementation of classes and implicit masks ..... | A-2  |
| Classless inter-domain routing (CIDR) .....              | A-3  |
| Supernetting: benefits for routing .....                 | A-4  |
| Support for explicit netmasks .....                      | A-5  |
| Deriving a supernet address .....                        | A-5  |
| A supernet routing example .....                         | A-6  |
| Forming a supernet address .....                         | A-8  |
| How the GRF uses a mask .....                            | A-9  |
| Routing look-up example .....                            | A-10 |
| Address-to-mask logical ANDing .....                     | A-10 |
| Result-to-address comparison .....                       | A-11 |
| Rules for matching .....                                 | A-12 |
| Longest match example .....                              | A-12 |

## 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:

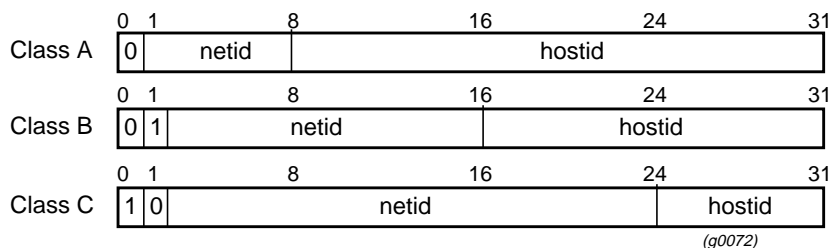


Figure A-1. Specification of classes in IP addresses

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:

| <u>Class:</u> | <u>Mask:</u>  |
|---------------|---------------|
| 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.

Users needing to address  $2^{24}$ th hosts or networks got a class A address; those needing to address  $2^{16}$ th hosts or networks got a class B; and those needing  $2^8$ th hosts or networks got a class C. It was assumed that small organizations would need class C; that most medium-sized companies would need class B; and that only a handful of the largest companies would need class A.

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  $2^{16}$  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  $2^{18}$  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:

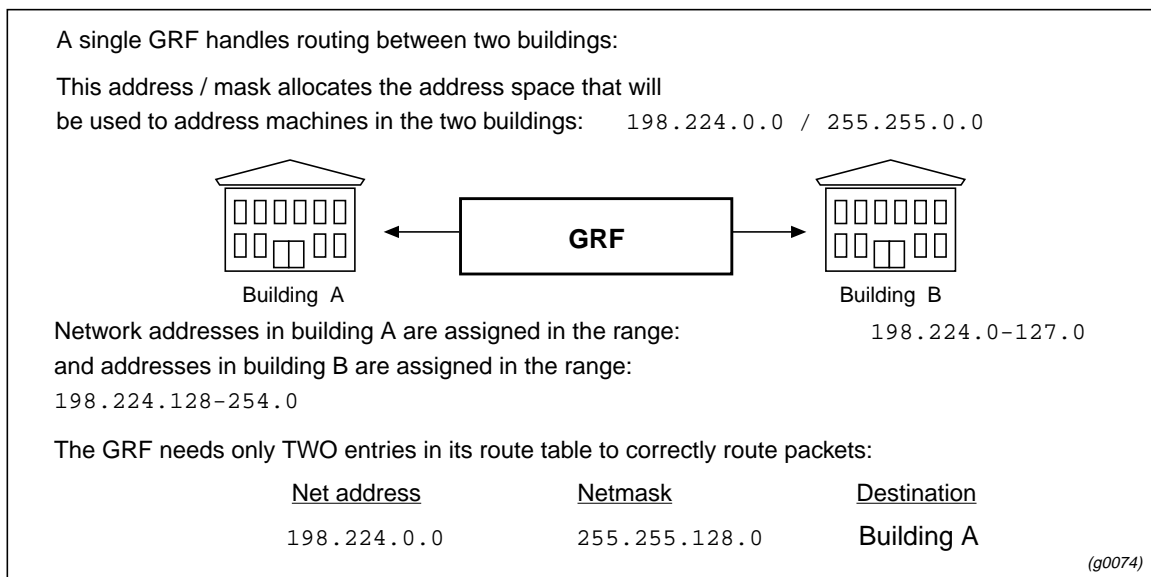


Figure A-2. Basic supernetting example

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 |
| <u>Destination netmask:</u>  | 255.255.0.0  |
| Supernet address:            | 139.51.0.0   |

### Example 2

|                              |               |
|------------------------------|---------------|
| Destination host IP address: | 137.66.12.48  |
| <u>Destination netmask:</u>  | 255.255.255.0 |
| Supernet address:            | 137.66.12.0   |

### Example 3

|                              |              |
|------------------------------|--------------|
| Destination host IP address: | 137.66.12.48 |
| <u>Destination netmask:</u>  | 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<br>(blocks) | Netmask<br>(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<br>(supernetted) | Netmask (binary)    | Destination address |     |
|------------------------------|---------------------|---------------------|-----|
| 192.24.0                     | (ff.ff.11111000.00) | 192.24.aaa.aaa      | (A) |
| 192.24.8                     | (ff.ff.11111100.00) | 192.24.bbb.bbb      | (B) |
| 192.24.12                    | (ff.ff.11111100.00) | 192.24.ccc.ccc      | (C) |
| 192.24.16                    | (ff.ff.11110000.00) | 192.24.ddd.ddd      | (D) |
| 192.24.32                    | (ff.ff.11111110.00) | 192.24.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, | 0  | = | 0000 0000 |
| the binary representation of  | 8  | = | 0000 1000 |
| each supernet address is:     | 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 the figure below, 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 |                       |
| Supernet 192.24.32 | 32 | = | 1111 1110 |                       |
| 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)<br>NOT a candidate |
| Result from logical AND → | = | 0000 1000 |                                                                   |
| 8                         | = | 0000 1000 | } Bits fail to match at position 6 (from left)<br>NOT a candidate |
|                           | = | 0000 1100 |                                                                   |
| 12                        | = | 0000 1100 | } Bits match in all masked positions !<br>CANDIDATE               |
|                           | = | 0000 1100 |                                                                   |
| 16                        | = | 0001 0000 | } Bits fail to match at position 4 (from left)<br>NOT a candidate |
|                           | = | 0000 0000 |                                                                   |
| 32                        | = | 0010 0000 | } Bits fail to match at position 3 (from left)<br>NOT a candidate |
|                           | = | 0000 1110 |                                                                   |

(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

This appendix contains warranty information for the GR-II, GRF 400, and GRF 1600.

## ***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.

# Index

## A

- AAL support
  - ATM OC-3c, 3-6
- abbreviating, CLI field names, 1-15
- action, in a filter binding, 11-8
- aitmd
  - ATMP daemon, 9-4
- alias address
  - as GateD routerid, 2-36
  - how to configure, 2-15
- alias for loopback interface, 2-15
- APS options
  - settings in Card profile, 1-25
  - SONET OC-3c, 8-14
- archiving configuration files, 2-50
- area, in IS-IS, 13-3
- argument field, in grifconfig.conf file, 2-14
- ARP, 3-21, 10-18
  - display table on FDDI card, 4-28
  - Ethernet, 7-3
  - inverse ARP, 3-22, 10-19
  - server in UNI signaling, 3-10
  - tables on HIPPI card, 5-40
- arrows, CLI
  - using up and down, 1-14
- aspath-opt, in GateD, 12-11
- asterisk
  - representing a profile index, 1-16
- ATM
  - ATM OC-12c media card, 10-1
  - ATM OC-3c media card, 3-2
  - ATM-MIB, 2-22
  - ATM/Q media card, 3-2
  - config files and profiles, 4-6
- ATM OC-12c
  - assign IP addresses, 10-10
  - broadcasting, 10-4
  - config files and profiles, 10-9
  - dump profile, 10-16
  - dump settings in card profile, 10-12
  - gratm.conf, configuring PVCs, 10-18
  - IS-IS support, 10-2
  - load profile, 10-14
  - logical interface name, 10-6
  - permanent virtual circuits (PVCs), 10-6, 10-8
  - PVCs per logical interface, 10-8
  - route table size, 10-2
  - selective packet discard, 10-2, 10-4
  - set ICMP in card profile, 10-12
  - set run-time code in card profile, 10-12
  - set selective packet discard, 10-11
  - setting output rates, 10-25
  - virtual circuits, 10-7
  - virtual paths, 10-7
- ATM OC-3c
  - AAL support, 3-6
  - assign IP addresses, 3-13
  - broadcasting, 3-4
  - buffer management during congestion, 3-3
  - config files and profiles, 3-12
  - controlled-load, 3-3
  - dump profile, 3-19
  - dump settings in card profile, 3-15
  - encapsulated bridging, 3-4
  - functions, 3-2
  - in ATMP configuration, 9-20
  - internal clock source, 3-6
  - IS-IS support, 3-3
  - load profile, 3-17
  - logical interface name, 3-8
  - packet buffering, 3-5
  - permanent virtual circuits (PVCs), 3-8, 3-10
  - ping times, 3-5
  - PVC reconfig without reset, 3-24
  - PVCs per logical interface, 3-10
  - selective packet discard, 3-37
  - set ICMP in card profile, 3-14
  - set internal/external clock source, 3-6
  - set run-time code in card profile, 3-15
  - set selective packet discard, 3-14
  - setting output rates, 3-34
  - signaling, 3-4
  - SVC example, 3-27
  - SVCs per logical interface, 3-10
  - virtual circuits, 3-9
  - virtual paths, 3-9

- 
- ATMP
    - atmp0 interface, 9-13
    - basic tunnel diagram, 9-2
    - configure foreign agent, 9-24
    - displaying related information, 9-33
    - encapsulation, 9-7
    - examples, 9-27
    - function of, 9-2
    - GRF as home agent, 9-3, 9-17
    - GRF in gateway mode, 9-5
    - handling large packets, 9-19
    - mobile node, 9-12
    - monitoring ATMP information, 9-27
    - OSPF broadcast, 9-15
    - pvcatmp, 9-15, 9-19
    - RADIUS profile, 9-12
    - starting aitmd, 9-4
    - tunnel ID, 9-8
    - tunnel negotiation, 9-8, 9-10
    - virtual private networks, 9-5
  - ATMP (Ascend Tunnel Management Protocol), 9-2
  - atmp0, in ATMP, 9-13
  - ATM/Q
    - configuring IS-IS, 13-6
  - autonegotiate
    - Ethernet setting in Card profile, 1-25
  - autonegotiation
    - Ethernet media card, 7-3
  - autonomous system
    - BGP connection, 12-3
    - description, 12-3
    - establishing policy, 12-5
    - transit AS, 12-5
  - autosensing
    - Ethernet, 7-3
- B**
- backing up configuration files, 2-50
  - BGP
    - AS path, 12-4
    - BGP4, A-5
    - confederations, 12-7
    - configuration, 12-1
    - group type, 12-5
    - MEDs, 12-14
    - multiple group statements, 12-11
    - new statement syntax, 12-12
    - path evaluation, 12-5
    - policy role, 12-5
    - route reflection, 12-17
    - sessions, 12-4
    - versions supported, 12-2
  - BGP statement, syntax of, 12-12
  - bindings, used with filters, 11-6
  - boot binary
    - settings in Card profile, 1-23
  - bredit utility
    - edits bridged.conf, 15-5
  - bridge filtering table, 15-4
  - bridge group
    - brinfo port information, 15-17
    - empty group, 15-10
    - how to configure, 15-10
    - IP address for, 15-11
    - number of groups allowed, 15-9
  - bridged.conf file, 15-5
  - bridging
    - and simultaneous routing, 15-3
    - ATM OC-3c, 3-4
    - bredit utility, 15-5
    - bridged functions, 15-5
    - brinfo, 15-6, 15-21
    - brsig command for debug, 15-24
    - brstat and brinfo management tools, 15-6
    - brstat output, 15-17, 15-22
    - configuration overview, 15-7
    - creating bridge groups, 15-10
    - debugging tips, 15-19
    - definition of interface states, 15-21
    - flags, 15-21
    - GRF implementation, 15-2
    - HIPPI role, 5-10
    - how to start bridged, 15-8
    - interoperability, 15-3
    - IP address for bridge groups, 15-11
    - IP fragmentation in, 15-4
    - LLC encapsulation restrictions, 15-13
    - number of groups, 15-9
    - obtaining bridge IDs via netstat, 15-23
    - obtaining trace output, 15-19
    - optional ARP configuration, 15-15
    - root bridge, 15-22
    - root path cost, 15-22
    - route table, 15-18
    - route tree, 15-18
    - sample configuration, 15-9
    - spanning tree, 15-4
    - trace log example, 15-16
    - virtual LAN support, 15-9
  - brinfo command, 15-6, 15-21
    - output sample, 15-17
  - broadcast
    - ATM OC-12c, 10-4
    - ATM OC-3c, 3-4
  - broadcast address
    - in grifconfig.conf, 2-14
  - brouting, 15-3
-



brsig command, 15-24  
brstat command, 15-6  
    sample output, 15-22  
buffer management  
    ATM OC-3c, 3-3

## C

camp-on, 5-3  
canonical output, FDDI, 4-23  
card profile  
    ATM OC-12c configuration, 10-9, 10-11  
    ATM OC-12c dump settings, 10-12  
    ATM OC-12c ICMP settings, 10-12  
    ATM OC-12c run-time code, 10-12  
    ATM OC-12c selective packet discard, 10-11  
    ATM OC-3c configuration, 3-12, 3-14  
    ATM OC-3c dump settings, 3-15  
    ATM OC-3c ICMP settings, 3-14  
    ATM OC-3c run-time code, 3-15  
    ATM OC-3c selective packet discard, 3-14  
    Cisco HDLC settings, 1-24  
    definition of, 1-17  
    drawing of levels, 1-21  
    Ethernet dump settings, 7-11  
    Ethernet ICMP settings, 7-10  
    Ethernet run-time code, 7-10  
    FDDI configuration, 4-7  
    FDDI dump settings, 4-10  
    FDDI ICMP settings, 4-7  
    FDDI optical bypass setting, 4-9  
    FDDI run-time code, 4-10  
    FDDI SAS/DAS setting, 4-8  
    FDDI SAS/DAS settings, 4-6  
    H0 HIPPI settings, 5-26  
    HIPPI configuration, 5-30  
    HIPPI dump settings, 5-32  
    HIPPI ICMP settings, 5-30  
    HIPPI run-time code, 5-31  
    how referenced, 1-17  
    HSSI configuration, 6-9, 6-13  
    HSSI dump settings, 6-17  
    HSSI Frame Relay settings, 6-27  
    HSSI framing protocol configuration, 6-13  
    HSSI HDLC settings, 6-23  
    HSSI ICMP settings, 6-15  
    HSSI PPP settings, 6-32  
    HSSI run-time code, 6-16  
    HSSI source clock setting, 6-14  
    packet discard setting, ATM OC-3c, 3-38  
    packet discard setting, Ethernet, 7-17  
    packet discard setting, FDDI, 4-22  
    packet discard setting, HSSI, 6-22

    packet discard setting, SONET OC-3c, 8-21  
    set HIPPI time-out values, 5-31  
    SONET OC-3c, 8-12  
    SONET OC-3c APS, mode, clock, payload settings, 8-14  
    SONET OC-3c dump settings, 8-16  
    SONET OC-3c framing protocol setting, 8-12  
    SONET OC-3c HDLC settings, 8-22  
    SONET OC-3c ICMP settings, 8-13  
    SONET OC-3c run-time code, 8-15  
    SONET OC-3c SPD settings, 8-13  
cd .., in profiles, 1-42, 1-44  
cd command  
    used with a profile, 1-40, 1-42  
CIDR, A-3  
Cisco-HDLC  
    keepalive settings in Card profile, 1-24  
    settings in Card profile, 1-24  
class-based addressing, A-2  
classless addressing, A-3  
CLI  
    abbreviating field names, 1-15  
    access to gated.conf, 11-2  
    limits to control characters, 1-16  
    line-editing commands, 1-14  
    list of commands, 1-8  
    on-line help, 1-12  
    paged line output, 1-16  
    printable characters, 1-16  
    setting password and permissions, 1-12  
    setting prompts, 1-12  
    typing shortcuts, 1-15  
    user profile, new user, 1-50  
    using control characters, 1-14  
    using up/down arrows, 1-14  
CLI, access to gated.conf, 12-7  
clock, HSSI, 6-8  
clock, SONET OC-3c, 8-8, 8-14  
commands, CLI  
    CLI line-editing, 1-14  
    displaying a list, 1-8  
    getting online help, 1-12  
    history buffer, 1-14  
    permission level, 1-8  
    repeating previous, 1-14  
    shortcuts, 1-15  
    system-level permissions, 1-8  
    update-level permissions, 1-8  
    user-level permissions, 1-8  
community  
    aspath-opt attribute, 12-11  
    in BGP, 12-10

- complex structure, in fields
  - define a list of fields, 1-19
  - how to view, 1-19
- confederations, in BGP, 12-7
- config\_netstart
  - changing system parameters, 2-49
- config\_netstart script
  - changing system parameters, 1-2, 1-5
- configuration
  - assign broadcast address, 2-14
  - assign destination address, 2-14
  - assign IP addresses, 2-13
  - assign mtu, 2-15
  - assign netmask, 2-14
  - BGP, 12-1
  - change GRF hostname, 2-17
  - duplicate among GRF systems, 2-45
  - GateD overview, 2-36
  - GateD trace/log file size, 2-6, 2-36
  - installing configuration files, 2-50
  - ISO address, 2-16
  - loose source routing, 2-38
  - NFS file system, 2-12
  - overview, 2-3
  - RADIUS client, 2-26
  - safely test a new configuration, 2-43, 2-44
  - saving alternate profiles, 1-47
  - securID client, 2-28
  - setting static routes, 2-31
  - SNMP steps, examples, 2-19
  - static-only routing, 2-31
  - telnet access, 2-18
  - using config\_netstart script, 2-49
- configuration changes
  - making them permanent, 2-5
- configuration files
  - archive, 2-4, 2-50
  - description of, 2-4
- configuration script
  - first-time power on, 2-9
- configurations, test
  - safely testing a new one, 2-43, 2-44
- configuring
  - Ethernet, 7-1
  - fast Ethernet, 7-1
  - SONET OC-3c, 8-1
- configuring ATM OC-12c, 10-1
- configuring ATM OC-3c cards, 3-1
- configuring ATMP, 9-1
- configuring bridging, 15-7
- configuring FDDI, 4-1, 11-1, 14-1
- configuring filters, 11-1
- configuring HIPPI, 5-2

- congestion threshold
  - settings in Card profile, 1-23
- congestion, selective packet discard
  - ATM OC-12c, 10-4
  - ATM OC-3c, 3-37
  - Ethernet, 7-16
  - FDDI, 4-21
  - HSSI, 6-21
  - SONET OC-3c, 8-20
- connectivity, simple tests, 2-52
- control characters
  - using in the CLI, 1-14, 1-16
- controlled-load
  - ATM OC-3c, 3-3
  - Ethernet, 7-4
  - FDDI, 4-3
  - HSSI, 6-4
  - SONET OC-3c, 8-4
- Controlled-Load, Integrated Services, 14-2
- CRC bits, HSSI, 6-7
- CRC, setting in Card profile, 1-25
- creating a new profile, 1-50

## D

- DAS
  - setting in Card profile, 1-24
- DAS settings (FDDI), 4-4
  - in Card profile, 4-17
- DCE, HSSI connection option, 6-7
- DCE, SONET OC-3c connection option, 8-7
- default route, in grifconfig.conf, 2-31
- deleting a profile, 1-46
- destination address
  - in grifconfig.conf, 2-14
- destination preference attribute, in GateD, 12-10
- dir command
  - displays list of GRF profiles, 1-39
- discovery facility, MTU, 2-15
- DLCI
  - in Frame Relay, SONET OC-3c, 8-27
  - multiple DLCIs per interface, 6-30, 8-29
- DLCI, in Frame Relay, 6-29
- DPA, in BGP, 12-10
- DTE, HSSI connection option, 6-7
- DTE, SONET OC-3c connection option, 8-7
- dual homing (FDDI), 4-21
- dump
  - settings in Card profile, 1-23

- dump profile
  - ATM OC-12c configuration, 10-16
  - ATM OC-3c configuration, 3-19
  - definition of, 1-17
  - drawing of levels, 1-27
  - Ethernet configuration, 7-14
  - FDDI configuration, 4-13
  - HIPPI configuration, 5-34
  - how referenced, 1-17
  - HSSI configuration, 6-19
  - SONET OC-3c configuration, 8-18
- dumps
  - sending to external flash device, 2-7
- dumps at panic
  - external flash support for, 2-6
- duplicate configurations
  - using external flash devices, 2-45
- dynamic routing
  - BGP, 12-2
  - selective packet discard, Ethernet, 7-16
  - selective packet discard, FDDI, 4-21
  - selective packet discard, HSSI, 6-21
  - selective packet discard, SONET OC-3c, 8-20
  - setting up GateD, 2-36
  - support from selective packet discard, 3-37, 10-4
- E**
- ect/ttys
  - enabling telnet sessions, 2-18
- EGP, comparison with BGP, 12-2
- emulate, 6-37
- encapsulated bridging
  - as implemented on ATM OC-3c, 15-2
  - PVC configuration on ATM interface, 15-12
- encapsulation, in ATMP, 9-7
- Enterprise MIBs, 2-22
- etc/bridged.conf, 15-5
- etc/fstab
  - editing for external flash logging, 2-7
- etc/gated.conf file
  - editing, 2-36
  - installing changes, 2-36
- etc/grarp.conf file
  - in configuring PVCs, 3-21
- etc/gratm.conf file
  - configuring ATM OC-3c PVCs, 3-21, 3-25
  - traffic shaping, ATM OC-3c, 3-33
- etc/grclean.logs.conf
  - after a software update, 2-6
- etc/grifconfig.conf file
  - configuring FDDI, 4-6
  - configuring PVCs, 3-21, 10-18
  - configuring SVCs, 3-25
  - format for entries, 2-13, 3-13, 10-10
  - GRF interface name, 4-5
  - identifying interfaces, 2-13
  - ifconfig command, 2-13
  - installing changes to, 2-16
  - ISO address, 2-15
  - loopback alias, 2-15
  - secondary address (alias), 2-15
  - setting MTU, 2-15
  - uses for argument field, 2-14
- etc/grinchd.conf, replacement of, 2-2
- etc/grlamap.conf file
  - in HIPPI IP routing, 5-19, 5-24
  - in logical addressing, 5-16
- etc/grroute.conf file
  - components and editing, 2-31
  - file format, 2-31
- etc/hosts file, 2-17
- etc/netstart file, 2-17
- etc/services
  - overwritten by upgrades, 2-41
- etc/snmpd.conf file
  - sample configurations, 2-19
- etc/syslog.conf
  - editing for logging, 2-8
  - editing for network logging, 2-9
  - GRF excerpt from, 2-10
- Ethernet
  - ARP, 7-3
  - autonegotiation, 7-3
  - autosensing, 7-3
  - bridging capability, 15-3
  - cabling, 7-4
  - Card profile settings, 7-9
  - configuration guide, 7-1
  - configuration overview, 7-6
  - controlled-load, 7-4
  - dump profile, 7-14
  - dump settings in card profile, 7-11
  - flow control, 7-3
  - implementation, 7-2
  - interface name, 7-5
  - large route tables, 7-4
  - LLC/SNAP support, 7-3
  - load profile, 7-13
  - logical interfaces, 7-5
  - maint commands, 7-18
  - physical interfaces, 7-5
  - proxy ARP, 7-3
  - selective packet discard, 7-2, 7-16
  - set ICMP in card profile, 7-10

- set run-time code in card profile, 7-10
- settings in Card profile, 1-25
- support for IS-IS protocol, 7-2
- transparent bridging, 7-4
- external flash device
  - configure for dumps, 2-7
  - configure for local logging, 2-7
  - support for dumps at panic, 2-6
  - use to duplicate configurations, 2-45

## F

fast Ethernet, see Ethernet

### FDDI

- bridging capability, 15-3
- chart of interface numbering, 4-4
- configuration, 4-2
- configuring SAS and DAS, 4-17
- connector keys, 4-18
- controlled-load, 4-3
- display interface statistics, 4-26
- dual attach A and B ports, 4-16
- dual homing, 4-21
- dump profile, 4-13
- dump settings in card profile, 4-10
- FDDI-MIB, 2-22
- functions, 4-2
- GRF interface name, 4-5
- large route table support, 4-2
- load profile, 4-11
- logical addresses, 4-4
- maint commands, 4-23
- master and slave ports, 4-16
- MTU, 4-5
- optical bypass, 4-19
- physical interface numbers, 4-5
- proxy ARP, 2-40, 4-3
- reset an individual interface, 4-29
- selective packet discard, 4-2, 4-21
- set ICMP in card profile, 4-7
- set optical bypass in card profile, 4-9
- set run-time code in card profile, 4-10
- set SAS/DAS in card profile, 4-8
- support for IS-IS protocol, 4-2
- transparent bridging, 4-3
- unused physical interfaces, 4-17

### FDDI/Q

- description, 4-2

### fields

- as complex structures, 1-19
- in profiles, 1-18

filterd, filter daemon, 11-2

- starting up, 11-14

filterd.conf file, 11-2, 11-15

### filtering

- and Controlled-Loading, 14-3
- applying a direction, 11-7
- applying an action, 11-8, 14-3
- binding filters, 11-6
- binding options, 11-2
- changing on the fly, 11-14
- configuration process, 11-14
- controlling access to RMS, 11-12
- defining filters, 11-3
- filter daemon, 11-2
- into me, 11-7
- logging loop, 11-10
- logical interface number, 11-6
- maint commands, 11-21
- packet header logging, 11-9
- rules, 11-3
- states, 11-8
- vlif, 11-6

flags,in bridging, 15-21

### flow control

- Ethernet, 7-3

fragmentation, 2-15

fragmentation, in bridging, 15-4

### Frame Relay, 6-2

- autoadding PVCs, 6-28
- configuring on HSSI, 6-26
- DLCI numbering, 6-29
- link parameters, 6-27
- MTU, 6-2
- nonbroadcast interfaces (NBMA), 6-26
- number of logical interfaces, 6-5
- with IS-IS, 13-6

### Frame Relay, SONET OC-3c

- autoadding PVCs, 8-26
- configuring, 8-24
- DLCI numbering, 8-27
- link parameters, 8-25
- nonbroadcast interfaces (NBMA), 8-24

Framing protocols, on HSSI, 6-2

### ftp

- obtaining HIPPI standards, RFCs, 5-10

## G

### GateD

- "passive" interface, 12-11
- BGP confederations, 12-7, 12-8
- communities, 12-10
- configuration overview, 2-36
- configuring BGP, 12-1
- configuring static routes, 2-34
- GateD State Monitor (GSM), 12-8
- gdc commands, 2-36

- limit on trace/log file size, 2-6, 2-36
- MEDs, 12-9
- multiple BGP groups, 12-11
- route preference biasing, 12-11
- route reflection, 12-10
- routing arbiter interaction, 12-10
- starting and reconfiguring, 2-36
- static statement, 2-34
- using an alias as routerid, 2-36
- gateway mode, ATMP, 9-5
- gdc commands (GateD), 2-36
- get command
  - used with a profile, 1-40, 1-41, 1-43
- grarp command
  - example, 5-21
  - function, 5-9
  - in HIPPI-HIPPI IP routing, 5-20
- grarp.conf file
  - in configuring PVCs, 10-18
- gratm command
  - ATM OC-3c on-the-fly PVCs, 3-24
- GRE (Generic Routing Encapsulation), AT-  
MP, 9-2
- grfr command
  - creating a PVC on-the-fly, 6-30, 8-28
- grifconfig.conf file
  - entry for ISO address (IS-IS), 13-3, 13-7
- GR-II
  - config\_netstart script, 2-49
  - installing configuration files, 2-50
  - RADIUS authentication, 2-26
- grinchd.conf, replaced by CLI profiles, 1-8
- grlmap command
  - in HIPPI-HIPPI IP routing, 5-24
  - in IP routing example, 5-20
  - in logical address example, 5-16
  - mapping logical addresses, 5-9
- group type, BGP, 12-5
- grppp command
  - how to display PPP status, 6-34
- greset command
  - to install grifconfig.conf, 2-16
  - to install grroute.conf, 2-31
- grrt command
  - example of data returned, 2-33
- grsnapshot command
  - using to test a new configuration, 2-43, 2-44
- grwrite command, 2-5, 2-9
  - using to test a new configuration, 2-43, 2-44
- GSM
  - viewing GateD route tables, 2-35

- GSM, GateD State Monitor, 12-8
- gx0yz interface name, 2-13

## H

- H0 HIPPI option, 5-26
- halt system
  - grms command, 2-51
- HDLC, 6-2
  - configuring on HSSI, 6-23
  - configuring on SONET OC-3c, 8-22
  - fields in Card profile, 1-24
  - keepalive settings in Card profile, 1-24
  - keepalive settings, HSSI, 6-24
  - keepalive settings, SONET OC-3c, 8-23
  - MTU, 6-2
  - number of logical interfaces, 6-5
  - with IS-IS, 13-6
- help, on-line
  - displaying CLI command usage, 1-12
- HIPPI
  - bridging, 5-10
  - configuration options explained, 5-11
  - dump profile, 5-34
  - dump settings in Card profile, 5-32
  - establishing a connection, 5-3
  - HIPPI-MIB, 2-22
  - host time-out values, 5-31
  - IBM H0 HIPPI, 5-26
  - I-field, 5-3
  - IP connection/routing, 5-2, 5-8
  - IP routing example, HIPPI-IP, 5-22
  - IPI-3 routing, 5-25
  - load profile, 5-33
  - logical address, 5-5, 5-6
  - logical address example, 5-14
  - maint commands for, 5-36
  - MTU, 5-10
  - raw mode, 5-2, 5-8
  - routing to bridge group, 5-10
  - set ICMP in Card profile, 5-30
  - set run-time code in Card profile, 5-31
  - settings in Card profile, 1-25
  - source routing, 5-4
  - source routing example, 5-12
- HIPPI switch example, 5-18
- HIPPI-SC, 5-7
  - obtaining ANSI standard, 5-10
- HIPPISW-MIB, 2-22
- host I-field
  - in IP routing, 5-19, 5-23
  - in logical addressing, 5-16
  - in source routing, 5-13

- hostname
  - hostname command, 2-17
  - need to change GRF hostname, 2-17
  - ways to set, 2-17
- HSSI
  - assign protocol per port, 6-3
  - configuring Frame Relay, 6-26
  - configuring HDLC, 6-23
  - configuring IS-IS, 13-6
  - configuring Point-to-Point, 6-32
  - connection options, 6-7
  - controlled-load, 6-4
  - dump profile, 6-19
  - dump settings in card profile, 6-17
  - Frame Relay, 6-2
  - framing protocols, 6-2
  - HDLC, 6-2
  - implementation specs, 6-2
  - in ATMP tunneling, 9-17
  - internal clock source, 6-7
  - large route table support, 6-4
  - load profile, 6-18
  - maint commands, 6-38
  - media card configuration steps, 6-11
  - multiple DLCIs per interface, 6-30
  - null modem cabling, 6-7
  - Point-to-Point, 6-2
  - selective packet discard, 6-4, 6-21
  - set framing protocol in card profile, 6-13
  - set ICMP in card profile, 6-15
  - set run-time code in card profile, 6-16
  - set source clock in card profile, 6-14
  - setting CRC bits, 6-7
  - support for IS-IS protocol, 6-3
- I**
- IBM, H0 HIPPI, 5-26
  - 3090 connectivity, 5-26
- ICMP throttling
  - settings in Card profile, 1-24
- I-field, 5-3
  - camp-on bit, 5-3
  - direction bit, 5-7
  - in IP routing, 5-9, 5-19, 5-20, 5-23
  - in logical addressing, 5-16
  - in source routing, 5-12
  - mapping to IP address (grarp), 5-9
  - path selection bits, 5-4
  - role in HIPPI, 5-8
- InATMARP, ATM OC-12c, 10-18, 10-19
- InATMARP, ATM OC-3c, 3-21, 3-22
- index
  - as profile name, 1-12
  - as used with profiles, 1-17
  - profile definition of, 1-12
  - to a profile, 1-16
- installing configuration files, 2-50
- Integrated Services
  - Controlled-Load, 14-2
- interface
  - passive in GateD, 12-11
- interface name
  - ATM OC-3c, 3-8
  - Ethernet, 7-5
  - FDDI, 4-5
  - how to create, 2-13
  - HSSI, 6-5
  - SONET OC-3c, 8-5
- interfaces, configuring, 2-13
- internal clock source
  - ATM OC-3c, 3-6
  - HSSI, 6-7
- Internet address
  - in grifconfig.conf, 2-14
- into me filtering, 11-7
- inverse ARP
  - ATM OC-3c, 3-6, 3-22
- inverse ARP, ATM OC-12c, 10-3, 10-19
- IP address
  - assigned to FDDI logical address, 4-4
  - in grifconfig.conf, 2-14
  - mapping HIPPI I-field to, 5-9
  - where assigned, 2-13
- IP datagram, 5-9
- IP packet filtering, 11-2
- IP routing
  - and HIPPI I-field, 5-9
  - filtering, 11-2
  - HIPPI-IP example, 5-22
  - loose source routing option, 2-38
  - supernet address look-up, A-10
  - using subnet masks, A-1
- IP routing, configuration tasks, 2-13
- IP service ports, 11-5
- IPI-3 routing, 5-25
- IP/SONET
  - see SONET OC-3c
- IS-IS
  - configuration, 13-2
  - configuration examples, 13-8
  - configuration statement, 13-2, 13-4
  - configuring on ATM/Q interfaces, 13-6
  - configuring on HSSI interfaces, 13-6
  - define a systemid, 13-3

- define an area, 13-3
- define an ISO address, 13-3
- description, 13-2
- designated router example, 13-11
- ISO address in grifconfig.conf, 2-16
- IS-IS support
  - ATM OC-12c, 10-2
  - ATM OC-3c, 3-3
  - Ethernet, 7-2
  - HSSI, 6-3
  - SONET OC-3c, 8-3
- ISO address (IS-IS), 2-16, 13-3
- ISO/CLNP
  - see IS-IS

## **K**

- keepalive settings in Card profile, Cisco-HDLC, 1-24
- keepalive settings, HSSI HDLC, 6-24
- keepalive settings, SONET HDLC, 8-23
- keys, for FDDI connectors, 4-18

## **L**

- level 1 and 2, in IS-IS, 13-4
- link parameters, Frame Relay
  - HSSI, 6-27
  - SONET OC-3c, 8-25
- Link parameters, PPP, 6-33
  - grppp status returns, 6-34
- list command
  - used with a profile, 1-40, 1-42
- LLC/SNAP support
  - Ethernet, 7-3
- lo0, loopback interface, 2-15
  - routerid for GateD, 2-36
- load command
  - used with a profile, 1-49
- load profile
  - ATM OC-12c configuration, 10-14
  - ATM OC-3c configuration, 3-17
  - definition of, 1-17
  - drawing of levels, 1-31
  - Ethernet configuration, 7-13
  - FDDI configuration, 4-11
  - HIPPI configuration, 5-33
  - how referenced, 1-17
  - HSSI configuration, 6-18
  - SONET OC-3c configuration, 8-17
- local logging, to external flash, 2-7

- log on
  - GRF 400, 1-2
  - non-privileged for grms command, 2-51
  - system using RMS node, 1-5
- logging loop, in filtering, 11-10
- logging server
  - setting up, 2-9
- logging to external flash device, 2-7
- logging, in header filtering, 11-8
- logging, size of Gated log file, 2-6, 2-36
- logging, system options, 2-6
- logical address
  - FDDI, 4-4
  - HIPPI, 5-5, 5-6
  - HIPPI example, 5-14
- logical interface
  - ATM OC-12c, 10-6
  - number for ATM OC-3c card, 3-8
  - virtual ATM circuits per, 3-8
- logical interfaces
  - assign in grifconfig.conf, 2-5, 2-13
  - for Frame Relay, 6-5
  - for HDLC, 6-5
  - for PPP, 6-5
  - on Ethernet card, 7-5
  - on HSSI card, 6-5
  - on SONET OC-3c, 8-5
- logical ring (FDDI), 4-16
- logs
  - examples in Getting Started manual
  - how to enable remote logging, 2-9
  - sending to external flash device, 2-7
- longest match, in subnetting, A-12
- loopback alias
  - configure for GateD, 2-36
  - how to set up, 2-15
  - ifconfig command, 2-37
- loose source routing (IP option), 2-38
- ls command
  - used with a profile, 1-40, 1-43

## **M**

- maint command
  - for ATM OC-12c media cards, 10-26
  - for ATM OC-3c media cards, 3-39
  - for Ethernet media cards, 7-18
  - for HIPPI media cards, 5-36
  - for HSSI media cards, 6-38
  - for SONET OC-3c media cards, 8-34
  - set for filtering, 11-21
- mask, in filtering, 11-4

- master/slave ports (FDDI), 4-16
- maximum burst size, 3-28, 3-29, 3-34, 10-20, 10-22, 10-25
- MBS, 3-28, 10-20
- MEDs, 12-9, 12-14
  - originating in BGP, 12-14
- MIBs
  - status of SMT MIB variables, 4-28
- mode, SONET OC-3c, 8-14
- MTU
  - ATM OC-12c, 10-2
  - ATM OC-3c, 3-6
  - discovery facility, 2-15
  - Ethernet, 7-3
  - FDDI, 4-5
  - Frame Relay, 6-2, 8-2
  - HDLC, 6-2, 8-2
  - HIPPI, 5-10
  - media/protocol defaults, 2-15
  - PPP, 6-3, 8-3
  - specifying in grifconfig.conf, 2-15

## N

- NBMA, Frame Relay interface, 6-26
- NBMA, Frame Relay SONET interface, 8-24
- netmask, 2-14
  - applying to IP address, A-5
  - explicit, A-3
  - implicit, A-2
  - in grifconfig.conf, 2-14
  - support for explicit masks, A-5
- NETSTAR-MIB, 2-22
- netstart
  - how to run config\_netstart, 2-49
- netstat command
  - used for bridging information, 15-18
- network logging, 1-2
  - setting up remote server, 2-9
  - via configuration script, 2-9
- new command
  - used with a profile, 1-50
- next-hop address, 2-31
- NFS file system
  - set up for GRF, 2-12
- NMBA interface, 2-14
- NSAP addresses
  - how built, 3-25
- NSEL parameter, in ISO address, 13-3
- null modem cabling, HSSI, 6-7
- null modem cabling, SONET OC-3c, 8-7

## O

- on-the-fly PVCs
  - HSSI Frame Relay, 6-30
  - SONET OC-3c Frame Relay, 8-28
- optical bypass
  - setting in Card profile, 1-24
- optical bypass switch (FDDI), 4-19
  - enable via software, 4-19
- OSPF
  - BGP interaction, 12-6
  - broadcast of ATMP addresses, 9-15
  - explicit mask support, A-5
- overview, system configuration, 2-3
- overwriting of etc/services file, 2-41

## P

- packet
  - MTU and fragmentation, 2-15
  - selective packet discard, 8-20, 10-4
  - selective packet discard, ATM OC-3c, 3-37
  - selective packet discard, Ethernet, 7-16
  - selective packet discard, FDDI, 4-21
  - selective packet discard, HSSI, 6-21
- packet buffering
  - ATM OC-3c, 3-5
- packet header logging, in filtering, 11-9
- paged line output, in CLI, 1-16
- panic dumps
  - external flash support for, 2-6
- passwd command, 1-4, 1-6
- passwords, how to set in User profile, 1-12
- path attributes, BGP, 12-2
- path selection bits (HIPPI I-field), 5-4
- path selection, BGP, 12-5
- payload identifier, SONET OC-3c, 8-14
- PCMCIA disk device, installing, 2-7
- PCR (peak cell rate), 3-28, 10-20
- peak cell rate, 3-28, 3-34, 10-20, 10-25
- permission levels
  - in user profile, 1-8
- permissions
  - where to set, 1-12
- physical interfaces
  - ATM OC-12c card, 10-6
  - ATM OC-3c card, 3-8
  - FDDI, 4-4
  - HSSI card, 6-5
  - on Ethernet card, 7-5
  - on SONET OC-3c, 8-5



- ping command
    - during connectivity testing, 2-52
  - Point-to-Point, 6-2
    - configuring on HSSI, 6-32
    - configuring on SONET, 8-3, 8-30
    - IPCP parameter, 6-34, 8-31
    - LCP parameters, 6-33, 8-30
    - LQR parameters, 6-34, 8-31
    - MTU, 6-2
    - number of logical interfaces, 6-5
    - option negotiation parameters, 6-33, 8-30
    - status returned by grppp command, 6-34
  - point-to-point, destination address, 2-14
  - policy
    - BGP role, 12-5
    - preventing loss of, 12-11
  - PPP MIB, 2-22
  - PPP, see Point-to-Point, 6-2
  - priority, of ATM OC-12c rate queue, 10-21
  - priority, of ATM OC-3c rate queue, 3-31
  - profile index, definition, 1-17
  - profiles
    - accessing (reading), 1-39
    - Card profile, 1-21
    - changing a field-value, 1-45
    - choosing get or list, 1-40
    - deleting a profile, 1-46
    - display the list of, 1-39
    - Dump profile, 1-27
    - field structure, 1-18
    - how to create new, 1-50
    - how to save alternate versions of, 1-47
    - how to use, 1-38
    - introduction, 1-17
    - Load profile, 1-31
    - looking at profile A while in profile B, 1-41
    - management commands, 1-38
    - names of types, 1-17
    - System profile, 1-33
    - type definitions, 1-17
    - types, 1-17
    - User profile, 1-35
    - writing a change (to save), 1-46
  - prompts
    - in the CLI, 1-12
  - protocols
    - ATMP (Ascend Tunnel Management Protocol), 9-2
  - proxy ARP
    - on Ethernet media card, 7-3
    - on FDDI media card, 2-40, 4-3
  - pvcatmp, in ATMP, 9-15, 9-19
  - PVCs, ATM OC-12c, 10-8
    - configuration overview, 10-18
    - per logical interface, 10-8
    - traffic shaping parameters, 10-20
  - PVCs, ATM OC-3c, 3-10
    - configuration overview, 3-21
    - per logical interface, 3-10
    - reconfiguration on-the-fly, 3-24
    - traffic shaping parameters, 3-28
  - PVCs, Frame Relay, 6-28, 6-29
    - configuration, 6-28
    - creating a PVC on-the-fly, 6-30
    - multiple DLCIs / interface, 6-30
    - parameters, 6-29
  - PVCs, SONET OC-3c Frame Relay, 8-27
    - configuration, 8-27
    - creating a PVC on-the-fly, 8-28
    - multiple DLCIs / interface, 8-29
    - parameters, 8-27
  - pwd, using in a profile, 1-44
- ## Q
- Q cards
    - ATM/Q, 3-2
    - Ethernet, 7-3
    - FDDI/Q, 4-2
    - SONET OC-3c, 8-4
  - QOS, 3-30, 3-31, 10-21
  - quality of service, 3-28, 3-31, 10-20, 10-21
- ## R
- RADIUS
    - client support for, 2-26
    - fields in User profile, 2-27
  - RADIUS profile, in ATMP, 9-12
  - rate queues, and QOS, 3-30, 10-21
  - raw HIPPI, HIPPI-SC, 5-2, 5-25
  - read command
    - used with a profile, 1-39
  - rebooting, saving configuration before, 2-5
  - remote logging, 2-9
    - avoid by using external flash, 2-7
    - how to set up, 2-9
  - RFC 1213, 2-22
  - RFC 1227, 2-23
  - RFC 1473, 2-22
  - RFC 1512, 2-22, 2-23
  - RFC 1573, 2-23
  - RFCs for HIPPI, 5-10

- RMS, filtering access to, 11-12
- root
  - adding users as, 1-50
- root bridge, 15-22
- root path cost, in bridging, 15-22
- route advertisements, BGP, 12-4
- route flapping
  - weighted route dampening, 12-23
- route preference biasing, 12-11
- route reflection, 12-10
  - configuring, 12-17
- route server, and GateD, 12-10
- route tables
  - data returned by grrt command, 2-33
  - for bridging, 15-18
  - GateD, access via GSM, 2-35
- route tree, bridging, 15-18
- routerid, configuring for GateD, 2-36
- routing arbiter interaction, 12-10
- run-time code
  - settings in Card profile, 1-22

## **S**

- SAS
  - setting in Card profile, 1-24
- SAS settings (FDDI), 4-4
  - in Card profile, 4-17
- save command
  - used with a profile, 1-47
- SCR, 3-28, 3-29, 10-20, 10-22
- script
  - first-time power on, 1-2, 2-9
  - initial configuration, 1-5
- SDH mode, setting, 10-3
- SDH mode, setting ATM OC-3c, 3-5
- secondary address (alias), configuring, 2-15
- securID
  - client support for, 2-28
  - fields in User profile, 2-30
- selective packet discard
  - and Controlled-Load, 14-2
  - ATM OC-3c, 3-37
  - FDDI, 4-21
  - HSSI, 6-4
- service ports, on the RMS, 11-5
- sessions, BGP, 12-4
- set command
  - display command usage, 1-45
  - multiple set commands, 1-45
  - used with a profile, 1-45
- setver command
  - use to test a new configuration, 2-43, 2-44
- shutdown command, 2-51
- signaling, ATM OC-3c, 3-4
- slave/master ports (FDDI), 4-16
- SNMP
  - configuration steps, examples, 2-19
  - numbering system, 4-5
  - problems viewing large route tables, 2-21
  - support on GRF, 2-22
  - traps supported, 2-23
- snmpd, 15 second time-out, 2-21
- SONET
  - settings in Card profile, 1-24
- SONET mode, setting, 10-3
- SONET mode, setting ATM OC-3c, 3-5
- SONET OC-3c
  - APS, mode, clock, payload settings, 8-14
  - clock source, 8-14
  - configuration, 8-1
  - configuring Frame Relay, 8-24
  - configuring HDLC, 8-22
  - configuring Point-to-Point, 8-30
  - controlled-load, 8-4
  - dump profile, 8-18
  - dump settings in card profile, 8-16
  - implementation, 8-2
  - interface name, 8-5
  - load profile, 8-17
  - logical interfaces, 8-5
  - maint commands, 8-34
  - media card configuration steps, 8-9
  - mode setting, 8-14
  - multiple DLCIs per interface, 8-29
  - null modem cabling, 8-7
  - payload identifier, 8-14
  - physical interfaces, 8-5
  - PPP implementation, 8-3
  - selective packet discard, 8-4, 8-20
  - set framing protocol in card profile, 8-12
  - set ICMP in card profile, 8-13
  - set run-time code in card profile, 8-15
  - set selective packet discard, 8-13
  - support for IS-IS protocol, 8-3
- source clock, setting in Card profile, 1-25
- source routing (HIPPI), 5-4
  - example, 5-12
- spanning tree, in bridging, 15-4
- static IP routing, 2-31
- static routes
  - configuration examples, 2-32
  - configuring via grroute.conf, 2-31
  - configuring via route add, 2-32
  - configuring with GateD, 2-34

subnet masks, A-1  
    how to use, A-1, A-9  
    router look-up process, A-9  
    variable-length, A-3  
SUNI clock (ATM OC-3c), setting, 3-6  
SUNI clock, setting, 10-4  
supernet address, A-5  
supernetting, A-4  
    address look-up, A-10  
    basic example, A-4  
    forming a supernet address, A-8  
    in typical routing, A-6  
    router "matching", A-12  
sustainable cell rate, 3-28, 3-29, 3-34, 10-20,  
    10-22, 10-25  
SVCs, ATM OC-3c, 3-10  
    configuration example, 3-35  
    configuration overview, 3-25  
    per logical interface, 3-10  
    traffic shaping parameters, 3-28  
syslog server  
    how to set up, 2-9  
syslogd  
    remote logging server, 2-9  
system profile  
    definition of, 1-17  
    drawing of levels, 1-33  
    how referenced, 1-17  
systemid, in IS-IS, 13-3

**T**

telnet  
    enabling sessions, 2-18  
test configurations  
    how to do safely, 2-43, 2-44  
testing connectivity, 2-52  
touch command, use on syslog server, 2-10  
traffic shaping, ATM OC-12c, 10-20  
    traffic\_shape names, 10-24  
traffic shaping, ATM OC-3c, 3-28  
    traffic\_shape names, 3-33  
transparent bridging  
    as implemented on FDDI, Ethernet, 15-2  
    on Ethernet media card, 7-4  
    on FDDI media card, 4-3  
tunnels, configuring ATMP, 9-2  
typing shortcuts, CLI, 1-15

**U**

UNI signaling, 3-4, 3-10  
unreachable messages, BGP, 12-4  
updating software  
    changes to grclean.logs.conf, 2-6, 2-8  
    grwrite to save changes, 2-5  
user profile  
    definition of, 1-17  
    drawing of levels, 1-35  
    how referenced, 1-17  
    RADIUS fields, 2-27  
    securID fields, 2-30  
user validation  
    via RADIUS, 2-26  
    via securID, 2-28

**V**

variable-length subnet masking, A-3  
VCI, ATM OC-12c, 10-7  
VCI, ATM OC-3c, 3-9  
virtual circuits, ATM OC-12c, 10-7  
    virtual circuit identifier (VCI), 10-7  
virtual circuits, ATM OC-3c, 3-9  
    virtual circuit identifier (VCI), 3-9  
virtual path, ATM OC-12c, 10-7  
virtual path, ATM OC-3c, 3-9  
virtual private networks, ATMP, 9-5  
    private address space, 9-5  
vlif, in filtering, 11-6  
VPI, ATM OC-12c, 10-7  
VPI, ATM OC-3c, 3-9  
VPI/VCI, ATM OC-12c, 10-8  
VPI/VCI, ATM OC-3c, 3-10

**W**

warranty description, B-1  
weighted route dampening  
    in GateD, 12-23  
    statement in GateD, 12-23  
whereami command  
    used with a profile, 1-41  
    using in a profile, 1-44  
write command  
    used with a profile, 1-46  
writing/saving config changes, 2-5

