IBM

Disaster Recovery Using HAGEO and GeoRM

- High availability through use of geographic clusters
- Optimize performance for geographic mirroring
- Step-by-step installation of HAGEO and GeoRM

Ole Conradsen
Cyril Collin
Fabio Sabatti
Junbiao Wang

Redbooks

ibm.com/redbooks

SG24-2018-01

International Technical Support Organization

Disaster Recovery Using HAGEO and GeoRM

May 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special notices” on page 379.

Second Edition (May 2000)

This edition applies to HAGEO and GeoRM Version 2.1 for use with AIX release 4.3.3 as described in announcement letter *5765-A25 IBM High Availability Geographic Cluster for AIX.*

This document was created or updated on May 23, 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Tables

Preface

This redbook is an essential guide to planning and implementing a successful HAGEO cluster with HAGEO or GeoRM Version 2.1. Besides providing the information and insights gained during the residency held at the International Technical Support Organization, Austin Center, it is a valuable supplement to the product documentation. In addition to discussing the network types available for geographic mirroring, this redbook provides tested examples, hints, tips, and answers to frequently-asked questions. The C source code and scripts listed in appendixes A through D are available on the ITSO Web page (see Appendix E.4, “Referenced Web sites” on page 384, for details).

Using the information provided in this book, system engineers, service providers, and customers can feel confident that they possess the technical insights necessary to plan and successfully implement an HAGEO cluster.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Austin Center.

Ole Conradsen is a Project Leader at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on all areas of AIX. Before joining the ITSO, he worked with AIX projects in IBM Denmark for 11 years. Ole holds a Master of Science degree in Electrical Engineering from The Technical University of Denmark.

Cyril Collin is an I/T Specialist at IBM France. He has seven years of experience with AIX and four years with HACMP. His areas of expertise include high availability, performance issues, and the SP system.

Fabio Sabatti is an I/T Specialist working for the Business Continuity and Recovery Service at IBM Italy. He has worked at IBM for four years. His areas of expertise include IBM AIX, IBM OS/2, IBM Warp Server, IBM PC/Netfinity Server, IBM RS/6000, and SP. He is an IBM Certifed Professional Server Specialist, Certified System Expert in OS/2 Warp, and Certified Novell Administrator.

Junbiao Wang is an Advisory I/T Specialist in IBM China, Shanghai Branch. He has five years of experience in the AIX and HACMP/HAGEO fields. He holds an IBM AIX Advanced Expert certification and a masters degree in Computer Science and Engineering from Suzhou University. His areas of expertise include CICS/6000 and ADSM.

Thanks to the following people for their invaluable contributions to this project:

Cindy Barrett, IBM Austin

Bernhard Buehler, IBM Germany

Michael Coffey, IBM Poughkeepsie

John Easton, IBM High Availability Cluster Competency Centre (UK)

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 393 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction to HAGEO and GeoRM

This chapter will introduce you to the concepts of high availability and disaster recovery, and then it will introduce the main features of GeoRM V2.1(Geographics Remote Mirroring) and HAGEO V2.1 (High Availability Geographics Cluster), which are IBM products for AIX platforms.

1.1 Introduction to high-availability and disaster recovery

Today, keeping a business operational increasingly means keeping critical data and information systems available around-the-clock. To compete successfully in today's global markets, companies are striving to protect critical information systems to help minimize costly business impacts, such as lost sales, loss of customer satisfaction, and loss of employee productivity.

Developing and deploying a business continuity plan is not cheap, but ignoring the issue can be even more costly. Table 1 shows the hourly cost by industry sector caused by service downtimes (source: Contingency Planning Research, Inc.).

Table 1. Cost of unplanned computer system outages by industry

Industry	Downtime cost per hour
Retail Brokerage	$6.45 M
Credit Card Sales Authorization	$2.6 M
Infomercial/ 800-Number Promotions	$200 K
Catalog Sales Center	$90 K
Airline Reservations	$90 K
ATM Service Providers	$15 K

For this reason, the I/T market offers many different solutions to keep critical data and applications continuously available. *Availability* is the degree to which the system continues to let you do your work. Many factors affect a system's level of availability. These factors range from the reliability of the hardware and software to the thoroughness of the training of the system administrators and the design of procedures for maintenance tasks.

But, to reach the *high availability* of your I/T system, you need to perform some steps to eliminate the single points-of-failure in your environment as well as design solutions to quickly recover the services in case of failure.

1.1.1 Fault tolerance concept

To reduce the number of single points of failure, new technologies have been developed to have *fault tolerant systems*. Fault tolerant systems rely on specialized hardware and software to detect hardware failures and instantly continue workload on redundant components built into the system. A fault-tolerant system is seen as one logical system entity. There is one system running one instance of operating system software and one instance of any application. Within that system, there is a complete set of redundant hardware components: Redundant memory, processors, buses, disks, network adapters, and so on. If any component fails, its redundant partner continues. The active processes are preserved and continue; so, there is no perceived impact on the users. While this lack of impact is an attractive advantage, there is a price to be paid. Since all redundant components are included in a single system entity, the redundant components cannot be used for any independent workload in non-failure conditions. There is another drawback to fault-tolerant systems: Since they are constructed of specialized hardware and software, they are too expensive for many businesses to justify. This cost becomes even higher if you consider it in terms of price and performance because the redundant hardware does not provide any application performance benefits.

1.1.2 Highly-available systems

Highly-available systems are made up of off-the-shelf components. A cluster is made up of several independent systems that share resources, such as disks and application accesses; so, if one system fails, the other can take over its function or applications in a mode that is transparent to the clients.

1.1.3 Distributed systems

Recent years have seen the proliferation of distributed systems in which the workload is spread over multiple systems in a network. This approach provides the benefit of being able to spread the workload of multiple applications over a number of systems. In availability terms, this has the benefit of avoiding the “all your eggs in one basket” scenario. Since all the applications are not housed on one server system, the failure of one server will not affect the users of other applications. Also, by having more than one server in place, it is possible to manually reconfigure hardware and software to move an application and data to another system if one should fail. Compared to fault-tolerant systems, distributed systems require planning and manual effort and offer a much slower recovery time. Another drawback of distributed systems is that, with more hardware systems involved, the probability of at least one component failing increases.

1.1.4 Cluster solutions

Cluster solutions, such as IBM High Availability Cluster Multiprocessing (HACMP), combine the best of both approaches: Fault tolerance at the device level and distributed systems at the application level. A well-designed high-availability cluster includes redundant components, such as disk drives, disk adapters, and network adapters. While there may be multiple servers, each can perform its own workload. Unlike a distributed environment, if a server fails because of hardware or software failure, a surviving node will automatically reconfigure to take over the resources and applications of the server.

With HACMP, as many as eight RS/6000s can be configured in a cluster, and the workload is distributed among the servers. If a server fails, another server will acquire its resources and quickly make them available to clients. Unlike fault-tolerant systems, all processors can actively perform work in normal operations. Unlike distributed systems, the work of a failed processor is automatically redistributed to the surviving cluster members without manual intervention.

A cluster is made up of several independent systems, called nodes, that share resources, such as disks and application access; so, if a node fails, a redundant node can take over its function. High availability is *not* fault tolerance. If a node fails, clients will lose access to system resources until the cluster recovers. However, recovery time can be relatively short. Many things impact recovery time, but resources are typically available within minutes. Figure 1 on page 4 shows an example of an HACMP cluster solution.

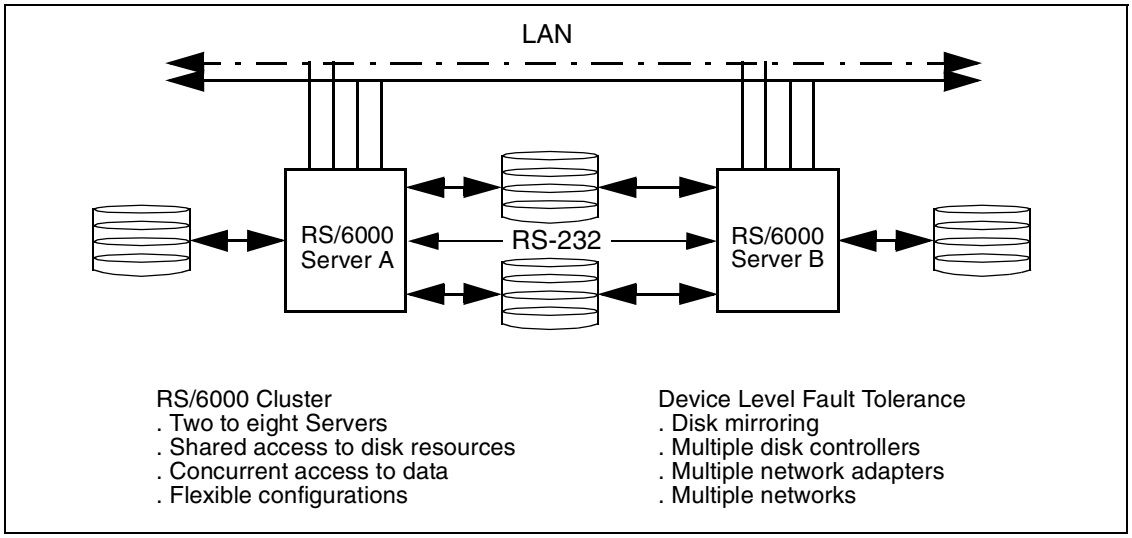

Figure 1. Example of an HACMP cluster solution

1.1.5 Disaster recovery

Is a high-availability fault-tolerant system enough to safeguard your business data and applications? Data must not only be available, but *secure*. It must be backed up, copied, and safely stored in case something happens and the current working copy becomes unavailable or is lost. Most enterprises back up data daily, if not more often. This helps to protect data from classic user errors. But then, where is it stored? How long is the copy in the same building, the same campus, or the same town as the working data? What if a major disaster, such as a fire or flood, wipes out the site where both the server running the application and the backup data are located? How much data is lost when something happens and you have to restore data using the backup copy? A day's worth of transactions? How valuable is the lost data? How valuable is the downtime required to restore the data?

Figure 2 on page 5 shows the most frequent causes of data loss or service stoppages. These causes can affect the infrastructure availability, the computer room, or a building complex.

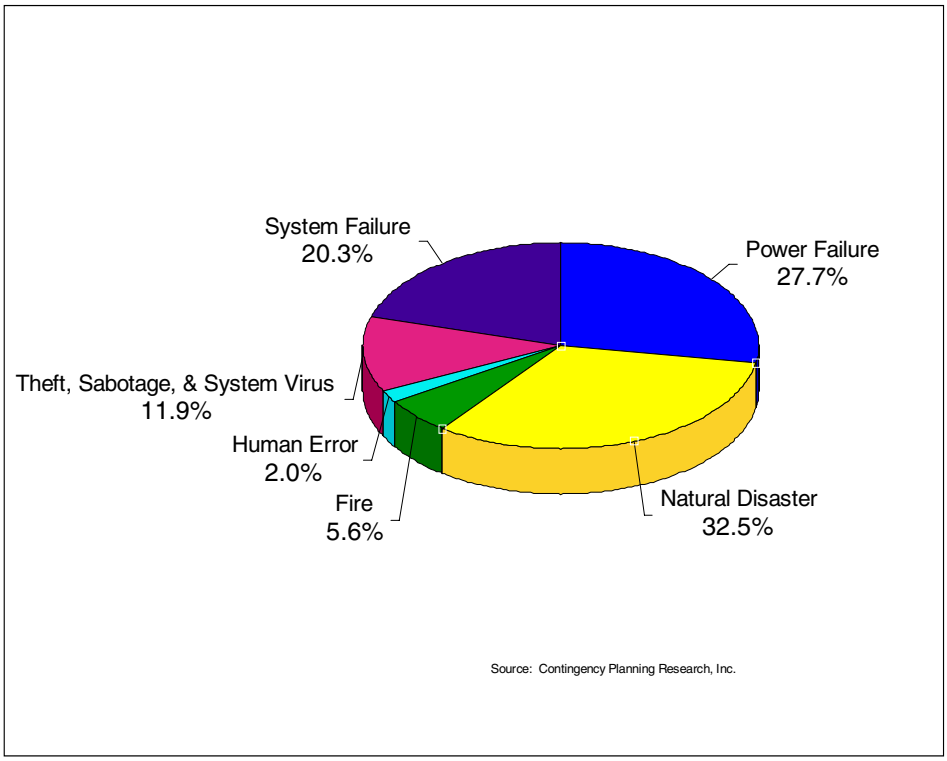

Figure 2. Causes of computer loss across all industries

Studies by IBM and others show that nearly half the companies that took longer than four to five days to get their information systems back on line were forced out of business entirely. Communications and financial services enterprises can hardly afford few hours off line, and for sure not days. The following Web sites describe recent disaster studies:

- `http://www-1.ibm.com/services/continuity/recover1.nsf/files/Downloads/$file/buscont.pdf`
- `http://www.drj.com/special/stats/tari.htm`
- `http://www.fema.gov/library/lib01.htm`

In case of a disaster, normal procedures and products that provide security and high availability for IT services are of no use. It is necessary to have a *disaster recovery* strategy in place so that, after a significant large-scale interruption in service, the company can reinstate its computers and their programs in an alternative recovery location in a quick data-reliable way.

The individual disaster recovery strategy design for a particular IT environment will often be a compromise between a series of conflicting objectives. These objectives include:

- Effective control and management of the second site
- Quick takeover (readiness of second site)
- Isolation (distance between sites)
- High-bandwidth interconnection technology
- Cost limitations

In summary, there are many conflicting disaster recovery design objectives; consequently, an individual solution is always a compromise.

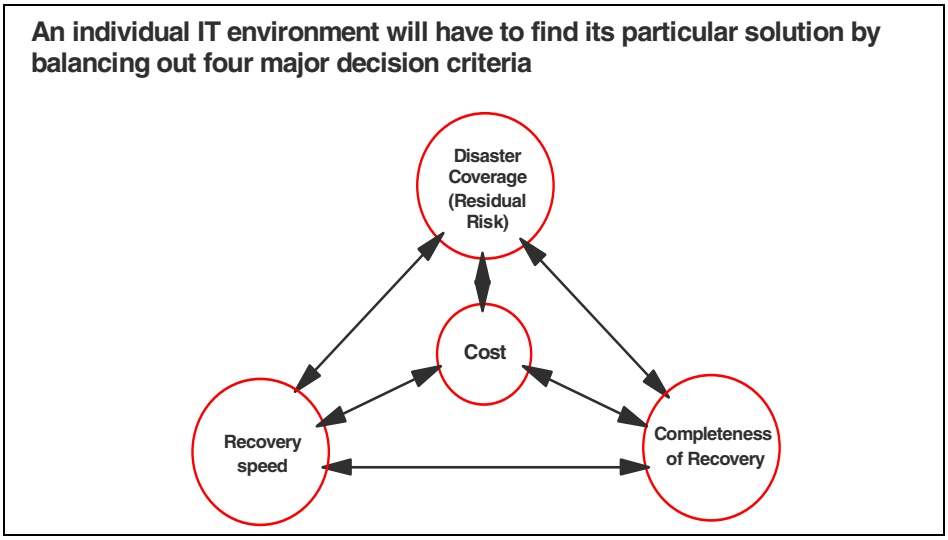

Figure 3. Disaster recovery design objectives

1.1.5.1 Off-site removable media storage

Tape or CD-ROM backup and restore are the most commonly-used methods of location disaster recovery. Systems are typically backed up nightly or every other day with the media being securely stored at an off-site location. Note the following characteristics of off-site removable media storage:

- It provides data protection from human or application errors.
- The reliability of the data depends on the last available backup.

- It necessitates the physical transportation of media from one location to another. As long as data remains at the primary location, it is exposed and may be lost in the event of a disaster.
- The time required to reload the data depends on the amount of data; recovery time can range from hours to days.
- The media can be of low reliability.

1.1.5.2 Remote electronic vaulting

Compared to manual media transportation methods, remote electronic vaulting reduces recovery time, data losses, and errors. It requires a dedicated network connection to support large or frequent data transfers. In most environments, it is usually impossible to save the data to a media device in real time. Rather, backups are written periodically during off-shifts or low utilization periods. While electronic vaulting configurations can be run to “hot” site locations for improved recovery time, recovery still depends on the most recent backup copy. Note the following characteristics of remote electronic vaulting:

- There is no need to manually transport data to the recovery site.
- A dedicated network connection is required to support large or frequent data transfers.
- Recovery still depends on the most recent backup copy.

1.1.5.3 Remote data mirroring

Remote data mirroring provides faster recovery and less data loss than remote electronic vaulting. Since data is transferred to disk rather than tape, performance impacts are minimized. Disk replication solutions include file mirroring and disk mirroring:

- File Mirroring, like database replication, is most useful in cases where the data must be read while more than one site is in operation. File mirroring is a possible choice for systems that do not include raw disk access; however, most major databases utilize raw disk for better performance.
- Disk mirroring is a good solution for disaster recovery. You can maintain a total disk and file system image at the backup site that tracks all changes made to data in the production environment. A key feature to look for in a disk mirroring solution is its support for synchronization of data copies. Unlike database replication and file mirroring, the data is only available for active applications at the recovery site when the original site fails. (It is read-only at the recovery site while the original site is sending data.) Synchronization is necessary at two points:

- When you initialize the recovery site and bring it up to date without taking the original site off-line
- When a failed site comes back up and needs to be updated

Another good feature to look for in a disk mirroring solution is the ability to use a variety of disks. An open-systems solution offers more flexibility in this regard. Since scheduled maintenance is always an issue for a high-availability solution, the disk mirroring solution must also have established processes and tools to propagate changes to the recovery site.

1.1.5.4 Distributed File Systems (DFS)

Distribute file systems, such as Network File System (NFS) and Andrew File System (AFS), provide data access over IP networks. Some characteristics of DFS are:

- Distributed file systems can allow read/write access from multiple client sites.
- Data consistency is only achieved over long periods of time.
- Data is cached on the clients to increase performance, but, in the event of a failure in the client, server, or network, data loss is possible.

1.1.5.5 Optically-attached storage

Optically-attached storage, such as Fiber Channel attached disks, SCSI optical extenders, and SSA optical extenders, provide a physical separation of storage from 600 meters to 10 kilometers.

1.1.5.6 Application-specific data replicator

Some applications, such as Oracle, Sybase, DB2, and Domino, provide data replication services as add-on features of their products. These facilities replicate their own data or the transaction logs associated with their data. Since only the application data is mirrored, system files and other data are not replicated. These offerings typically only monitor the loss of application availability, and depend on high-availability products, such as HACMP or user-written scripts, to handle hardware failures and complicated recovery scenarios. Some of these offerings only support mirroring of the transaction logs, and these logs must be “replayed” by a database administrator who can extend recovery periods to more than one day. Others, such as DB2, can replicate just the “commit” records, or, like Domino, they can replicate just the entire “document”, but in asynchronous mode. Therefore, they vary in terms of data loss (from minutes to no data loss) and recovery time (ranging from less than 30 minutes to greater than 12 hours). The majority of replicators are asynchronous and, therefore, have the potential for data loss upon disaster.

1.1.5.7 High-Availability Geographics Cluster (HAGEO)

HAGEO is at the high end of disaster recovery solutions. It adds to remote data mirroring the advantage of a High Availability Cluster solution, such as HACMP, to provide automatic failover between systems located in different geographic locations without distance limitations.

1.2 Products introduction

The IBM RS/6000 platform offers two applications to implement disaster recovery solutions:

- GeoRM provides real-time wide-area mirroring of customer data between systems connected by IP-based networks.
- HAGEO adds to GeoRM functionality the feature of an HACMP solution without any limitation on distance between the two sites.

When we discuss GeoRM and HAGEO and its functions, we use some new terms. Some useful definitions for this purpose are listed in Table 2.

Table 2. HAGEO definitions

Term	Definition
Geographic mirroring	The act of providing a mirror image of data at a computer site that is geographically distant from the originating site.
GeoRM (Geographic Remote Mirror)	Software that allows one to mirror data on disks at sites that can be separated by a significant geographical distance.
Global network failure	Total communication failure on a network between the sites.
HAGEO (High Availability Geographic Cluster)	Software that allows one to mirror data and extend the functionality of a high-availability cluster at sites that can be separated by a significant geographical distance.
Kernel Remote Procedure Calls (KRPC)	The GeoMessage subsystem that the local and remote parts of a GeoMirror device use for communication. This subsystem provides multi-network, highly-available, load-optimizing communication in a Remote Procedure Call (RPC) manner. This is the protocol engine for GeoMessage.

Term	Definition
Site isolation	Loss of communication over GeoMessage service networks. At least one machine at each site is active, but the communication channels between sites are down.
Site failure	The failure of all machines at a site. - Catastrophic loss of resources for an entire site of operations. - Prevents any useful work from being performed at that site. - Recovery of resources would seriously impact business because of time to recover. - Example: World Trade Center bombing.
State map	A record of the current state of all data regions written on a GeoMirror device by the host machine. When a site fails and recovers, the HAGEO software reads the state map on each recovered machine in order to update the mirror on the recovered machine. This process is automatic.

1.2.1 GeoRM overview

GeoRM is a data-mirroring product that provides a point-to-point method of duplicating the contents, in real-time, of any given logical volumes over unlimited geographic distances. Since GeoRM is both database and file-system independent, there is no modification required of applications that utilize GeoRM's mirroring capabilities. In the unlikely event of a failure (for example, CPU, disk, network, or power failure), you can be safe in the knowledge that GeoRM has been designed to mirror any data destined for an RS/6000 (source server) and to be mirrored across your IP network to another RS/6000 (target server).

A total failure of the source server at the local location will not cause the loss of data on the target server at the remote location. GeoRM also has the ability to continue operations while recovering from a server failure. A GeoRM configuration can support a maximum of eight RS/6000 servers. This allows for flexibility in the design of the correct configuration in your environment. For example, a target server can support up to seven source servers. Each of these source and target servers can be as close to each other (in the same room) or as far away (halfway around the world) as required.

The GeoRM software consists of the following components:

- GeoMirror: A pseudo-device driver and a logical device that mirrors data entered at one location to a second location.
- GeoMessage: Provides reliable delivery of data and messages between GeoMirror devices at the source and target locations.

Figure 4 shows a three-machine GeoRM configuration.

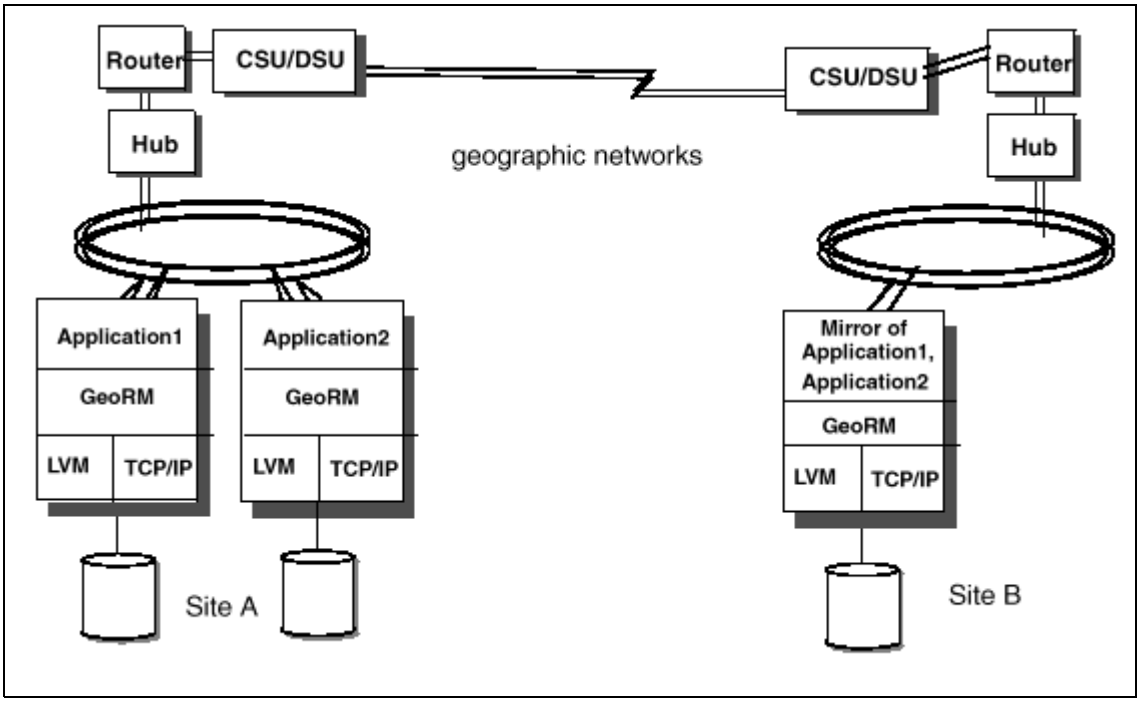

Figure 4. Basic GeoRM architecture

Site A is the active site, where two machines run applications to be mirrored to the one machine at Site B.

The machines at Site A can be geographically separated, even though you configure them as belonging to one GeoRM site. When you configure GeoRM, you establish a one-to-one pairing between GeoMirror devices on machines A and C and a one-to-one pairing between GeoMirror devices on machines B and C. The three machines have access to the geographical networks through the routers. Each machine at each site has unique disks (the disks can be shared if the machines are not geographically separated). Data entered at either machine at Site A is sent over a geographic network and mirrored on a corresponding disk at Site B.

1.2.2 HAGEO overview

The High-availability Geographic Cluster (HAGEO) provides real-time mirroring of customer data between systems connected by local or point-to-point networks, bringing disaster recovery capability to an RS/6000 cluster placed in two widely-separated geographic locations. HAGEO is an extension to HACMP and must be installed with HACMP to provide an extended HACMP cluster.

Key prerequisites are as follows:

- AIX 4.1.4 or later for server with HACMP 4.1.1
- AIX 4.2.1 for servers with HACMP 4.2.1
- AIX 4.3 for servers with HACMP 4.2.2
- Redundant communication paths trough a LAN or point-to-point network

HACMP clusters can be configured in several modes for different types of processing requirements. Concurrent access mode fits environments where all of the processors must work on the same workload and share the same data. In mutual takeover mode, the processors share the workload and back each other up. Idle standby allows one node to back up any of the other nodes in the cluster.

HAGEO provides a data and system resource replication environment that extends HACMP clustering functionality to the wide area by enabling mirroring, failover, cluster heartbeat, and event management over a variety of LAN and point-to-point communication adapters, including token ring, Ethernet, FDDI, T1, and ATM.

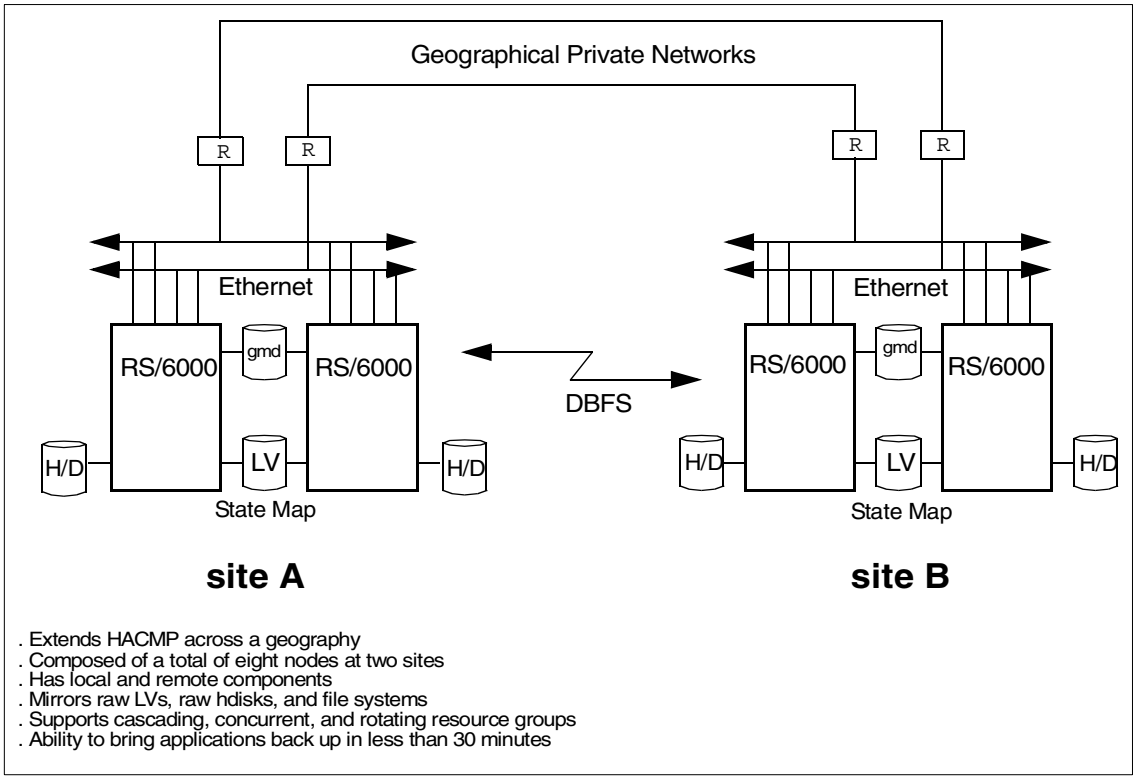

Figure 5. Site failure protection with HAGEO

An HAGEO cluster consists of two geographically-separated sites, each capable of supporting up to four high-availability cluster system nodes. There are three modes of disaster protection and one mode of recovery: Remote hot backup, remote mutual takeover, concurrent access, and remote system recovery.

HAGEO uses the same components that form GeoRM to provide the data mirroring plus the GeoManager that supervises the distributed cluster and adjusts the "heartbeat" on long-distance communication links to detect site failure, control fail-over, and avoid an incorrect conclusion of site failure.

1.3 Position, differences, and HACMP integration

HACMP is a high-availability solution, and it can be used for local high availability failover. It can connect up to eight systems.

As part of a Data Mirroring Strategy, GeoRM can be installed on servers that are already part of an HACMP cluster to provide remote data mirroring to back up the local server data or the cluster data to a geographically-separate server. This allows you to have the critical data saved to the remote location in real time in case of a site disaster. However, there is no integration between GeoRM 2.1 and HACMP at this moment.

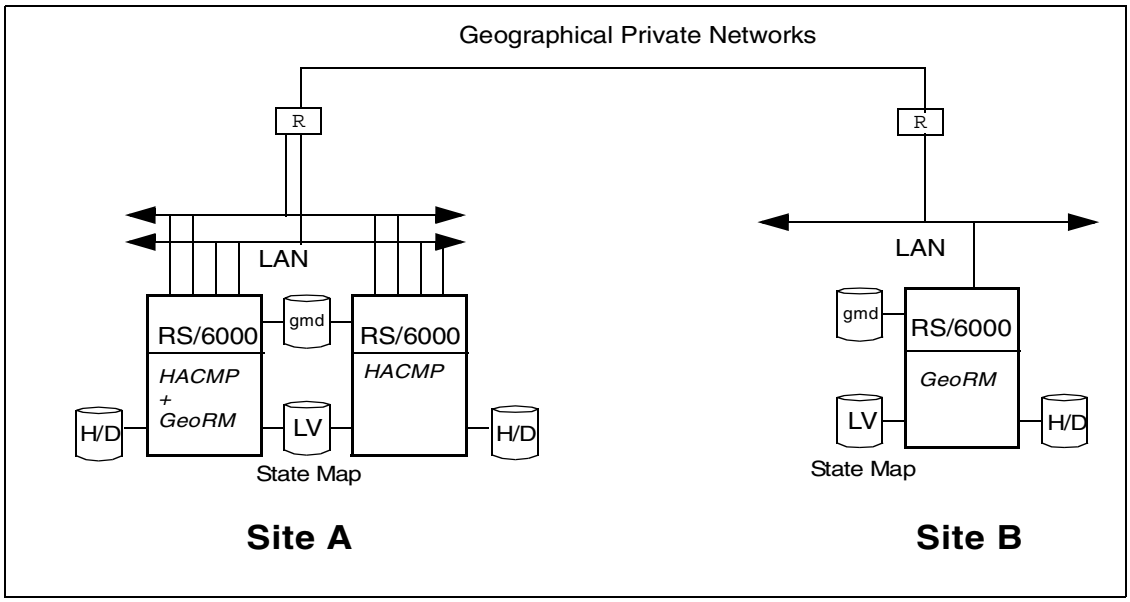

Figure 6. GeoRM can mirror the HACMP shared LV on the other site

HAGEO is the answer to a request by an HACMP system, but without any distance limitations. HAGEO allows not only the mirroring of data between two geographical sites but also the automatic failover of the application to the other site in case of disaster.

1.3.1 GeoRM vs. other disaster recovery solutions

You can use the information in Table 3 to compare GeoRM to other disaster recovery solutions.

Table 3. GeoRM vs. other disaster recovery solutions

Solution	GeoRM/HAGEO advantages/disadvantages
Off-Site tape backup	- Recovery with minimal or no data loss - Shorter time of recovery data - Can operate with both locations active and acting as a backup for each other
Remote Electronic Vaulting	- Provide more timely data recovery and less data loss, because the data is transferred trasparently to the application to the disk rather than a tape.
Distributed File Systems	- Also allows the mirror/backup of raw LV - Read/Write access is allowed from only one site - More consistency of data; it's not caching data on one single site
Optically-attached storage	- No limitation on distance between the two sites - Expensive communication line
Application-specific data replication	- Independent for data mirror. Can indifferently mirror JFS or raw LV. - Minimal or no data loss - Doesn't require a trained DBA at both sites

1.3.2 HAGEO and other disaster recovery solutions

How about current disaster recovery solutions? Table 4 lists some alternative disaster solutions.

Table 4. Current disaster recovery solutions

Solution	Description
Off-site tape backup	Up to several days to recover.
Mobile units	- Computer room fully contained in a trailer. - 24 to 48 hour response time plus time to load the software.

Solution	Description
Hot/Cold sites	The time required to bring up a Hot Backup site is around 24 hours.
Remote copies of data (replicators)	- Only replicates proprietary databases. - No automatic failover. - Requires trained DBA at both sites.
Application replicators	These solutions are database system-dependent.

Database replication servers are available from several vendors. These mirror the contents of databases over long distances. Depending on the product considered, they can be set up to work in either synchronous or asynchronous mode. In some cases, the replicator products have a higher price tag than HAGEO, and, in all cases, the data that can be protected is restricted to the database contents. HAGEO can protect database contents and anything else in the system that is contained on a disk.

1.4 Summary

After reading this chapter, you should be able to answer the following questions:

- What is disaster recovery?
- What is high availability?
- What are the goals with and differences between HAGEO and GeoRM?
- What can GeORM do for me?
- What can I achieve with HAGEO?

Chapter 2. GeoRM and HAGEO

This chapter describes the components of the GeoRM and HAGEO software: GeoMirror, GeoMessage, and Geomanager. We will explain the different modes of geographic disk mirroring and the supported HAGEO and GeoRM configurations.

2.1 Basic components

The GeoRM and HAGEO software both have three basic components:

- **GeoMirror** - A pseudo-device driver and a logical device that mirrors data entered at one location to another location.
- **GeoMessage -** Provides reliable delivery of data and messages between GeoMirror devices at the source and target locations.
- **GeoManager -** Provides configuration and management tools.

Figure 7 describes the links between different components.

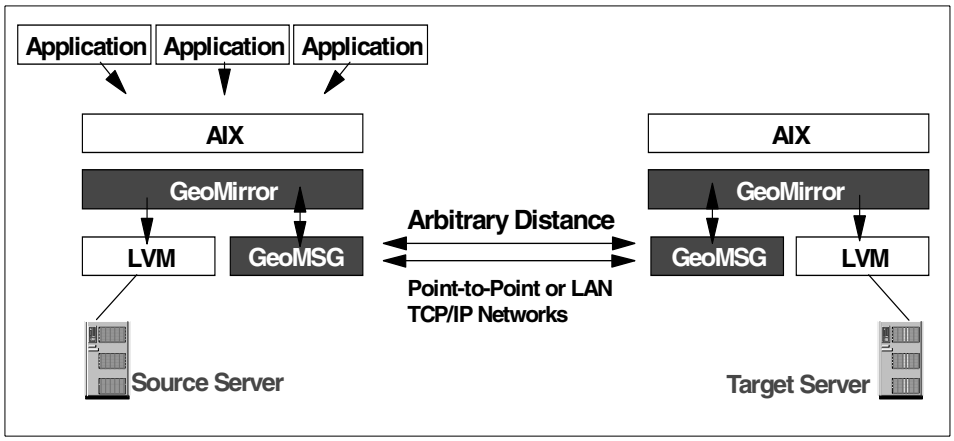

Figure 7. Concept of GeoRM/HAGEO

GeoMirror acts as a layer between the application and the AIX Logical Volume Manager (LVM). It replicates and forwards block-device or character-device write requests to its partner instance at the remote location via the GeoMessage facility.

2.1.1 GeoMirror

Geographic mirroring is the process of writing the same data onto disk devices on machines at separate physical locations connected by a point-to-point network.The geographic mirroring is accomplished by using GeoMirror Devices (GMD).

A GeoMirror device is a logical device that has a local and a remote component. The GMD acts as an interface between the application and the underlying disk or LVM device drivers. The device driver supports exactly the same interfaces as those for disks or logical volumes; so, the application cannot tell the difference. The definition of the GMD contains the following information:

- Its relationship to a logical volume that stores the actual data.
- Its relationship to a state map logical volume that maintains the state of the mirrored copies of the data.
- The peer machines for the mirror, both local (at the same site - only for HAGEO software) and remote (at a different site).
- The mode of geographical mirroring to be used: Synchronous, MWC, or asynchronous. These modes are described in Section 2.3, “Modes for geographic disk mirroring” on page 25.

Like other ordinary AIX devices, GMDs are defined in the following ODM classes:

- **PdDv** - Pre-defined Device
- **PdAt** - Pre-defined Device Attribute
- **CuDv** - Customized Device
- **CuAt** - Customized Device Attribute

The maximum size of a GeoMirror device is 32 GB, and you can define a maximum of 1024 GeoMirror devices in a configuration. A machine can use any type of GeoMirror (synchronous, asynchronous, or MWC) or a mixture of these types. You must define a role for each part (local and remote) of the device. See the sections on GeoRM or HAGEO for more information about this role. GeoMirror devices can be in one of three states: DEFINED, STOPPED, or AVAILABLE. In the DEFINED state, the device exists in the ODM, but it is not configured and not available for use in the system. In the STOPPED state, the device is configured but not available for use by applications. The device is AVAILABLE when it is configured and available for use by applications. When you create a GeoMirror device, it is created in the

STOPPED state. If you want to change any attributes of a GeoMirror device, the device must be in the DEFINED state.

When you define a GMD device, you can change the following two attributes:

- The **High Water Mark** attribute controls how many 1 KB blocks of data are allowed to be in-flight asynchronously at any given moment. This attribute is only used by async devices on the primary side. The default value is 128. In the section about the Asynchronous mode of mirroring, we will see the utilization of this attribute.
- The **Default Sync Concurrency Rate** specifies the maximum total amount of data that is transferred at a time during a sync operation. This is the data being read from the remote site by the local site when the local site is reintegrating. This number only affects the sync rate for when the local node is reintegrating. Speeding up the sync_rate makes data available sooner on the resyncing side, but it degrades performance on the remote site. The default value is 32 KB, and it can be changed to 64 KB.

2.1.2 GeoMessage

GeoMessage is a combination of the KRPC (kernel-based RPC) kernel extension, KRPC-related commands, and ODM definitions. After all the necessary information has been defined to ODM, starting and stopping GeoMessage is equivalent to loading/unloading the KRPC kernel extension into/from kernel. GeoMessage provides reliable messages over multiple UDP/IP networks and automatically reroutes requests if a request fails to reach a destination machine because of a network failure. GeoMessage guarantees message delivery as long as at least one functioning network exists in the GeoRM or HAGEO configuration. If GeoMessage has more than one network to work with, it performs load balancing in the following way: GeoMessage calculates the round trip time (RTT) for each message it sends out on a given interface. This latency measurement measures the time it takes for a packet to get all the way to its destination and back. This round trip time updates an exponentially-smoothed average round trip. This figure is maintained for each interface. When GeoMessage is presented with a message to send, it checks the RTTs of all the interfaces it knows to be up and queues the message to the best one. The algorithm used is such that if all links have the same value, the first link is always chosen.

When several messages are sent in sequence, they are queued to the first link. If the internal AIX queue or the link starts to build up, the RTT will naturally increase. This would happen if, for example, the data rate for arriving messages (arrival rate times average message size) is larger than

the bandwidth of the link. Eventually, the RTT for the first link becomes larger than the rest. At this point, further messages will be queued to the next link. This would also happen if link quality degraded.

The load balancing behavior of GeoMessage tends to equalize RTT on all the links it knows about, rather than link utilization. One would reasonably expect that, if the steady-state data flow to GeoMessage is higher than that of a single link, the load will be roughly balanced across the links. On the other hand, if the data rate is nearly matched by the first link, the first link will tend to be used exclusively.

2.1.3 GeoManager

GeoManager provides the system administrator with tools that facilitate configuration and management of a GeoRM or HAGEO system.

GeoManager functionality includes:

- Configuration and management utilities
- New ODM object classes (GEOnode and GEOsite)
- Errorlog messages to ensure GeoMirror and GeoMessage components are logged

The GeoManager provided with HAGEO software is more complete and contains the tools that allow the HACMP Cluster Manager to detect and react to the HAGEO cluster site-related events (site failure and site isolation) and integrate the GeoMessage components as HACMP cluster topology entities.

New GeoManager functionality with HAGEO software includes:

- Geographic point-to-point network HACMP Network Interface Modules (NIMs)
- Scripts and programs that integrate the handling of GeoMirror and GeoMessage in cluster events, such as node and network joins and failures
- Scripts that integrate starting and stopping the GeoMirror and GeoMessage components into HACMP start and stop scripts

2.2 State map

The GeoMirror state map is a guaranteed mechanism to ensure data integrity across sites. The state map name must be unique per GMD and per node. It is managed by the GeoMirror device driver directly and is usually pinned in

the kernel heap where it occupies 512 KB of virtual memory for each GMD. The contents in virtual memory are transferred to the state map logical volume by the end of the periods defined by the device operation mode (synchronous, MWC, or asynchronous). Each state map requires 640 KB of disk space. While the state map itself takes up 512 KB (for virtual memory), an extra 128 KB is used for the Logical Volume Control Block (LVCB). If the GMD can be taken over locally within a site, the logical volumes of state maps must be created in a shared volume group. The state map LV is divided into 4-bit cells, each of which maps to 32 KB in the logical volume associated with the GMD. The possible values for these cells are shown in Table 5 on page 21. Figure 8 illustrates the state map concept.

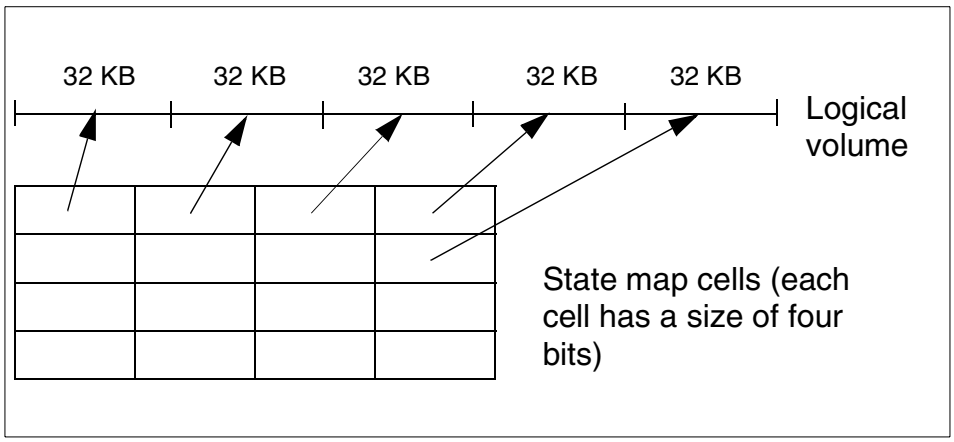

Figure 8. State map concept

The value of the cell represents the status of the 32 KB data logical volume block to which it maps. Only when the status of all cells is consistent (value 0x0) is the data completely synchronized between the sites. The operation to change the value of the cell to 0x0 is called a *clean*. A state map with all the cells consistent between the sites is called a clean state map. When the values of the cells are different than 0x0, the data may be inconsistent between mirror copies. When one or more cells have a value different than 0x0, the state map is said to be dirty.

Table 5. State map values

Values	Status	State Map Operation	Description
0x0	Consistent	Clean	Data is the same on both the local and remote node.

Values	Status	State Map Operation	Description
0x1 - 0x7	Inconsistent or vulnerable		Data may not match between local and remote nodes.
0xf	Stale		Data does not match between local and remote nodes.

All write requests handled by the GMDs change the status of the affected state map cell to inconsistent (value from 0x1 to 0x7). After receiving an acknowledgment that the mirror write was successful in the remote site, the status of the cell is switched back to consistent. If the remote GMD is unavailable, the mirroring to the other site will not be accomplished, and the status of the cell is changed from inconsistent to stale (value 0xf).

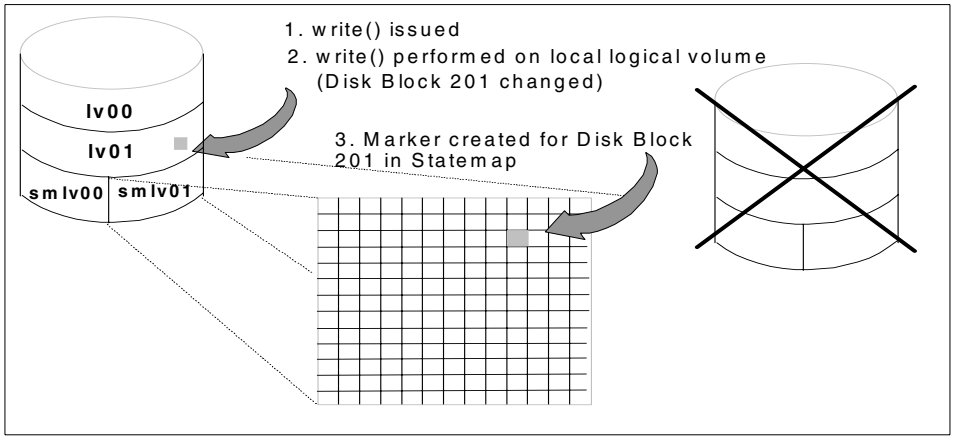

Figure 9. Failure state

Figure 9 shows an example of marking the statemap on the source server when the target server has failed. Disk Block 201 in lv01 is written, and a marker is created for Disk Block 201 in smlv01, which is the state map logical volume for the lv01 logical volume. The next figure shows the target server reintegration. This server consolidates its own state map with the source server state map to see what data has changed, and it requests the specific disk blocks (in this case, disk block 201) to be mirrored across and written locally. The markers are then removed from the statemap on the source server.

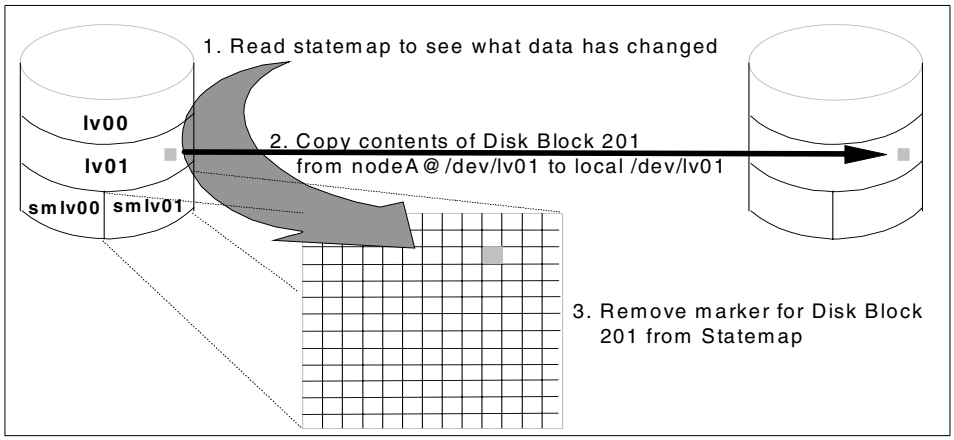

Figure 10. Mirror reintegration

When the GMD is started, the local and remote state maps are examined to verify whether any inconsistencies exist. If the remote node has cells marked as stale or inconsistent, the GMD will not be started.

If both state maps have inconsistent or stale cells, the local GMD is not started. In this case, data divergence has occurred, and you have to choose which site has the current data. Once this is done, a process called *unifying the state maps* synchronizes all of the data based on the version in one of the mirror copies. This process is better described in Chapter 8, “Failure and recovery” on page 247. In the local failover case (where a node fails and there is a local peer node to take over for it), the state maps of both nodes are used to ensure the consistency of data. Figure 11 on page 24 shows a site called Austin where there are two nodes: A and B. The datavg volume group is shared between the nodes and contains the logical volumes for the data0gmd GMD. Node A has the highest priority over the volume group. The LV, data0lv, is the data logical volume; data0sm_11 is the state map logical volume for node A, and data0sm_12 is the state map logical volume for node B. In the state map representation, the white block represents a consistent status, and the gray block represents an inconsistent status.

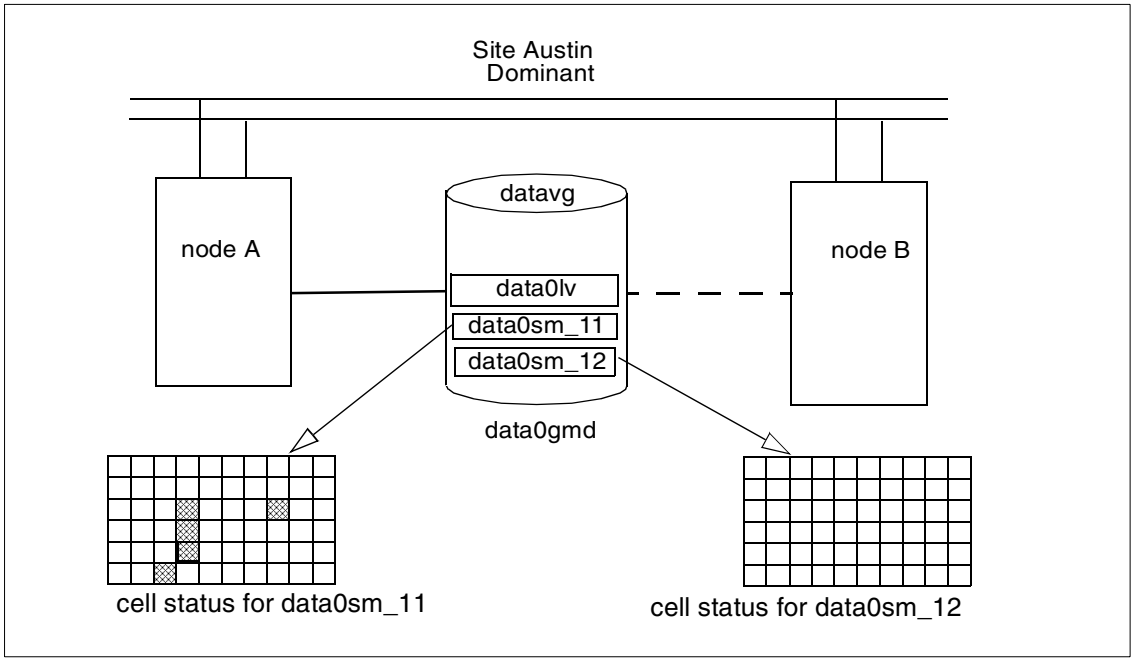

Figure 11. State map status before local failover

After local failover (node A becomes unavailable), node B varies on the volume group and handles the state maps in the following way.

1. It copies the cell values from the state map of node A (data0sm11) to its own state map (data0sm12). This copy is done using bit-wise OR since there is a possibility that the state map from node B might not be clean. This is possible in concurrent access mode.
2. It cleans all the cells in the state map of node A.
3. After performing these steps, node B no longer uses the state map from node A. The final status for the state maps is represented in Figure 12 on page 25.

For a detailed description of how to manage state maps, see Chapter 8, “Failure and recovery” on page 247.

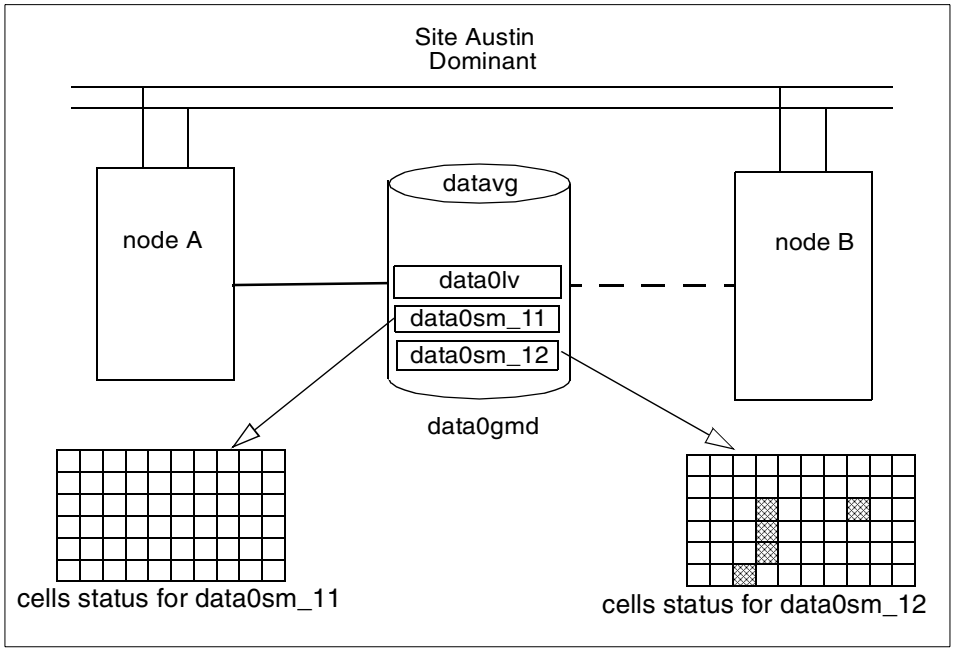

Figure 12. State map status after local failover

2.3 Modes for geographic disk mirroring

GeoRM provides you with the ability to mirror data in three distinct modes: Synchronous, asynchronous, and synchronous with Mirror Write Consistency (MWC). In all cases, reads of data are handled locally. You can then configure your environment to mirror chosen logical volumes in any of the three given mirroring methods.

2.3.1 Synchronous mode

The Synchronous Geographic Mirror Device writes data to the remote site and then writes to the local site before returning control to the application. Once the write returns, you can be sure that data is secure on both sites. The synchronous mode of geographical mirroring is the first type of mirroring to use the synchronous model, where a transaction must be successfully written to both sides of the mirror before control is returned to the transaction. It is analogous to the serial mirroring policy in LVM mirroring. When a GMD is defined to use synchronous (or sync) mirroring, either side of the GMD can receive input. In this discussion, the side of the mirror where the I/O is initiated is referred to as the local site, and the other site is the remote site.

Remember that, since I/O can be initiated from either side of the mirror, this role is valid only on a transaction basis.

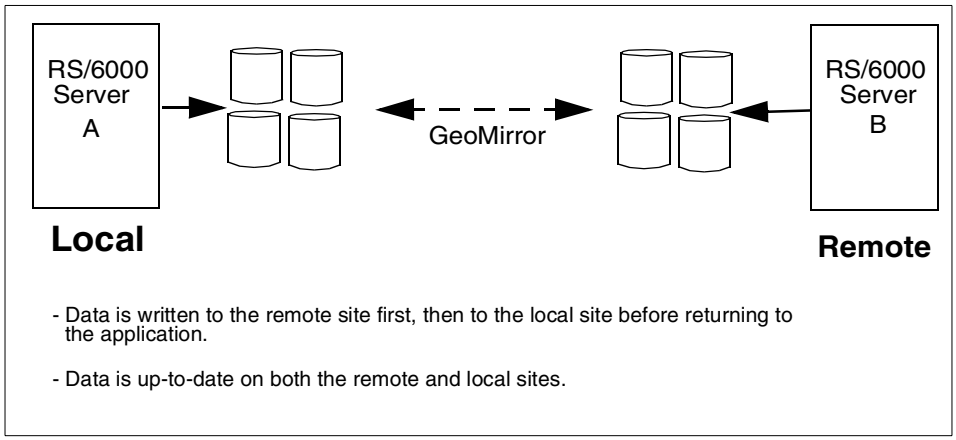

Figure 13. Synchronous mode

During normal operations in synchronous mode, the local side of the GeoMirror device receives write requests and forwards them using the GeoMessage reliable protocol to the remote part of the GeoMirror device that resides on the paired machine at the remote site. At the remote site, the GeoMirror receives the transaction and passes it to the LVM to write to the physical disk. Once the data is written at the remote site, the GeoMirror device there sends an acknowledgment back to the local site, and the data is then written to the local disk. Write requests can originate at either side of the mirror. The local state map is marked consistent as soon as the writes are complete on both sides. The write waits on the local site until GeoMessage receives the acknowledgment of the remote write, or until it receives the message that the remote site is not available. In the latter case, local writes continue, and the state map is marked to keep the record of all writes since communication with the mirrored site was lost. No update is made to the state map if the write fails to complete on the local site. There is no protection from data being written to either side or to both sides at the same time. You do not want this to happen unless you are running concurrent access, with its protection of the cluster lock manager. Normally, the application in a non-concurrent mode cluster will be set up to do reads and writes on only one site. HACMP will bring up the application on only one site at a time, under control of the application server definition. The application is brought up on the remote site only if there is a site failure.

2.3.2 Mirror Write Consistency (MWC) mode

The Synchronous with Mirror Write Consistency writes are issued to both the remote site and the local disk at the same time. State map devices are used to ensure data integrity, and control is returned to the application only when writes have been completed on both the local and remote sites. MWC mode is also based on the synchronous model, in which a transaction needs to have both sides of the mirror successfully written to before control is returned to the application. This time, however, the writes to both the local site and the remote site are dispatched at the same time. This is analogous to the parallel mirroring policy of the LVM. If a GMD is defined to do its mirroring in MWC mode, the state map for the GMD keeps track of the state of each of the mirror writes during the transaction. If there is a failure midway through, the state map is used to recover the consistency of the data. Figure 14 illustrates MWC mode.

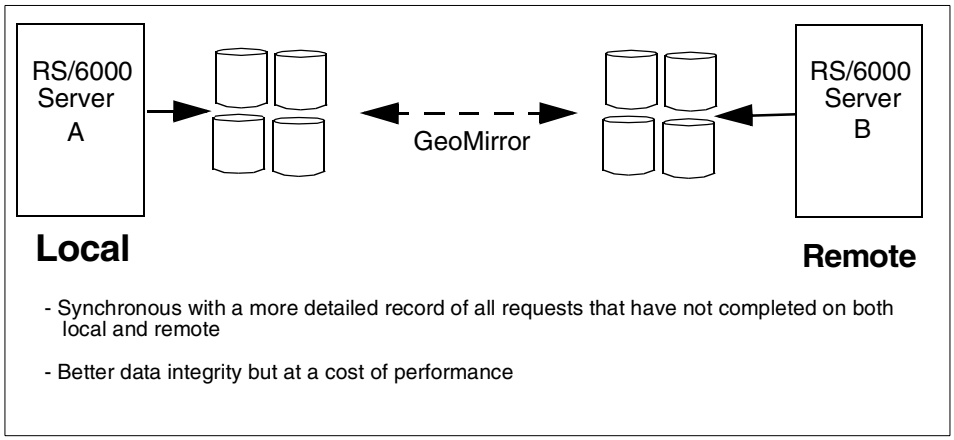

Figure 14. MWC mode

In Mirror Write Consistency mode, the device operates as it does in synchronous mode, but it also keeps a physical record in the state map of all data regions that have not completed both local and remote writes. This increases the security of the data, but may slow down the throughput. The throughput effect of MWC, compared to synchronous, is a trade-off between the benefit gained by dispatching both writes at the same time versus the cost of having to update the state map at every stage of the mirror writes.

2.3.3 Asynchronous mode

The Asynchronous Mirror Device offers the highest performance of the mirroring options. Data is written to the local disk before returning control to

your application. Writes are then queued to the remote site through the network. To ensure protection of data, writes are performed in temporal order, and the mirrors will revert to synchronous mode should your data become out of step between the two sites by a predefined amount. In a synchronous mode, when a write request is received from the application, the local copy is written first, and the remote request is dispatched over the network. Control is returned to the application when the local write has been completed. HAGEO/GeoRM then completes the remote write in the background tracking its progress in the state map. The writes to the remote disk cause lag behind the writes to the local disk. You can control the number of kilobytes by which the local and remote disks are allowed to differ. You can specify this number when you configure the GeoMirror device by setting a parameter called the High Water Mark. By default, this parameter is set at 128 KB. If the number of outstanding updates exceeds the High Water Mark, the GMD reverts to MWC operations until it is completely caught up, after which it switches back into asynchronous mode. Asynchronous mode is depicted in Figure 15.

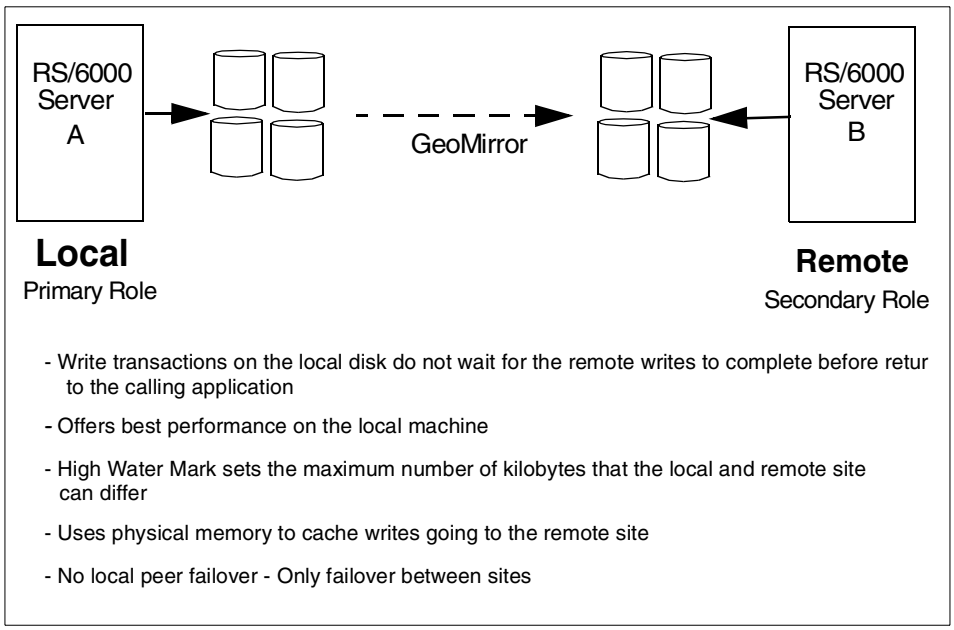

Figure 15. Asynchronous mode

The local device also keeps a detailed record in the state map of the state of all requests that have not completed both their local and remote writes. In asynchronous mode, you can only have one machine at each site for each GMD. This means that, with asynchronous GMDs, you do not have the ability

to have local failover, where the GMD can be taken over by a second machine physically connected to the disks as in a traditional HACMP cluster.

Also, unlike GMDs using either synchronous or MWC mirroring, with an asynchronous GMD, you can only perform I/O from one side of the mirror. This applies to both reads and writes. You must define one side of each GeoMirror device as the primary, where I/O requests originate, and the other side as secondary, where requests are received and mirrored. Any read or write requests to the secondary side of the GMD are rejected. If a site failure occurs, such that the remote site must take over the application, its role must be switched from secondary to primary. HACMP does this automatically as part of its recovery procedure. This operation must be done manually with GeoRM software.

2.4 GeoRM

A GeoRM configuration can have from two to eight machines, and the machines must be distributed between two GeoRM sites. For example, a target server can support up to seven source servers. Each of these source and target servers can be as near (in the same room) or as far (halfway around the world) as is required.

GeoRM does not provide automated failure and recovery. Manual procedures to achieve this automated failure and recovery in GeoRM require you to have a technical understanding of how GeoRM operates.

For complete security, geo-mirrored data at the backup site should be mirrored locally with the LVM command, and the mirrored data can also be periodically backed up and stored on other media.

GeoMirror Device
With GeoRM, you must define a role for each part (local and remote) of the device. The local side of each GeoMirror device has the primary role, where write requests originate, and the remote side has the secondary role, where requests are received and mirrored. If necessary, you can reconfigure the device to change roles.

2.4.1 GeoRM network

Sites may be connected via a variety of local or wide-area TCP/IP networks. Each site should be connected to at least two geographic data networks between sites.The GeoRM networks should be dedicated to exclusive use by GeoMessage. No clients should have access to these networks.

It is important to use diverse routing, if possible, to avoid losing more than one path at once. Using more than one router avoids having the router be a single point of failure.

2.4.2 GeoRM configurations

The basic GeoRM configurations are as follows:

- **GeoRM active-backup** - All applications run at the active site and are mirrored to the backup site.

  One variation of this configuration includes having one or more applications run on machines in dispersed geographic locations; these are all configured as belonging to one active site. These applications are mirrored to one or more machines at the backup site. Figure 16 shows a three-machine GeoRM configuration.

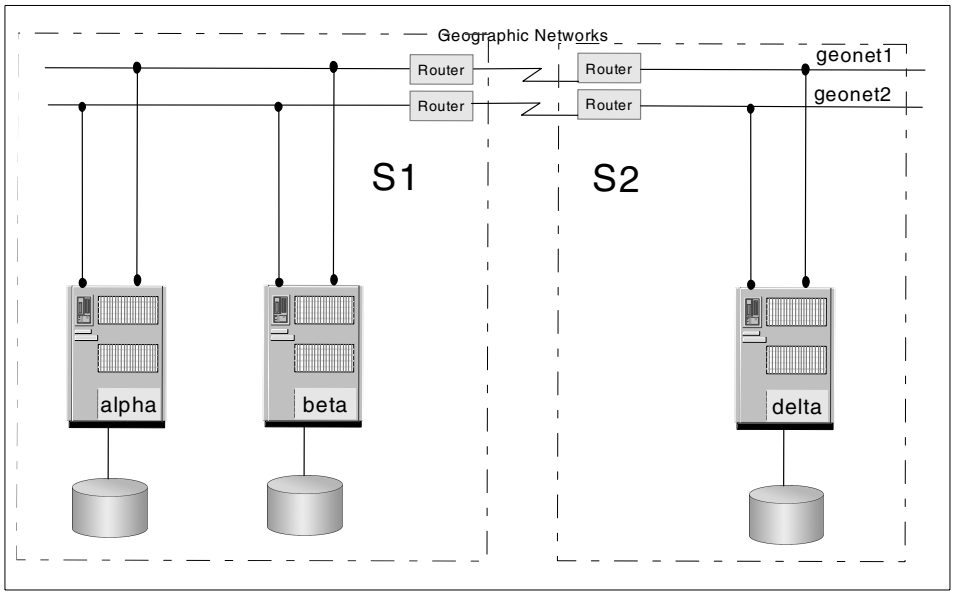

Figure 16. Advanced GeoRM configuration

The configuration has two different locations with two machines at site S1 and one machine at site S2. The two machines at site S1 can be geographically separated, even though they are considered to be one site by the GeoRM software. The three machines have access to the geographical networks through the routers. Each machine has its own disks. When you configure GeoRM, you establish a one-to-one pairing between GeoMirror devices on machines alpha and delta and a

one-to-one pairing between GeoMirror devices on machines beta and delta. Data entered at either machine at site alpha is sent over a geographical network and mirrored on a corresponding disk at site delta.

A second variation of this configuration includes having an application running on two or more machines in the same location, sharing a disk to facilitate handling local machine failure or maintenance while keeping the application and geo-mirroring running. Figure 16 on page 30 illustrates this configuration. The disks at site S1 are shared between the machines alpha and beta.

- **GeoRM mutual backup** - Applications run at each site and are mirrored at the other site.

All configurations provide a rapid remote backup for critical data. The data at the backup site is guaranteed to be up-to-date with the data entered at the active site until the failure of an active site or one of its components interrupts the geo-mirroring process. Basically, machines at the backup sites serve as storage devices.

2.5 HAGEO

HAGEO is an extension of the HACMP software. HACMP ensures that the computing environment within a site remains highly-available. The HAGEO software ensures that the critical data remains highly-available even if an entire site fails or is destroyed by a disaster. The first thing to notice is that an HAGEO cluster is just like an HACMP cluster. An HAGEO cluster can consist of two to eight nodes, just as with HACMP.

However, in HAGEO, these nodes are now spread between two sites. The sites are connected by one or more geographic networks, even though you may be inclined to think of this as two HACMP clusters connected by geographic networks.

When you configure the HAGEO cluster, you define a new component: The HAGEO site. You define the nodes as belonging to one of the sites, and you include the user ID of the person to be notified in case of a site-related event. You also define one site as the dominant site and include information about the backup communications link between sites you have set up. The dominant site is the site you choose to leave running if site isolation occurs.

In each site, the two nodes have a set of shared disks physically cabled together in the same way as an HACMP cluster. In most cases, these shared disks will contain the data that will be geographically mirrored from one site to another.

GeoMirror device
If you use GeoMirror devices in synchronous or Mirror Write Consistency (MWC) mode, either side of the device can originate or receive requests (not at the same time). You do not define a particular role (primary or secondary) when you configure the device; instead, you use the default role that has *none*. This facilitates using either site as the one where requests originate, in case of a site failure. You can also define a chain of local peers to take over a failed synchronous GeoMirror device at the same site.

But, if you use GeoMirror devices in asynchronous mode, you must define roles for each part (local and remote) of the device. One side of each GeoMirror device has the primary role, where write requests originate, and the other side has the secondary role, where requests are received and mirrored. You cannot define a local peer to take over a failed asynchronous GeoMirror device at the same site.

2.5.1 HAGEO networks

Each site should be connected to at least two point-to-point geographic networks for communicating both date and HACMP heartbeat traffic between sites. The HAGEO networks must be dedicated to exclusive use by GeoMessage; no clients should have access to these networks, and the IP Address Takeover (IPAT) is not supported.

HAGEO software adds Geo_Primary and Geo_Secondary Network Information Modules (NIMs) to HACMP. You define the HAGEO networks to HACMP using these NIMs. The Cluster Manager can then monitor the state of these networks.

If possible, it is important to use diverse routing to avoid losing more than one path at once. Using more than one router avoids the problem of having the router be a single point of failure. The networks must use multicasting if they are going through a router so that the IP address of the original sender is not changed during the transmission process. This is a restriction imposed by the HACMP software.

A secondary network set up for HACMP heartbeat communication only is required. You can either use Dial Back Fail Safe (DBFS) or a secondary geographic network, but not both. This backup path gives the Cluster Manager a way of distinguishing between site failure and site isolation.

2.5.2 DBFS

You can use Dial Back Fail Safe (DBFS) as a secondary geographic network. DBFS uses a modem with Asynchronous Terminal Emulation (ATE), which

enables a user to run commands on the remote system and send and receive files. The user must be a member of the UUCP group in order to use ATE.

You can use the subcommands of the `ate` command to test, check, or alter the current settings for the connection.

If DBFS is configured and all the geographic networks fail, the non-dominant site calls up the dominant site. It will try to call each node on the dominant site three times until it finds at least one node up. If it gets no answer from any of the nodes on the primary site, it will maintain its site takeover of resources. If it finds at least one node up on the dominant site, it will shut down all nodes on the non-dominant site.

For security reasons, we recommend that you use a non-root user for DBFS. For more information about the configuration, see Section 5.2, “Dial Back Fail Safe (DBFS)” on page 123.

2.5.3 Overview of HACMP

HACMP uses loosely-coupled clustering technology to prevent individual components, including processors and network adapters, from being single points of failure within clusters. It ensures that the computing environment within a site remains highly-available.

When configuring HACMP, you identify the set of cluster resources essential to processing and defining takeover relationships among the cluster nodes that access the resources.

For complete information on HACMP, see the *HACMP V4.3 AIX: Installation and Administration Guide*, SC23-4283.

Resource group
HACMP provides a highly-available environment by identifying a set of cluster-wide resources essential to uninterrupted processing, and it then defines relationships among nodes that ensure these resources are available to client processes.

HACMP considers the following to be resource types:

- Volume Groups
- Disks
- File Systems
- File Systems to be NFS mounted
- File Systems to be NFS mounted

- Service IP Addresses
- Applications

Each resource in a cluster is defined as part of a resource group. This allows you to combine related resources that need to be together to provide a particular service. A resource group also includes the list of nodes that can acquire those resources and serve them to clients.

A resource group is defined as one of three types:

- Rotating
- Cascading
- Concurrent

Each of these types describes a different set of relationships between nodes in the cluster and a different set of behaviors upon nodes entering and leaving the cluster.

Rotating resource groups
A rotating resource group is associated with a group of nodes rather than a particular node. A node can be in possession of a maximum of one rotating resource group per network.

As participating nodes join the cluster for the first time, they acquire the first available rotating resource group per network until all the groups are acquired. The remaining nodes maintain a standby role.

When a node holding a rotating resource group leaves the cluster either because of a failure or gracefully while specifying the takeover option, the node with the highest priority and available connectivity takes over. Upon reintegration, a node remains as a standby and does not take back any of the resources that it initially served.

Cascading resource groups
All nodes in a cascading resource group are assigned priorities for that resource group. These nodes are said to be part of the group's resource chain. In a cascading resource group, the set of resources cascades up or down to the highest priority node active in the cluster. When a node that is serving the resources fails, the surviving node with the highest priority takes over the resources.

By default, the first node in a group's resource chain to join the cluster acquires all the resources in the resource group only if it is the node with the highest priority for that group.

Member nodes of a cascading resource chain always release a resource group to a reintegrating node with a higher priority.

Concurrent resource groups
A concurrent resource group may be shared simultaneously by multiple nodes. The resources that can be part of a concurrent resource group are limited to volume groups with raw logical volumes, raw disks, and application servers.

When a node fails, there is no takeover involved for concurrent resources. Upon reintegration, a node again accesses the resources simultaneously with the other nodes.

2.5.4 HAGEO configurations

You can configure an HAGEO cluster in any of the configurations supported by HACMP. These include hot standby, one-sided takeover and mutual takeover configurations. You can also use concurrent configurations with HAGEO.

HAGEO users will choose their configurations based on the following criteria:

- Availability requirements
- Performance requirements
- Budget considerations

In a non-concurrent environment, the intent is for each protected application to only read and write from one site at a time. This is accomplished by using an HACMP application server definition to start and stop the application. However, you must know that there is no built-in protection in HAGEO to prevent someone from writing to the remote site's GMD or underlying logical volume when you are using synchronous or MWC modes of geographic mirroring. With asynchronous GMD mirroring, each side of the mirror is always either in the primary or the secondary role. I/O is only allowed on the primary site. The asynchronous GMDs on the secondary site give you either a read or write error if you attempt I/O.

The concurrent environment can be set up for either concurrent intra-site access or concurrent inter-site access. Intra-site access means that the application is being run on multiple nodes at a single site at any one time. Meanwhile, the data is being mirrored to the other site, which is ready to take over in case of failure. Inter-site access means that the application is running on multiple nodes at both sites at any one time. In all cases of a concurrent

access configuration, use of the HACMP Cluster Lock Manager or the ORACLE Distributed Lock Manager is required for data integrity.

2.5.4.1 Hot standby

A remote geographic site and system are designated as the hot backup site and system. The backup system includes hardware, system and application software, and application data and files. In the event of a failure, the failed system's application workload automatically transfers to the remote hot backup system.

If the application to be mirrored is running on only one site and high performance on the local site is of the utmost importance, you might choose a GeoMirror asynchronous mode with the second site functioning as a standby. One site is clearly primary, and the other is clearly secondary, even though the second site can also be running other local non-geographically-mirrored applications.

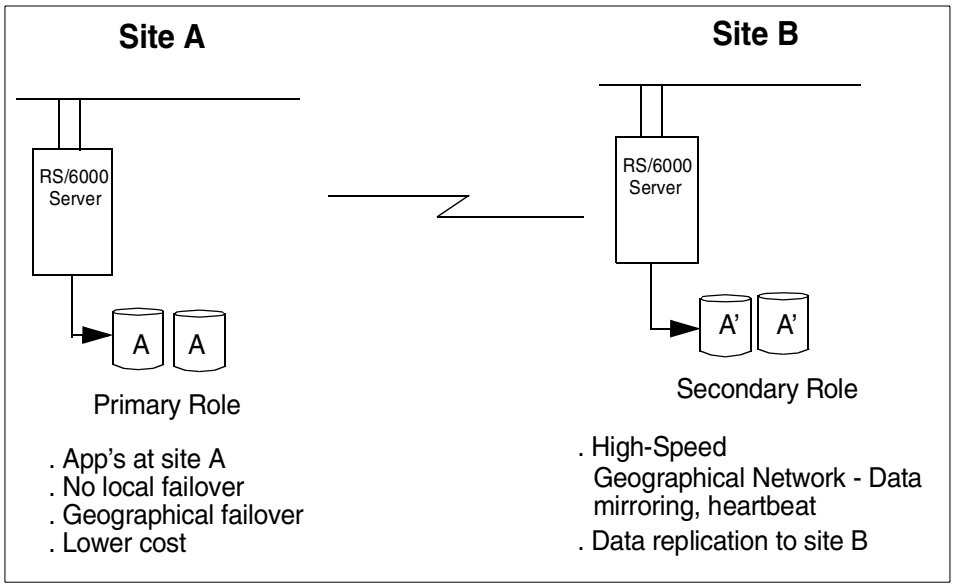

Figure 17. HAGEO hot standby configuration

This figure above shows a two-node HAGEO cluster. It has one node per site, with HAGEO clients accessing data at the primary site. All data for the Geo application entered at site Austin is asynchronously mirrored to site Dallas. Each site has a resource group for its GeoMirror and state map volume group.The application is in a separate resource group defined across both sites. This resource group is configured as a cascading group, with node A

having the higher priority for ownership. If node A fails, node B takes over the resource group.

The GeoMirrors on node A at the primary site are configured as primary. The GeoMirrors on node B at the backup site are configured as secondary. If node A fails (a site failure in this case), the GeoMirrors on node B will automatically be switched to primary to allow for client access.

Note that client access is through a separate LAN connection and not through the geographic network.

When you bring node A back up, it will rejoin the GeoMirrors defined as primary. Clients must be reconnected to node A.

2.5.4.2 Mutual takeover

Geographically-separated sites and systems can be designated as hot backups for one another. Mutual takeover allows each to operate as an independent system or as part of a distributed system. Should either experience a failure, the other acts as a hot backup and automatically takes over the designated application workload of the failed system.

The GeoMirrors are configured to use the synchronous or MWC GeoMirror mode, thus, enabling local takeover of resources on a failed node within a site. Figure 18 illustrates an HAGEO mutual takeover configuration.

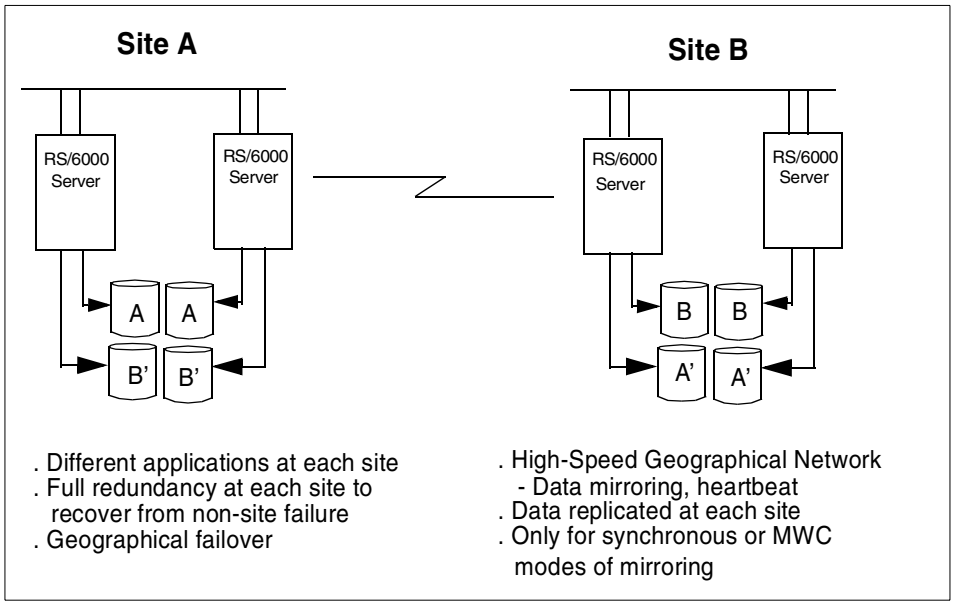

Figure 18. HAGEO mutual takeover configuration

Figure 18 on page 38 shows a four-node HAGEO cluster. It has two nodes per site with HAGEO clients accessing Geo applications at both sites. All data for applications entered at site Austin are synchronously mirrored to site Dallas. All data for applications entered at site Dallas is mirrored to site Austin synchronously.

The Geo1 application resource group is configured as a cascading group, with node A having the higher priority for ownership. If node A fails, node B takes over the resource group and continues offering those service. If the site Austin suffers a disaster, the Geo1 application is first taken over by node C and then by node D in site Dallas.

The Geo2 application resource group is configured as a cascading group, with node D having the higher priority for ownership. If node D fails, node C takes over the resource group and continues offering those services. If the site Dallas suffers a disaster, the Geo1 application is first taken over by node B and then by node A in site Austin.

Each site has resource groups for its GeoMirror and state map volume groups.

2.5.4.3 Concurrent access

In a concurrent access configuration, all nodes at one site have simultaneous access to the concurrent volume group and own the same disk resources. The other site is set up the same way. If a node leaves the site, availability is not affected since other nodes have the concurrent volume group varied on. If a site failed, the other site offers concurrent access on nodes at that site.

Concurrent access in an HAGEO cluster is configured almost the same as in HACMP. Instead of accessing raw LVs, the application accesses GeoMirror devices. With concurrent access, you have to plan your database so that you do not get continual lock requests and lock grants across the geography. This is a particular concern if you are planning to run in an inter-site concurrent access mode.

Chapter 3. Planning and utilities

Before proceeding further, we assume that you are familiar with the basic concepts and facilities of HAGEO and GeoRM. You should also have some knowledge of HACMP; otherwise, we suggest that you first read the *HAGEO Concepts & Facilities Guide V2.1*, SC23-1922 or the *GeoRM Concepts and Facilities, V1.1*, SC23-4307. Both manuals come with the software product. It will also be better if you have read the related chapters in the *GeoRM Planning and Administration Guide, V1.1*, SC23-4308.

We should make one thing clear: A good plan of the whole system is the most important part of the final result that we obtain. There are little things we can do to tune an HAGEO cluster or a GeoRM configuration after the software is installed and configured; so, we need to think about many of the aspects involved, such as sites, machines, disk subsystems, networks, and so on. Our goals are the best High Availability and the best performance of the system, but, in the real situation, we have to compromise between cost and performance. Through good planning, we can, as much as possible, eliminate the Single Points of Failure and have the best possible performance.

This chapter will cover some useful tips for planning an HAGEO cluster or a GeoRM configuration including rules of thumb and some utilities we can use to help us get a better estimate. These utilities are gmdsizing, krpcstat, gmdstat, and filemon. In fact, there are many aspects we need to think about (many of them case-by-case). Here, we highlight some general considerations that are helpful for you to think about in more detail when making decisions. We do not intend to be very detailed, like a planning guide. Rather, we will specify the key points. For more detailed information, we can refer to the related planning and administration Guide. It is also better to use the worksheets appended in the Planning and Administration Guide to keep all records.

3.1 Site planning

This section will discuss the site planing of HAGEO and GeoRM separately.

3.1.1 Planning HAGEO sites

In HAGEO, two sites are in one HAGEO cluster. We need to think about the cluster as a whole and the relationship between sites. Following are some items you need to consider:

- According to the real environment, which site will be the dominant site? The dominant site is the site that, in case of site isolation, takes over and

continues operations in order to avoid data divergence. Another site will be the non-dominant site; so, you can choose one site as the dominant site. Perhaps, it is more reliable than another site or has more applications running on this site.

- What application will be run at each site? According to this, you can decide the appropriate network type and bandwidth and the size of disk subsystems. We suggest that only the critical data be geographically mirrored.
- How many machines in each site will participate in the HAGEO cluster? Not all the machines in one site have to be included in the HAGEO cluster; you can still define separate HACMP clusters at each site if only local takeover is required. In this case, ensure that any local applications, networks, or adapters that are not to be included in the HAGEO cluster are handled properly.
- What type of communication system do you plan to use between sites? This also means the WAN type used for site communication. This is one of the key factors that affect performance. It is strongly recommended that you use more than one Geo_Primary network between sites. Two networks are recommended to maintain High Availability. You also need to choose the type of backup communication for keeping message verification alive if site failure occurs. A DBFS or Geo Secondary network are the alternatives. If you choose none, you cannot identify site isolation or site failure, which results in the data divergence.
- Which GeoMirror mode will you use? You need to trade off between data integrity and performance, and if you are using asynchronous mirror mode, each GeoMirror device is limited to one remote peer. Only one side of the device can do I/O while the other side can only be the mirror, and there is a data lag behind this mode. It also means that when site failures occur, there is data loss. Consider whether you can afford the data loss. Normally, MWC mode is recommended. In most cases, this mode performs better than synchronous while keeping the data consistent. An MWC supports more than one machine in one site. It means you can have local takeover function to avoid too many site takeovers when only one machine goes down in the site.

3.1.2 Planning GeoRM sites

Most considerations are quite similar to those for HAGEO when you plan GeoRM sites. But, in GeoRM, if a site contains more than one machine, the machines in the site do not have to share the same geographic location or disks. You also need to consider the following:

- What application do you want to mirror data to a remote site? What applications are critical to your business? Ensure that any local applications, networks, or adapters that are not to be included in the GeoRM configuration are properly handled.
- Where do you want your application to reside? This will decide whether the GeoRM type is Active Backup or Mutual Backup. If you run applications on one site, another site should just be the mirror, that is, the active backup. If you run applications on both sites, that should be mutual backup. If you have a mutual backup environment, you should be very careful when you use the `stopgmd -A` or `startgmd -A` commands because both sites have primary and secondary GMDs.
- If a site has multiple machines, will the machines share disk resources? If the machines contain different GMDs, they do not need to share disks. If the machines at one site are remote peers for the same GMD, they need to share disks for failover access.
- What type of GeoMirror mode will you use? You need to compromise between data integrity and performance, and, if you are using asynchronous mirror mode, each GeoMirror device is limited to one remote peer, and only one side of the device can do I/O while the other side can only be the mirror. Normally, MWC mode is recommended. You can refer to the preceding information in the HAGEO site consideration.

3.2 Planning machines

The following are some issues that need to be considered when planning machines:

- What size processors and memories are needed for your machines? What we would like to do here is avoid having CPU and memory be the performance bottleneck in the HAGEO cluster or GeoRM configuration. Then, when we are talking about the performance issues, we can focus on the disks, networks, and applications. Another thing we need to take into account is the workload after a failover occurs. The failover machine will take on extra workload in addition to what it owned originally.
- In HAGEO, if you choose DBFS for backup communication, you need to assign a phone number for the modem connected to every machine, and the DBFS will be the important facility to your HAGEO cluster. You need to configure it well and test it often. Make sure DBFS works correctly.
- In GeoRM, there is no DBFS or secondary network to carry keep alive messages, but there is a parameter provided when you define a machine into a GeoRM configuration. Each machine has a Promotes Failure

Timeout attribute, which is used when a machine experiences communication loss because of machine or network failure. If the local machine cannot communicate with a remote peer after trying the amount of time in seconds specified by this value, the local machine determines that the remote peer is down. This is referred to as promoting a failure to a gmddown condition.The value of zero disables the timeout and will only be set when there is an external monitoring system is in place to detect and take specific action when a machine experiences a communication failure. Otherwise, it should always be non-zero. A tip here is to define this value according to the number of the geographic network interface you have. GeoRM will try each interface 20 seconds before it uses the value; so, set this value to no less than the number of geographic network interfaces times 20 seconds.

3.3 Planning disks and GeoMirror devices

This section will highlight some general concerns when we plan the disk subsystem and GeoMirror devices. Some of the following issues are related to High Availability, and some issues are related to high performance. You can select some or all of them according to your situation.

- Disk capacity must be sufficient at each site to handle at least one mirror of all data from the other site except for this site, but multiple mirrors of the same data are strongly recommended because redundancy can help us eliminate SPOF. Especially when there is only one machine in each site configuration, every physical volume failure will cause site failure if there is no other good copy in another physical volume. This can be done by choosing disk hardware that supports data redundancy or through the AIX LVM feature.
- We recommend that you mirror the rootvg including paging space and the JFS log. In this configuration, you cannot use hd6 as the default dump device. You need to define another dump device that can be mirrored; so, in this situation, even if one copy of the rootvg is damaged, this machine can still stay on line.
- If you have more than one application using GeoMirror devices, you may place the GeoMirror device for different applications in separate volume groups to avoid all applications being down if a disk failure causes a volume group failure.
- A similar suggestion is to not co-locate a GMD’s underlying logical volume and the related state map logical volume in one physical volume; otherwise, state map access will inevitably be the performance bottleneck.

- If you want to perform local takeover when a site contains two or more machines, the GeoMirror and statemap logical volumes should reside in a shared volume group.
- Each GeoMirror device is comprised of a local logical volume, a remote logical volume, and a state map logical volume. The underlying logical volumes on both sites must have the same size.
- GeoMirror's local and remote device minor numbers must match, and you must use `r` for raw device in the logical volume name. Use the proper naming convention for easy management.
- It will be a great improvement if it is possible to put the state map to the write-cache disks because write-cache disks greatly decrease the latency when writing data to the disk. Some rules of thumb follow: For asynchronous systems, which are limited to two nodes in total, the cache should be configured on the active system. For synchronous systems, use a cache on both sides of the geomirror. For MWC systems, use a cache locally at first and then remotely. A local cache has a greater effect than a remote one.
- In GeoRM, the device role is assigned to every GMD regardless of the GeoMirror mode, but, in HAGEO, if the mode is sync or MWC, the device role is set to none. If the mode is async, you need to specify primary and secondary for the GMD on both sites.
- If you have a choice, use raw devices rather than file systems for the application.

3.4 Planning networks

When planning a network, you need to think about the hardware you use in the network and the bandwidth the applications will require. We will discuss these considerations in the following sections.

3.4.1 Network components in a geographic mirroring configuration

Firstly, let us look at a possible network topology for the geographic mirror as shown in Figure 19 on page 46.

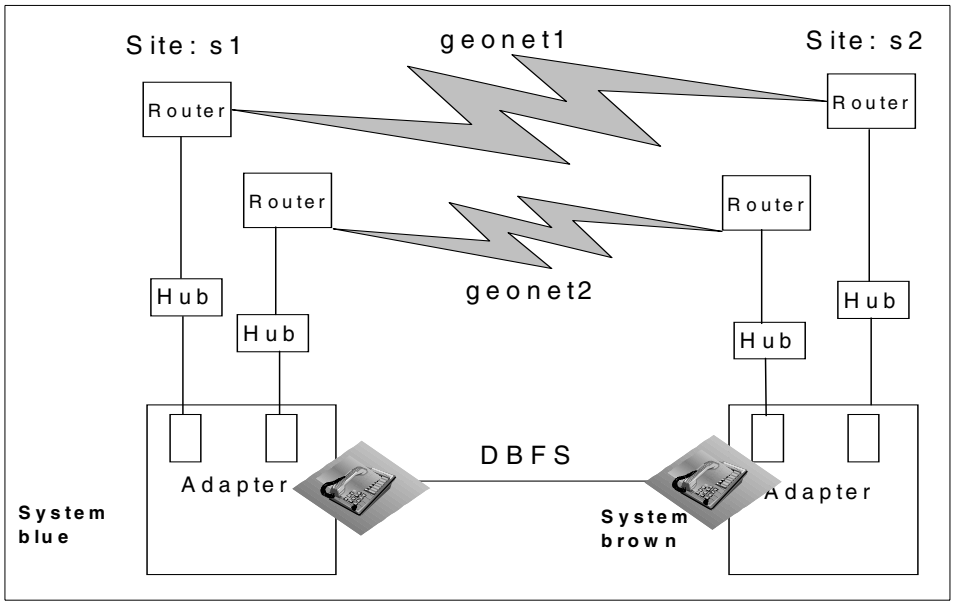

Figure 19. An example of HAGEO/ GeoRM network topology

In this example (as in all cases), both LAN and WAN are involved in an HAGEO cluster or a GeoRM configuration. The hubs here are not necessary and are only considered when there are not enough ports in the router. It is better to use two separate hubs to connect. Using two routers on one site is for the SPOF consideration. Both geonet1 and geonet2 are Geo_Primary networks. Later, we will talk about how to estimate the bandwidth of geonet1 and geonet2. We use the DBFS as the backup communication; so, we need modems and telephone lines.

The communication components in an HAGEO cluster or a GeoRM configuration are a LAN, network devices, such as routers, and a WAN. LAN adapters, such as Ethernet, Token Ring, or FDDI, which participate in the geographic mirroring, typically connect to some sort of router or equivalent network device. These network devices are then connected via some wide-area data communications network.

As shown in Figure 19, we do not have any local network adapters, but it is obvious that, in the case of HAGEO, we would need another two adapters for the HACMP configuration. In the case of GeoRM, we might need additional adapters for other purpose. This is determined by the real requirements.

HAGEO and GeoRM require a TCP/IP-based network. They also require that all network products support UDP. UDP support in HAGEO or GeoRM will be discussed in Section 3.4.5, “UDP consideration” on page 51.

3.4.2 LAN considerations

LAN adapters are used in an HAGEO cluster or a GeoRM configuration to connect the machines to a router or equivalent network devices, such as switches. We can use all kinds of TCP/IP- and UDP/IP-based adapters, which are supported by the AIX operating system. Ethernet, Token Ring, and FDDI are very common. ATM adapters can also be used for LAN communication.

As mentioned earlier, we strongly recommend that you use two Geo Primary networks for high-availability considerations and performance issues. If you do not, once the Geo network goes down, a site isolation occurs, but if you have a backup Geo Primary network, the system can continue to operate. This means we need two LAN adapters in each machine that participates in the geographic mirror configuration. Now, AIX supports a 100 Mbps Ethernet adapter and a 1Gbps Ethernet adapter. For LAN, these adapters are sufficient.

We can change the adapters default configuration to achieve better performance, such as the size of the send and receive queue, and we need to configure the adapters matched with the network device configuration. We can tune AIX network options via the `no` command to change some parameters, such as sb_max, tcp_sendspace, tcp_recvspave, udp_sendspace, udp_recvspace, and so on. We can add these changes to the /etc/rc.net file, and other options can be tuned according to your real environment.

3.4.3 WAN considerations

When planning the WAN, we need to choose the network type, network bandwidth, and so on. WAN is a complex topic; we do not intend to talk about the media or technology here because there is much information, both books and on the Web, that covers this in more detail. We assume that you have skilled resources to verify your design criteria based upon your geomirroring requirements.

3.4.3.1 Network type

There are three basic types of network used over a wide area. These are:

- **Point-to-point networks** - This kind of network is also known as a lease line, such as T1, E1, T3, OC1, OC3, and so on.

- **Circuit switched networks**, such as ISDN.
- **Packet switched networks** - Some examples are ATM, Frame Relay, SMDS, and X.25.

In selecting the best overall solution, we must consider the entire corporate topology, not just a few discrete links. We can use the estimated bandwidth to help us make decisions. We need to think about the cost, security, reliability, maintainability, and consistency with international standards. HAGEO and GeoRM require a permanent connection because even though there is no geomirroring data transferred, the network is still carrying the keep alive messages; so, the point-to-point network, ATM, Frame Relay, and SMDS will be good choices if they meet the other requirements. ISDN is a generic term. In terms of narrowband ISDN, we can look at Frame Relay; In terms of broadband ISDN, we can look at ATM. The major concern is whether ISDN's router-to-router link can support IP. Some ISDN providers do not allow IP; they insist on protocols, such as PPP, which will not allow KRPC's UDP transport to run. In this case, we cannot use ISDN; If an ISDN link is used for voice and data, it will interfere with the available bandwidth. Also, consider that many ISDN implementations at the low end are in the 64 to 128 Kbps bandwidth range, which is likely to be insufficient for most HAGEO/GeoRM users unless the applications have low enough disk activity to stay below these numbers (64 to 128 Kbps) and gmdsizing has verified it.

3.4.3.2 Primary geographic network consideration

The Primary Geographic networks are dedicated to carrying geomirroring data; clients data should be not allowed. To prevent the SPOF, at least two separate communication paths are needed. If one of them goes down, we still have the backup. If there is only one router in each site to connect both geographic networks, the router is the SPOF; so, ideally, these two network paths need to be connected to different routers in each site described in Figure 19 on page 46. Otherwise, once the router goes down, all Geo networks are unavailable.

Another advantage is that GeoMessage can switch transmissions between these two communication paths. The GeoMessage device driver periodically measures the round-trip time on each of the networks defined to it. It then uses this information to determine which network is to be used to send KRPC packets across. But, only one network is used at a time regardless of how the two Geo Primary networks are configured to the cluster. In other words, GeoMessage does not load balance across these networks. See Section 3.4.4, "Network device considerations" on page 50.

3.4.3.3 Site isolation consideration for HAGEO

If both geographic networks go down and there is no other way for two sites to exchange keep-alive messages, then site isolation occurs. To prevent site isolation, we need another way of keeping the communication. Besides the Primary Geographic networks, we can use the Secondary Geographic network or the Dial Back Fail Safe facility to identify site isolation.

3.4.3.4 Geographic network performance considerations

Geographic network performance is related to the network bandwidth and network latency.

For common network media bandwidth data, you can refer to Table 20 on page 78. You can use the estimated bandwidth value, which we will discuss in Section 3.7, “Estimating necessary geographic network bandwidth” on page 54, to choose the appropriate one. One thing that needs to be understood is that you should never expect network traffic to be as fast as the indicated throughput. For example, the standard transfer rate of E1 is 2 Mbps. It will be very good if we can get 80 percent (200 Kbps) efficiency taking the overhead into account. We also need to take cost into account.

Network bandwidth is often seen as the key (and often the only) criteria that determines how an HAGEO cluster is designed. In fact, for the vast majority of workloads, it is latency and not network bandwidth that is the real constraining factor in designing a cluster. Consider the following workload examples:

- Writing 10 X 4 KB blocks per second = 40 KBps
- Writing 80 X 4 KB blocks per second = 320 KBps
- Writing 60 X 16 KB blocks per second = 960 KBps
- Writing 40 X 128 KB blocks per second = 5120 KBps

In the above examples, despite having significantly differing throughput characteristics, only the last example is likely to benefit from a higher bandwidth network. A 10 Mbps Ethernet network is good for up to approximately 1250 Kbps in a streaming environment. Consequently, increasing the bandwidth to 100 Mbps has no perceivable effect. In this case, you need to think about the latency; it is also used to measure round-trip time, which is used by GeoMessage to choose the communication path.

3.4.4 Network device considerations

As we have discussed, in order to prevent SPOF, we suggest that you use separate routers to connect different geographic networks. Also, give the same consideration to switches or hubs if they exist.

One important point is that the Proxy ARP argument of the router must be turned off. This is because we use a multicast environment where the keep-alive packets sent from one site to another appear as if they are coming from the originating host instead of from the router.

Another point is the router load balancing. There are two commonly-used load balancing mechanisms using the so-called per-destination and per-packet algorithms. However, there are definite implications for using these parallel links for Geo Primary networks.

Using a per-destination algorithm, all packets for a given destination are forwarded along the same path. This preserves the packet ordering but does not make equal use of the links. If one host receives the majority of the traffic, all packets will use one link. The other links remain unused. In an HAGEO environment, this may mean that the bandwidth of the Geo Primary is limited once again to that of a single E1 or T1 link. More equitable use may be made of the links by performing routing on a destination host rather than a destination network. Traffic for different hosts on the same destination network can then use the different paths. This is especially useful in larger HAGEO configurations with multiple active nodes at each site.

Per-packet load balancing guarantees an equal load across all links by allocating packets on a round-robin basis; However, it is possible for the packets to arrive out-of-order at the destination because different delays may exist between the different E1/T1 links. Packets arriving out of order cause the GeoMessage layer to perform error recovery most likely involving packet retransmission. This then causes every subsequent packet to be blocked pending error recovery completion. When out-of-order packet delivery occurs, the performance of the Geo_Primary link may fall almost to zero.

Such an environment should *never* be used to provide a Geo_Primary network unless the potential for out-of-order packet delivery is very well understood and the implications of its occurrence on geomirror performance are fully accepted.

In most cases, the routers are used not only by the geographic network, but also for other purposes, such as LAN or internet access. These other uses may limit the degree of freedom we have in configuring the routers.

3.4.5 UDP consideration

As we have said often: UDP is important for the HAGEO or GeoRM. Now, let us talk about it.

As we know, GeoMessage uses KRPC. The KRPC packet header looks like that shown in Figure 20.

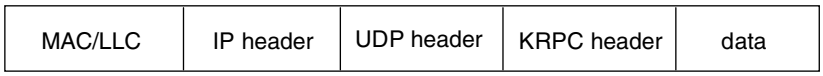

Figure 20. KRPC packet header

HAGEO or GeoRM use UDP/IP as their underlying transport protocol. UDP provides a low-cost protocol for applications that have the facilities to deal with communication failures. In the HAGEO/GeoRM case, the KRPC layer provides these facilities to ensure that packets are received and processed in the correct order. Also, KRPC needs to properly handle the packets lost or dropped situation. UDP does not acknowledge the receipt of packets; so, KRPC has to do this and deal with any required retransmission of packets.

In this case, the UDP socket send and receive buffer becomes very important. You need to set the appropriate value of udp_sendspace, udp_recvspace, and sb_max. You can get the geographic network communication status report using the `krpcstat` command, which is described in Section 3.7.2, “krpcstat” on page 58. If it specifies the retransmission errors caused by insufficient space in those buffers, you need to increase the buffers with the `no` command.

GeoMessage uses ports 6755; we can find these two lines in the /etc/services file:

```
gmd_port 6755/TCP
gmd_port 6755/UDP
```

Make sure there are no other applications using this port.

3.4.6 IP subnetting consideration in HAGEO cluster

This is the special consideration for HAGEO: When configuring your HAGEO networks, you should know that your geographical networks must be seen as one logical network across the geography. This is because all adapters in the network on all nodes at both sites are part of one HACMP network. HACMP’s configuration rule is that all adapters in a single HACMP network must be in the same logical subnet. Since normal TCP/IP configuration puts adapters in

different physical locations on different logical subnets, the HAGEO network setup requires some special configuration. This special configuration can be accomplished through netmasks. A simple example is shown in Figure 21.

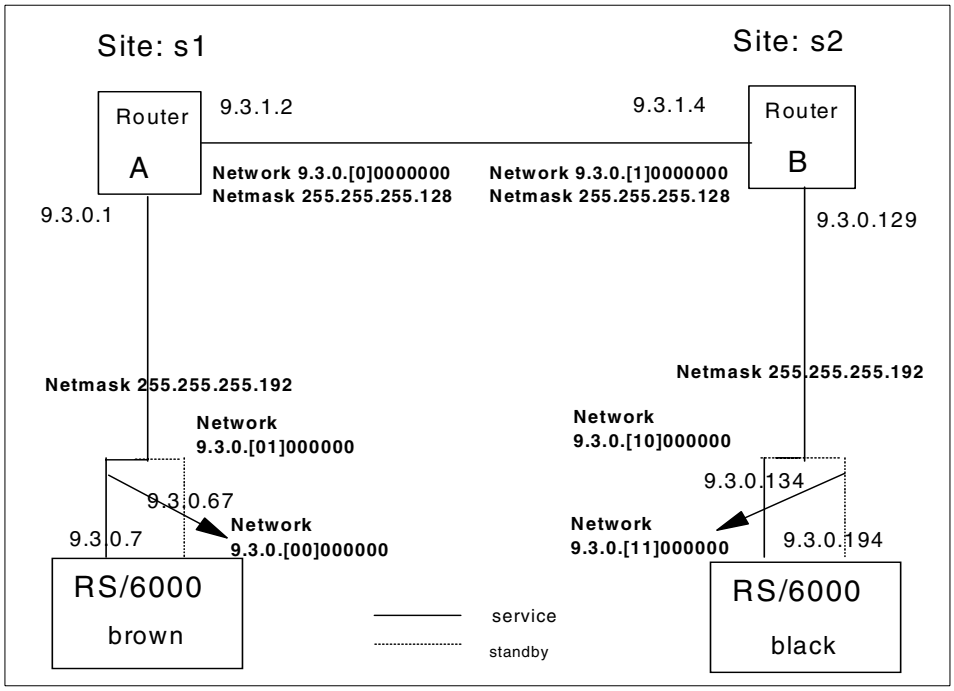

Figure 21. IP subnet in HAGEO networks

Figure 21 shows two sites: S1 and S2, each with one node. The geographic network includes one router at each site between the two nodes. Here is a listing of the interfaces and IP addresses of the nodes on the geographic network:

- Site 1 Node brown:

  9.3.0.7 - service, netmask 255.255.255.192

  9.3.0.67 - standby, netmask 255.255.255.192

- Site 2 Node black:

  9.3.0.134 - service, netmask 255.255.255.192

  9.3.0.194 - standby, netmask 255.255.255.192

The following is a list of the interfaces and IP addresses for the routers on the geographic network:

- Site 1 Router A

  9.3.0.1 - netmask 255.255.255.128

- Site 2 Router B

  9.3.0.129 - netmask 255.255.255.128

You have to add routing tables on nodes brown and black. The routing tables should look like those listed in Table 6. Note that the routers are set up as listed previously. Proxy ARP must be turned off.

Table 6. List of routing tables for HAGEO nodes

Node	Network	Netmask	Gateway	Interface
brown	0.0.0.0	0.0.0.0	9.3.0.1	9.3.0.7
A	0.0.0.0	0.0.0.0	9.3.1.4	9.3.1.2
	9.3.0.128	255.255.255.128	9.3.1.4	9.3.1.2
B	0.0.0.0	0.0.0.0	9.3.1.2	9.3.1.4
	9.3.0.0	255.255.255.128	9.3.1.2	9.3.1.4
black	0.0.0.0	0.0.0.0	9.3.0.129	9.3.0.134

3.5 Cost considerations

As we just described, when we implement a disaster recovery solution using HAGEO or GeoRM, we need high-end servers, sufficient disk subsystems, two geographic networks, routers, and so on. It might cost a great deal. Very often, customers will complain about this. Just face it: Any disaster recovery solution is expensive. Chances are, if you think that HAGEO or GeoRM are expensive, you have not worked out what the outage would really cost you. Sometimes, customers claim that they can afford the lost data; so, they select asynchronous mode. We are always surprised how many say they can and how many actually cannot. Therefore, considering the benefits, HAGEO or GeoRM are worth trying. By the way, not all the money is spent on this solution. Even if you do not think about HAGEO or GeoRM, you need to buy new servers and storage products for your business growth requirements, and communications costs will most likely dwarf the cost of anything else. Do it; you will be surprised!

3.6 Application consideration

In a normal situation, we have no control over the application. However, the application is very important for the performance of an HAGEO cluster or a GeoRM configuration. Sometimes, customers complain that performance is poor after HAGEO or GeoRM installation, but we find the same problem when this application runs on a stand-alone machine.

When an application is running with geographic mirroring, it is obvious that it cannot run as fast as when it is on a stand-alone machine. It is always a good idea to first test the application in an environment without HAGEO or GeoRM. This can help us get some feel for how the application performs: Is it running well in this stand-alone environment? If we find the application has I/O-bound or CPU-bound problems, how can we expect it to perform well in an HAGEO or GeoRM environment? Therefore, the first thing we suggest is to build a stand-alone environment of the system on a single RS/6000 server. We can do some performance measurements and simple tests; then we will have a baseline to which to refer. Some tools will be discussed in Section 3.7, “Estimating necessary geographic network bandwidth” on page 54, but the `filemon` command will be more useful than other utilities in this situation because the `filemon` command is an AIX tool. The `gmdsizing`, `gmdstat`, and `krpcstat` commands are only available after HAGEO or GeoRM is installed.

Assume data01 and data02 are the logical volumes used by the applications and would be geographically mirrored later in the HAGEO cluster. After we have installed the application in a stand-alone machine, we can use the `filemon` command to collect data about data01 and data02 activity during a reasonable period using following command:

```
# filemon -o /tmp/filemon.out -O lv
```

The data in `/tmp/filemon.out` can be analyzed to estimate the network bandwidth we need for this application. The `filemon` command, its usage, and examples of the bandwidth requirement calculations are described in Section 3.7.5, “Using filemon to estimate network bandwidth” on page 77.

3.7 Estimating necessary geographic network bandwidth

We have discussed many of the concerns involved in designing an HAGEO or GeoRM. This section introduces some utilities to estimate the necessary geographic network bandwidth for a given application to be mirrored via a geographic network. There are four useful commands we can use to monitor disk usage, GeoMessage performance, and GeoMirror device performance.

They provide statistical reports to help you find the necessary network bandwidth. These utilities are:

- `/usr/sbin/gmd/gmdsizing`
  Monitors disk usage on your system over a period of time and prints a statistical report
- `/usr/sbin/krpc/krpcstat`
  Monitors GeoMessage performance and prints a statistical report
- `/usr/sbin/gmd/gmdstat`
  Monitors GeoMirror device performance and prints a statistical report
- `/usr/sbin/filemon`
  Monitors a trace of file system and I/O system events and reports on the file

3.7.1 gmdsizing

Use the /usr/sbin/gmd/gmdsizing utility to evaluate current disk usage on your system. This utility supplies information that can help you choose geographic network bandwidth.

The gmdsizing utility can monitor either a set of physical disks or a set of volume groups, and you can specify the amount of time you want to measure. When the gmdsizing process is interrupted, or when it completes, it generates a statistical report on disk utilization over the time the command was running.

The statistical report includes information on the average number of KBs written per second as well as the peak values and low values. It also consolidates all the specified disks into a system-wide average/peak/valley KB written.

The syntax and arguments of the `gmdsizing` command are as follows.

/usr/sbin/gmd/gmdsizing -i -t {[**-p** pv...] [**-v** vg...]} **-f** filename [**-h**] [**-o** odmdir] -**V**

The arguments of the `gmdsizing` command are listed in Table 7.

Table 7. Arguments of the gmdsizing command

Arguments	Description
-i *interval*	Sampling interval for disk activity

Arguments	Description
-t *time*	Time duration the command should sample. The unit of time defaults to seconds. The minimum number of seconds is 10. The argument can be appended with the following letters to change the unit of time. *number of days* **d** *number of minutes* **m** *number of hours* **h** *number seconds* **s** For example, to check over five days, you could use: 5d, 120h, or 7200m.
-p *pv_name*	Name of physical disks to monitor (can be more than one).
-v *vg_name*	Name of volume groups to monitor (can be more than one).
-f *filename*	File to store report, instead of writing to stdout.
-h	Help message.
-o *odmdir*	ODM directory (other than default).
-V	Verbose mode. Adds summary at end of the report.

Some examples follow:

- To measure disk usage on hdisks 00 and 01 once a minute for 24 hours, enter:

```
# /usr/sbin/gmd/gmdsizing -i 60 -t 24h -p hdisk00 -p hdisk01
```

- To measure disk usage for the volume groups named db1vg and db2vg once a minute for 24 hours, enter:

```
# /usr/sbin/gmd/gmdsizing -i 60 -t 24h -v db1vg -v db2vg
```

- To measure disk usage on hdisk1 once every five minutes for 300 hours and get a verbose report, enter:

```
# /usr/sbin/gmd/gmdsizing -i 300 -t 300h -p hdisk1 -V
```

- To measure disk usage for the volume group named db1vg and the physical disk hdisk8 once every two minutes for 12 hours, enter:

```
# /usr/sbin/gmd/gmdsizing -i 2m -t 12h -v db1vg -p hdisk8
```

As we stated earlier, the `gmdsizing` utility will generate a statistical report. The following example shows how we can use this information to estimate the bandwidth requirement:

```
# /usr/sbin/gmd/gmdsizing -i 1m -t 5m -p hdisk0 -f /tmp/gmdsizing.out -V
```

We get the sample statistical report shown in Figure 22.

```
Disk                     Reads               Writes
hdisk0                     0                 689
=======================================================================
Disk                     Reads               Writes
hdisk0                  1792                 732
=======================================================================
Disk                     Reads               Writes
hdisk0                     0                 689
=======================================================================
Disk                     Reads               Writes
hdisk0                     8                 637
=======================================================================
Disk                     Reads               Writes
hdisk0                   112                 624
=======================================================================
            block       total            minimum           maximum
Disk        size     read    write     read    write     read    write
hdisk0       512     1912     3371        0      624     1792     732
```

Figure 22. gmdsizing sample statistical report

We can calculate that the read/write ratio is 1912:3371. This is approximately 36 percent read and 64 percent write. Notice that very write-intensive and very read-intensive environments are not ideally suited to HAGEO or GeoRM.

For the purpose of sizing the networks, we are only interested in write traffic. We know the period and the total write number; so, we can get an average write rate. Remember that all data reported by gmdsizing is given in disk blocks.

3371 X 512 = 1725952 (bytes)

1725952 / 300 = 5753 (bytes /second)

This is the average. Let us look at the worst case. Using the maximum write value, divide the interval. We get the value:

732 X 512 = 374784 bytes

1040384 / 60 = 6246 bytes/second

Compare these two sets of values. If they are relatively close, we are likely to have a fairly even network utilization just as in this example. If they differ widely (having a worst-case data requirement of between seven and 10 times that of the average is not unusual), we say we have uneven network utilization; so, we need to do a more detailed analysis to determine the network bandwidth requirement. From this data, we can get a feel for the

network. In this example, the sample period is a short time, and we just use it to introduce the method.

The bandwidth ought to be:

5753 x 8 = 46 Kbps

46 / 0.8 = 58 Kbps (Assume 20 percent network overhead)

and

6246 x 8 = 50 Kbps

50 / 0.8 = 63 Kbps (Assume 20 percent network overhead)

Then, we get the rough range of bandwidth as 58 Kbps to 63 Kbps.

The sampling period in this example is short; we just use it to illustrate the method. Normally, we need to collect data for a long time for a real application (one day, two days, or five days) and use a big interval; then, we can get closer results. To get more precise results, we can do it more than once.

We can also use SMIT to run the `gmdsizing` utility, and we will get the same report.

3.7.2 krpcstat

Use the `krpcstat` utility to evaluate GeoMessage performance. It monitors the geographic messaging performance and prints a statistical report.

The `krpcstat` command produces the following information:

- The amount of data sent and received per network since the last time interval measured in Kb, Kbps, RPCs and RPCs/sec. The number of RPCs reported includes all fragmented RPCs but excludes resends.
- The degree of RPC fragmentation since boot time, reported as the number of original requests, the number of fragmented requests, the number of fragments, and the percentage of fragmentation. The percentage of fragmentation is defined by the total number of original requests versus the number of those requests that were fragmented.
- The degree of fragmentation since the last time interval reported as the number of original requests, the number of fragmented requests, the number of fragments, and the percentage of fragmentation. The

percentage of fragmentation is defined by the total number of original requests versus the number of those requests that were fragmented.

- The number of failures and time-outs per network since boot time and since the last time interval.
- The number of resends per network since boot time and the last time interval.
- The number of duplicate packets received per network since boot time and the last time interval.
- Request dispatch latency (if applicable).
- Request round-trip times per network.

The syntax and arguments of the `gmdsizing` command follow.

usr/sbin/krpc/krpcstat [**-s**] [**-m** *machine*...] [**-n** network...] [**-i** interval] [**-c** count] [**-v**] [**-h**]

Table 8 lists the arguments of the `krpcstat` command.

Table 8. Arguments of the krpcstat command

Arguments	Description
-s	Print a summary report. (By default, it prints a detailed report.)
-m *machine*	The name of a GeoMessage machine to monitor. If no machines are listed, the utility reports on all machines.
-n *network*	The name of a GeoMessage network to monitor. If no networks are listed, the utility reports on all networks.
-i *interval*	The number of seconds in the sampling interval. The default interval is 1 second.
-c *count*	The number of intervals for which to report. The default is one interval.
-v	The version number.
-h	The help message.

Two examples follow.

- To see one summary report for the performance of a machine named brown, enter

  ```
  # /usr/sbin/krpc/krpcstat -s -m brown
  ```

- To see a report for five intervals of 10 seconds for the performance of all geomirroring machines and networks, enter

```
# /usr/sbin/krpc/krpcstat -i 10 -c 5
```

A detailed report on two machines with low activity is shown in Figure 23.

```
 Machines                        KRPC Statistics
---------- ------------------------------------------------------------------
               rt     Orpc (Orpc/s)     Irpc      Okb (Okb/s)      Ikb
brown 0.00        0 (  0.00)        0      0.0 (0.00)      0.0

           replies     failures      rpcs fragd      frags    % frag
                 0            0               0          0        0.00

               rt     Orpc (Orpc/s)     Irpc      Okb (Okb/s)      Ikb
black 0.00        1 ( 0.00)        0      0.7 (0.00)      0.0

           replies     failures      rpcs fragd      frags    % frag
                 0            0               0          0        0.00

    rt=retries, Orpcs = output remote procedure calls,
    Irpc=number of input RPCs. kb = kilobytes, Ikb=input KB
    rpcs fragd = number of requests fragmented, frags=number of fragments,
  %frag = frag/total packets

 Networks                        KRPC Statistics
---------- ------------------------------------------------------------
                 snd (snd/s)       rcv       Okb (Okb/s)       Ikb
geonet1            1 (0.00)         0       0.9 (0.00)       0.0

           resends      timeouts       failures       dups       other
                 3             4              0          0          29

                 snd (snd/s)       rcv       Okb (Okb/s)       Ikb
tok1               0 (0.00)         0       0.0 (0.00)       0.0

           resends      timeouts       failures       dups       other
                 5             4              0          0           0

    snd=number of requests sent, rcv=number of receives
    Okb=KB of data written, Ikb=input KB, resends= number of packets
    sent over another network, timeouts=#of packets sent without
    receiving ack, failures=#packets that exceed allowed # resends
    dups= number of duplicate packets
```

Figure 23. KRPC sample statistics report

We can calculate the network's performance from this data. Remember the the overhead. Another thing we need to pay attention to is the resend, timeout, and failure items listed previously. If we get many resends, timeouts, or failures, we can check the UDP socket buffer first. Does the buffer overflow? If it does, we need to increase the size of the buffer. Another

reason is that the network is likely to be unreliable. HAGEO and GeoRM require a reliable network. This is a problem that needs to be solved by the network provider. We suggest running this command when you do the synchronization.

After the `krpcstat` command is executed, it will return one code to indicate the result of execution. The exit code will show the status of `krpcstat` execution. For example, if the exit code is 0, it means the execution of `krpcstat` is complete and no errors were encountered. These exit codes are listed in Table 9.

Table 9. Exit codes used by the krpcstat command

Exit codes	Description
-1	A system call or API call failed. An error message describes the failure.
0	No errors were encountered.
1	An illegal combination of switches was specified.
2	User-specified information is not available.

3.7.3 gmdstat

Use the `gmdstat` utility to evaluate the performance for each GeoMirror device and print a statistical report. The `gmdstat` command produces the following information for each device. Note that data written locally includes all write requests to the local device that originate from the local host. Data written remotely describes all write requests to the local device that originate from a remote host including:

- The amount of data written locally and remotely since boot time measured in KB, KB/second, requests, and requests/second.
- The amount of data written locally and remotely since the last time interval measured in KB, KB/second, requests, and requests/second.
- The amount of time spent satisfying KRPC requests since boot time.
- The amount of time spent satisfying KRPC requests since the last time interval.
- The amount of data sent to and received from KRPC for each remote host since boot time measured in KB, KB/second, requests, and request/second.

- The amount of data sent to and received from KRPC for each remote host since the last time interval measured in KB, KB/second, requests, and request/second.

The syntax and arguments of the `gmdstat` command are as follows:

/usr/sbin/gmd/gmdstat [**-s**] {**-A** | **-l** gmd [**-l** gmd...]} [**-i** interval] [**-c** count] [**-v**] [**-h**]

Table 10 lists the arguments of the `gmdstat` command.

Table 10. Arguments of the gmdstat command

Arguments	Description
-s	Print summary report. (By default, prints detail.)
-A	Monitor all GMD devices.
-l *gmd_name*	Name of a GMD to monitor.
-i *interval*	Number of seconds in the sampling interval. The default interval is 1 second.
-c *count*	Number of intervals to report for. The default is 1 interval.
-v	Version number.
-h	Help message.

Two examples follow.

- To monitor the device named gmd0 and print a detailed report, enter:

```
# /usr/sbin/gmd/gmdstat -l gmd0
```

- To monitor for two intervals of 30 seconds each and report on all GMDs defined in the ODM, enter:

```
# /usr/sbin/gmd/gmdstat -A -i 30 -c 2
```

A sample report (with an inactive GMD) is shown in Figure 24 on page 63.

```
GMDs                                  Device Statistics
---- ------------------------------------------------------------------------
                  lx       lx/s        lw        lw/s       rx       rw
gmd3              17943    4.37        70381.0   15.28      678      0.0

       Hosts                          KRPC Statistics
      -------  ------------------------------------------------------------
                  xm       xm/s     rt      rp       kw     kw/s     kr
       brown      164771   4.13     0.00    663      63231  13.79    0.0

    lx=local transfers, lx/s=local transfers/sec
    lw=local writes (KB), local writes/sec, rx=remote transfers,
    rw=remote writes (KB), xm=KRPC transmits, xm/s=KRPC transmits
    per sec, rt=retries, rp=replies, kw=KRPC writes (KB),
    kw/s = KRPC writes /sec, kr=KRPC reads
```

Figure 24. Sample output for the gmdstat command

After the `gmdstat` command is executed, it will return one code to indicate the result of execution. The exit code will show the status of `gmdstat` execution. For example, if the exit code is `0`, it means the execution of `gmdstat` is complete and no errors were encountered. These exit codes are listed in Table 11.

Table 11. List of exit codes used by the gmdstat command

Exit code	Description
-1	A system call or API call failed. An error message describes the failure.
0	No errors were encountered.
1	An illegal combination of switches was specified.
2	User-specified information is not available.

3.7.4 filemon

Another way of gathering the information you need to estimate bandwidth and measure the application is to use the `filemon` command. Compared to the `gmdsizing` command, we have found the `filemon` command to be more useful. The `gmdsizing` command allows you to measure disk usage with a granularity that is only to the disk or volume group level. The `filemon` command is much more flexible allowing you to collect and present trace data on the various layers of file system utilization including the logical file system, virtual memory segments, LVM, and physical disk layers.

3.7.4.1 Introduction of the filemon command

The `filemon` command is not an AIX base operating system command. It is contained in the command sets of the Performance Toolbox for AIX LPP. It monitors the performance of the file system and reports the I/O activity on logical files, virtual memory segments, logical volumes, and physical volumes.

The `filemon` command monitors a trace of file system and I/O system events and reports on the file and I/O access performance during that period.

In its normal mode, the `filemon` command runs in the background while one or more application programs or system commands are being executed and monitored. The `filemon` command automatically starts and monitors a trace of the program's file system and I/O events in real time. By default, the trace is started immediately; optionally, tracing may be deferred until the user issues a `trcon` command. The user can issue `trcoff` and `trcon` commands while the `filemon` command is running in order to turn monitoring off and on as desired. When tracing is stopped by a `trcstop` command, the `filemon` command generates an I/O activity report and exits.

The `filemon` command can also process a trace file that has been previously recorded by the AIX trace facility. The file and I/O activity report will be based on the events recorded in that file.

To provide a more complete understanding of file system performance for an application, the `filemon` command monitors file and I/O activity at four levels:

- **Logical file system** - The `filemon` command monitors logical I/O operations on logical files. The monitored operations include all read, write, open, and lseek system calls, which may or may not result in actual physical I/O, depending on whether or not the files are already buffered in memory. I/O statistics are kept on a per-file basis.
- **Virtual memory system** - The `filemon` command monitors physical I/O operations (that is, paging) between segments and their images on disk. I/O statistics are kept on a per-segment basis.
- **Logical volumes** - The `filemon` command monitors I/O operations on logical volumes. I/O statistics are kept on a per-logical volume basis.
- **Physical volumes** - The `filemon` command monitors I/O operations on physical volumes. At this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical volume basis.

As specified by the command line flags, any combination of the four levels can be accepted. By default, the `filemon` command only monitors I/O

operations at the virtual memory, logical volume, and physical volume levels. These levels are all concerned with requests for real disk I/O.

The `filemon` command writes its report to standard output or to a specified file. The report begins with a summary of the I/O activity for each of the levels being monitored and ends with detailed I/O activity statistics for each of the levels being monitored.

Note

- The reports produced by the `filemon` command can be quite long. Consequently, the `-o` option should usually be used to write the report to an output file.
- When a physical device is opened and accessed directly by an application, only reads and writes of complete 512-byte blocks are reflected in the report. Short reads and writes used by the device driver to issue device commands and read device status are ignored.
- CD-ROMs, unlike hard files, do not have concentric tracks or cylinders (there is one spiral track). Consequently, it is not possible to report seek distance statistics for CD-ROMs in terms of cylinders.
- The `-u` flag is used to generate reports on files opened prior to the start of the trace daemon. Some of this data can be useful, but much of it applies to daemons and other unrelated activity. This background information can be overwhelming, especially on large systems.
- If the /unix file and the running kernel are not the same, the kernel addresses will be incorrect causing the `filemon` command to exit.
- When using the `filemon` command from a shell script, allow for a slight delay prior to viewing the contents of the output file. The `filemon` command may take a few seconds to produce this report.

The syntax and arguments of the `filemon` command are as follows:

filemon [**-d**] [**-i** File] [**-o** File] [**-O** Levels] [**-P**] [**-T** n] [**-u**] [**-v**]

Table 12 lists arguments of the `filemon` command.

Table 12. Arguments of the filemon command

Arguments	Description
-i *File*	Reads the I/O trace data from the specified file instead of the real-time trace process. The `filemon` report summarizes the I/O activity for the system and period represented by the trace file. **Note:** Trace data files are usually written in a circular manner. If the trace data has wrapped around, the chronological beginning and end of the trace may occur in the middle of the file. Use the raw mode of the `trcrpt` command to rewrite the data sequentially, before invoking the `filemon` command, as follows: `trcrpt -r file > new.file` For the report to be accurate, the trace file must contain all the hooks required by the `filemon` command.
-o *File*	Writes the I/O activity report to the specified file instead of the `stdout` file.
-d	Starts the `filemon` command, but defers tracing until the `trcon` command has been executed by the user. By default, tracing is started immediately.
-T *n*	Sets the kernel's trace buffer size to *n* bytes. The default size is 32,000 bytes. The buffer size can be increased to accommodate larger bursts of events, if any. (A typical event record size is 30 bytes.) **Note:** The trace driver in the kernel uses double buffering; so, in fact, there will be two buffers of size *n* bytes allocated. Also, note that these buffers are pinned in memory; so, they are not subject to paging. Large buffers may affect the performance of paging and other I/O.
-P	Pins monitor process in memory. The `-P` flag causes the `filemon` command's text and data pages to be pinned in memory for the duration of the monitoring period. This flag can be used to ensure that the real-time filemon process is not paged out when running in a memory-constrained environment.
-v	Prints extra information in the report. The most significant effect of the `-v` flag is that all logical files and all segments that were accessed are included in the I/O activity report, instead of only the 20 most active files and segments.

Arguments	Description
-O *Levels*	Monitors only the specified file system levels. Valid level identifiers are: **lf** Logical file level **vm** Virtual memory level **lv** Logical volume level **pv** Physical volume level **all** Short for **lf**,**vm**,**lv**,**pv** The **vm**, **lv**, and **pv** levels are implied by default.
-u	Reports on files that were opened prior to the start of the `trace` daemon. The process ID (PID) and the file descriptor (FD) are substituted for the file name. Since PIDs and FDs are reusable, it is possible to see different files reported with the same name field.

Some examples of the usage of the `filemon` command follow:

- To monitor the physical I/O activity of the virtual memory, logical volume, and physical volume levels of the file systems, enter

  ```
  # filemon
  ```

  The `filemon` command automatically starts the system trace and puts itself in the background. After this command, enter the application programs and system commands to be run at this time, then enter

  ```
  # trcstop
  ```

  After the `trcstop` command is issued, the I/O activity report is displayed on standard output (but will probably scroll off the screen). The virtual memory I/O report will be limited to the 20 segments that incurred the most I/O.

- To monitor the activity at all file system levels and write the report to the `fmon.out` file, enter

  ```
  # filemon -o fmon.out -O all
  ```

  The `filemon` command automatically starts the system trace and puts itself in the background. After this command, enter the application programs and system commands to be run at this time, then enter

  ```
  # trcstop
  ```

  After the `trcstop` command is issued, the I/O activity report is written to the `fmon.out` file. All four levels of the file and I/O system (the logical file, virtual memory, logical volume, and physical volume levels) will be monitored. The logical file and virtual memory I/O reports will be limited to the 20 files and segments (respectively) that incurred the most I/O.

- To monitor the activity at all file system levels and write a verbose report to the `fmon.out` file, enter

  ```
  # filemon -v -o fmon.out -O all
  ```

  The `filemon` command automatically starts the system trace and puts itself in the background. After this command, enter the application programs and system commands to be run at this time, then enter

  ```
  # trcstop
  ```

  This example is similar to the previous example, except that a verbose report is generated to the fmon.out file. The primary difference is that the `filemon` command will indicate the steps it is taking to start the trace, and the summary and detailed reports will include *all* files and segments that incurred any I/O (there may be many), instead of just the top 20.

- To report on I/O activity captured by a previously-recorded trace session, enter:

  ```
  # filemon -i trcfile |pg
  ```

- To monitor the I/O activity only for logical and physical volumes while controlling the monitored intervals using the `trcon` and `trcoff` commands, enter

  ```
  # filemon -d -o fmon.out -O pv,lv
  ```

  The `filemon` command automatically starts the system trace and puts itself in the background. After this command, you can enter the monitored application programs and system commands to be run at this time, then enter

  ```
  # trcon
  ```

  After this command, you can enter the monitored application programs and system commands to be run at this time, then enter:

  ```
  # trcoff
  ```

  After this command, you can enter the unmonitored application programs and system commands to be run at this time, then enter:

  ```
  # trcon
  ```

  After this command, you can enter the monitored application programs and system commands to be run at this time, then enter:

  ```
  # trcstop
  ```

  In this example, the `-O` flag is used to restrict monitoring to logical and physical volumes only. Only those trace events that are relevant to logical and physical volumes are enabled. Also, as a result of using the `-d` flag, monitoring is initially deferred until the `trcon` command is issued. System

tracing can be intermittently disabled and reenabled using the `trcoff` and `trcon` commands so that only specific intervals are monitored.

3.7.4.2 Reports from the filemon command

The following command gives a simple example of `filemon` usage, which is commonly used to monitor GMD's activity. `/fs2` is a file system mounted over the /fs2lv. The output is customized for easy reading.

```
# filemon -o fmon.out -O lv,pv; cp /tmp/testfile /fs2; trcstop
Wed Nov 24 14:48:04 1999
System: AIX brown Node: 4 Machine: 000001885C00
2.271 secs in measured interval
Cpu utilization:  41.7%
```

The `2.271 secs` is the elapsed time in seconds that the trace was active. The CPU utilization is 41.7 percent during the 2.271 seconds.

```
Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume               description
------------------------------------------------------------------------
  0.24      0   4544 1000.3  /dev/fs2lv           gmd
  0.07      0    480  105.7  /dev/fs2smlv         sm
  0.00      0      8    1.8  /dev/fs2loglv        jfslog
Most Active Physical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume               description
------------------------------------------------------------------------
  0.23      0   4552 1002.0  /dev/hdisk3          IBM 7137-412
  0.06      0    480  105.7  /dev/hdisk4          IBM 7137-412
------------------------------------------------------------------------
Detailed Logical Volume Stats   (512 byte blocks)
------------------------------------------------------------------------
VOLUME: /dev/fs2lv  description: gmd
writes:142(0 errs)
  write sizes (blks): avg    32.0 min      32 max      32 sdev     0.0
  write times (msec):avg   3.897 min   3.715 max   4.886 sdev   0.189
  write sequences: 3
  write seq. lengths:avg  1514.7 min      64 max    4384 sdev  2029.0
seeks:3(2.1%)
  seek dist (blks):init     80,
avg   136.0 min       8 max     264 sdev   128.0
time to next req(msec): avg  15.727 min   4.594 max 1345.326 sdev 111.996
throughput:1000.3 KB/sec
utilization:0.24
```

```
VOLUME: /dev/fs2smlv  description: sm
writes:60(0 errs)
  write sizes (blks): avg     8.0 min       8 max       8 sdev     0.0
  write times (msec):avg   2.498 min   2.335 max   3.772 sdev   0.290
  write sequences: 60
  write seq. lengths:avg     8.0 min       8 max       8 sdev     0.0
seeks:60(100.0%)
  seek dist (blks):init    256,
avg     8.0 min       8 max       8 sdev     0.0
time to next req(msec): avg  37.143 min  11.685 max 1456.922 sdev 184.862
throughput:105.7 KB/sec
utilization:0.07

VOLUME: /dev/fs2loglv  description: jfslog
writes:1(0 errs)
  write sizes (blks): avg     8.0 min       8 max       8 sdev     0.0
  write times (msec):avg   2.428 min   2.428 max   2.428 sdev   0.000
  write sequences: 1
  write seq. lengths:avg     8.0 min       8 max       8 sdev     0.0
seeks:1(100.0%)
  seek dist (blks):init     64
time to next req(msec): avg 1329.086 min 1329.086 max 1329.086 sdev   0.000
throughput:1.8 KB/sec
utilization:0.00
--------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
--------------------------------------------------------------------------
VOLUME: /dev/hdisk3  description: IBM 7137-412
writes:143(0 errs)
  write sizes (blks): avg    31.8 min       8 max      32 sdev     2.0
  write times (msec):avg   3.695 min   2.236 max   4.696 sdev   0.224
  write sequences: 4
  write seq. lengths:avg  1138.0 min       8 max    4384 sdev  1874.3
seeks:4(2.8%)
  seek dist (blks):init 225600,
avg  2824.0 min       8 max    8200 sdev  3802.8
  seek dist (%tot blks):init 10.76270,
avg 0.13472 min 0.00038 max 0.39120 sdev 0.18142
time to next req(msec): avg  15.618 min   4.598 max 1329.154 sdev 110.257
throughput:1002.0 KB/sec
utilization:0.23

VOLUME: /dev/hdisk4  description: IBM 7137-412
writes:60(0 errs)
  write sizes (blks): avg     8.0 min       8 max       8 sdev     0.0
  write times (msec):avg   2.303 min   2.145 max   3.578 sdev   0.290
  write sequences: 60
```

```
  write seq. lengths:avg     8.0 min       8 max       8 sdev     0.0
seeks:60(100.0%)
  seek dist (blks):init 487936,
avg     8.0 min       8 max       8 sdev     0.0
  seek dist (%tot blks):init 23.27797,
avg 0.00038 min 0.00038 max 0.00038 sdev 0.00000
time to next req(msec): avg  37.144 min  11.685 max 1456.985 sdev 184.870
throughput:105.7 KB/sec
utilization:0.06
```

In this example, we only list the logical volume and physical volume that concerns us for geographic mirroring. The output is composed of two different types of reports: Global and detailed. If we use the `-O all` option, we will get a more detailed report. In the following section, we will list the explanation of a verbose report.

Global reports

The global reports list the most active files, segments, logical volumes, and physical volumes during the measured interval. They are shown at the beginning of the filemon report. By default, the logical file and virtual memory reports are limited to the 20 most active files and segments as measured by the total amount of data transferred. If the `-v` flag has been specified, activity for all files and segments is reported. All information in the reports is listed from top to bottom, from most active to least active.

Table 13 lists the most active files.

Table 13. Most active files report

Column	Description
#MBs	The total number of megabytes transferred to or from a file. The rows are sorted by this field in decreasing order.
#opns	The number of times the file was opened during the measurement period.
#rds	The number of read system calls made against the file.
#wrs	The number of write system calls made against the file.
file	The name of the file (full path name is in a detailed report).
volume:inode	The name of the volume that contains the file and the file's i-node number. This field can be used to associate a file with its corresponding persistent segment shown in the virtual memory I/O reports. This field may be blank, for example, for temporary files created and deleted during execution.

Table 14 lists the most active segments.

Table 14. Most active segments report

Column	Description
#MBs	The total number of megabytes transferred to/from segment. The rows are sorted by this field, in decreasing order.
#rpgs	The number of 4096-byte pages read into segment from disk (that is page).
#wpgs	The number of 4096-byte pages written from segment to disk (page out).
segid	The internal segment ID.
segtype	The type of segment: Working segment, persistent segment (local file), client segment (remote file), page table segment, system segment, or special persistent segments containing file system data (log, root directory, .inode, .inodemap, .inodex, .inodexmap, .indirect, .diskmap).
volume:inode	For persistent segments, the name of the volume that contains the associated file and the file's inode number. This field can be used to associate a persistent segment with its corresponding file shown in the file I/O reports. This field will be blank for non-persistent segments. **Note:** The virtual memory analysis tool, svmon, can be used to display more information about a segment, given its segment ID (segid), as follows: `svmon -S <segid>`

Table 15 lists the most active logical volumes.

Table 15. Most active logical volumes report

Column	Description
util	The utilization of the volume (fraction of time busy). The rows are sorted by this field in decreasing order.
#rblk	The number of 512-byte blocks read from the volume.
#wblk	The number of 512-byte blocks written to the volume.
KB/sec	The total transfer throughput in kilobytes per second.
volume	The name of the volume.
description	The contents of the volume: Either a file system name or logical volume type (paging, jfslog, boot, or sysdump). This also indicates whether the file system is fragmented or compressed.

The utilization is presented as a percentage: 0.24 means 24 percent busy during the measured interval. The logical volume, /fs2lv, has 4544 blocks

written and has an average data transfer rate of 1000.3KBps. The logical volume, fs2smlv, has 480 blocks read and an average data transfer rate of 105.7KBps. The other LVs have low utilization and transfer rates. Table 16 lists the most active physical volumes.

Table 16. Most active physical volumes report

Column	Description
util	The utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.
#rblk	The number of 512-byte blocks read from the volume.
#wblk	The number of 512-byte blocks written to the volume.
KB/sec	The total volume throughput, in Kilobytes per second.
volume	The name of volume.
description	The type of volume. For example, 120 MB disk, 355 MB SCSI, or CD-ROM SCSI. Logical volume I/O requests start before and end after physical volume I/O requests. For that reason, total logical volume use will appear to be higher than the total physical volume use.

The headings have a meaning for physical volumes that is similar to that for logical volumes.

Detailed reports

Finally, detailed reports are generated for each logical volume and physical volume level being monitored. By default, the logical file and virtual memory reports are limited to the 20 most active files and segments as measured by the total amount of data transferred. If the -v flag has been specified, activity for all files and segments is reported. There is one entry for each reported file, segment, or volume. The fields in each entry are described below for the four detailed reports.

Some of the fields report a single value and others report statistics that characterize a distribution of many values. For example, response time statistics are kept for all read or write requests that were monitored. The average, minimum, and maximum response times are reported as well as the standard deviation of the response times. The standard deviation is used to show how much the individual response times deviated from the average. Roughly two-thirds of the sampled response times are between average -1 standard deviation and average + 1 standard deviation. If the distribution of response times is scattered over a large range, the standard deviation will be large compared to the average response time.

Detailed File Stats

Detailed file statistics are provided for each file listed in the Most Active Files report. These stanzas can be used to determine what access has been made to the file. In addition to the number of total bytes transferred, opens, reads, writes, and lseeks, the user can also determine the read/write size and times. Table 17 lists detailed file statistics.

Table 17. Detailed file stats report

Column	Description
FILE	The name of the file. The full path name is given, if possible
volume	The name of the logical volume/file system containing the file
inode	The I-node number for the file within its file system
opens	The number of times the file was opened while being monitored
total bytes xfrd	The total number of bytes read/written to/from the file
reads	The number of read calls against the file
read sizes (bytes)	The read transfer size statistics (avg/min/max/sdev), in bytes
read times (msec)	The read response time statistics (avg/min/max/sdev), in milliseconds
writes	The number of write calls against the file
write sizes (bytes)	The write transfer size statistics
write times (msec)	The write response time statistics
seeks	The number of lseek subroutine calls

Detailed VM Segment Stats

Each element listed in the Most Active Segments report will have a corresponding stanza that shows detailed information about real I/O to and from memory. Table 18 lists detailed VM segment statistics information.

Table 18. Detailed VM segment stats report

Column	Description
SEGMENT	The internal AIX segment ID.
segtype	The type of segment contents.
segment flags	The various segment attributes.
volume	For persistent segments, the name of the logical volume containing the corresponding file.

Column	Description
inode	For persistent segments, the i-node number for the corresponding file.
reads	The number of 4096-byte pages read into the segment (that is, paged in).
read times (msec)	The read response time statistics (avg/min/max/sdev), in milliseconds.
read sequences	The number of read sequences. A sequence is a string of pages that are read (paged in) consecutively. The number of read sequences is an indicator of the amount of sequential access.
read seq. lengths	Statistics describing the lengths of the read sequences, in pages.
writes	The number of pages written from the segment (that is, paged out).
write times (msec)	Write response time statistics.
write sequences	The number of write sequences. A sequence is a string of pages that are written (paged out) consecutively.
write seq. lengths	Statistics describing the lengths of the write sequences, in pages.

By examining the reads and read-sequence counts, you can determine whether the access is sequential or random. For example, if the read-sequence count approaches the reads count, the file access is more random. On the other hand, if the read-sequence count is significantly smaller than the read count and the read-sequence length is a high value, file access is more sequential. The same logic applies for the writes and write-sequence. In the example, segment IDs 0241, 196d, and 19cd have the same write and write-sequence counts; so, these corresponding file accesses are more random.

Detailed Logical/Physical Volume Stats

Each element listed in the Most Active Logical/Physical Volumes report will have a corresponding stanza that shows detailed information about the logical/physical volume. In addition to the number of reads and writes, the user can also determine read and write times and sizes and the initial and average seek distances for the logical/physical volume. Table 19 lists detailed logical and physical volume statistics information.

Table 19. Detailed logical/physical volume stats reports

Column	Description
VOLUME	The name of the volume.

Column	Description
description	The description of the volume. (Describes contents if discussing a logical volume, describes type if dealing with a physical volume.)
reads	The number of read requests made against the volume.
read sizes (blks)	The read transfer size statistics (avg/min/max/sdev), in units of 512-byte blocks.
read times (msec)	The read response time statistics (avg/min/max/sdev), in milliseconds.
read sequences	The number of read sequences. A sequence is a string of 512-byte blocks that are read consecutively and indicate the amount of sequential access.
read seq. lengths	Statistics describing the lengths of the read sequences, in blocks.
writes	The number of write requests made against the volume.
write sizes (blks)	The write transfer size statistics.
write times (msec)	The write response time statistics.
write sequences	The number of write sequences. A sequence is a string of 512-byte blocks that are written consecutively.
write seq. lengths	Statistics describing the lengths of the write sequences, in blocks.
seeks	The number of seeks that preceded a read or write request; also expressed as a percentage of the total reads and writes that required seeks.
seek dist (blks)	Seek distance statistics, in units of 512-byte blocks. In addition to the usual statistics (avg/min/max/sdev), the distance of the initial seek operation (assuming block 0 was the starting position) is reported separately. This seek distance is sometimes very large; so, it is reported separately to avoid skewing the other statistics.
seek dist (cyls)	(Hard files only.) Seek distance statistics in units of disk cylinders.
time to next req	Statistics (avg/min/max/sdev) describing the length of time, in milliseconds, between consecutive read or write requests to the volume. This column indicates the rate at which the volume is being accessed.
throughput	The total volume throughput in kilobytes per second.

Column	Description
utilization	The fraction of time the volume was busy. The entries in this report are sorted by this field in decreasing order.

A long seek time may increase I/O response time and result in decreased application performance. In the example, the logical volume, fs2lv, has 142 writes, but there are only three write sequences and three seeks. This means that the logical volume is being accessed sequentially. The utilization is 24 percent, and average throughput is 1000.3 KBps during the measured interval. By examining the reads and read sequence counts, you can determine if the access is sequential or random. The same logic applies to the writes and write sequence.

3.7.5 Using filemon to estimate network bandwidth

`gmdsizing` can be used here to estimate the network bandwidth. What we are concerned with is the *most active logical volumes* statistics report and which logical volumes will be geographically mirrored. To estimate the necessary bandwidth for logical volume writes, we simply count the writes of logical volumes that are to be used as geographic mirror devices, and then we perform the calculation. Here is a simple example generated by the `filemon` command and measured in 30 second intervals.

```
...
Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume               description
------------------------------------------------------------------------
  0.75    24   6912  115.6  /dev/templv          /gmd1
  0.70  6897     64  116.1  /dev/test_lv         /mygmd
  0.02    88    144    3.5  /dev/hd4             /
  0.02   176    144    4.8  /dev/hd2             /usr
  0.00     0     48    0.7  /dev/hd8             jfslog
  0.00     0      8    0.1  /dev/loglv01         jfslog
  0.00     0      8    0.1  /dev/loglv00         jfslog
...
```

For example, if we want to estimate the necessary network bandwidth for the writes of /dev/templv, the formula will look like the following.

The necessary network bandwidth for /dev/templv is:

6912 / 2 x 8 = 3456 X 8 = 27648 Kbps

27648 / 30 = 921.6 Kbps

Since we need to allow for some network overhead for the network itself and also for GeoMessage, assume that we do not want our workload to exceed 75 percent of the bandwidth of the network. Thus, the following additional calculation is:

921.6 Kbps / 0.75 = 1228.8 Kbps = 1.2 Mbps

Therefore, the minimum necessary network bandwidth is 1.2 Mbps. In practical terms, we need at least a T1 (1.544 Mbps) as our HAGEO or GeoRM network. Note that there will be different network latencies between different networks. For example, T1's network latency factor is 0.8. This means the network latency will impact a T1's performance down to 1.544 Mbps x 0.8 = 1.2352 Mbps. If the estimated necessary network bandwidth exceeds T1, it will take a longer time to synchronize the geomirrored data to the remote site.

3.8 Estimating synchronization time

After discussing how to estimate the network bandwidth, we will now talk about how long it will take to synchronize geographic mirroring data using different media with the different bandwidth. Assume the overhead of the network is 20 percent; so, the efficient bandwidth is 80 percent.The following figure shows some estimates of the time required to synchronize 1 GB, 10 GB, 100 GB, and 1000 GB of HAGEO data using T1, E1, Ethernet, T3, FDDI, ATM OC3, and ATM OC12 media. For example, if you use T1 as a communication media, it is estimated to take 1799.08 hours to synchronize 1000 GB (1 terabyte) of geomirroring data to a remote site. This time represents about 75 days.

An example is 1GB of data using a T1 network:

1000 x 8 / (1.544 x 0.8) / 3600 = 1.80 hours (20 percent network overhead)

So, we get Table 20. You can refer to this table to evaluate how long it will take for synchronization. You can also calculate this yourself according to the rules.

Table 20. Time to sync geomirroring data using different media

Media type	Mbps	hrs/1 GB	hrs/10 GB	hrs/100 GB	hrs/1000 GB
T1	1.544	1.80	17.99	179.91	1799.08
E1	2	1.39	13.89	138.89	1388.89

Media type	Mbps	hrs/1 GB	hrs/10 GB	hrs/100 GB	hrs/1000 GB
10 MB Ethernet	10	0.28	2.78	27.78	277.78
100 MB Ethernet	100	0.03	0.28	2.78	27.78
GB Ethernet	1000	0.003	0.03	0.28	2.78
T3	44.736	0.06	0.62	6.21	62.09
FDDI	100	0.03	0.28	2.78	27.78
155 MB ATM	155	0.02	0.18	1.79	17.92
622 MB ATM	622	0.003	0.04	0.45	4.47

Better performance depends on better communication media. We do not recommend anything less than a T1 network for geographic mirroring.

Chapter 4. Installation and configuration

This chapter will describe how to install and configure GeoRM 2.1 and HAGEO 2.1 products. We will use AIX 4.3.3 for our examples.

4.1 Naming the GMD components

Before planning a configuration, we suggest that you define a clear naming convention for the GMDs and their logical volumes. This will make the administration of your environment easier in the future. Table 21 provides a description of the naming conventions that we use throughout this book.

Table 21. GMD naming conventions

Device description	Device name	LV type
Raw data LVs	<XXX><minor#>lv	gmd
GMD name for data LVs	<XXX><minor#>gmd	N/A
File systems	<XXX>	N/A
State Maps for data LVs	<XXX><minor#>sm	sm
Raw LV for jfslog	log<minor#>lv	gmdlog
GMD name for jfslog	log<minor#>gmd	N/A
State Maps for jfslog	log<minor#>sm	sm

<XXX> is the name of the mount point of the file system or logical volume (LV). Since the limit for the LV name is 15 characters, try to make this name nine characters long at most. Otherwise, you will not be able to follow the naming convention when creating the state maps.

<minor#> is the GMD device minor number and ranges from 0 to 1023.

Note

The LV type field is actually optional for the LVM. The logical volume types, gmd, gmdlog, and sm, are used for naming convention purposes only.

4.2 GeoRM setup general procedure

The following section introduces the steps required to install the GeoRM software and set up the configuration to mirror the data to a remote site.

As part of the planning, we strongly suggest that you fill out the GeoRM planning worksheets before starting to install the software and configure it. A

A sample GeoRM planning worksheet can be found in Appendix A of the *GeoRM Planning and Administration Guide, V1.1*, SC23-4308.

4.2.1 GeoRM installation and setup guideline

The following is the general procedure for planning, setting up, installing, and configuring a generic GeoRM configuration.

1. Plan sites, machines, disks, networks, and network interfaces. Fill in the GeoRM planning worksheets with all the definitions for the components that are going to be part of the configuration.
2. Check disk and create an LV. For both site machines, create VGs for data and one VG for the state map LV. Create a data LV (data##lv) and a corresponding state map LV (data##sm).
3. Install GeoRM networks and adapters.
4. Install the GeoRM software on all machines at each site.
5. Edit the /.rhosts file on all machines at each site
6. Configure GeoMessage on one machine. Synchronize GeoMessage definitions to all machines. In this step, you define the GeoRM machines, networks, and interfaces, and then synchronize the definitions to all machines.
7. Configure GeoManager on one machine. Synchronize GeoManager definitions to all machines. In this step, you define the GeoRM sites and associate GeoRM machines with each site, and then synchronize the definitions to all machines.
8. Start GeoMessage. GeoMessage must be started in order to configure the GeoMirror devices.
9. Configure GeoMirrors. Define GeoMirror devices for each machine individually. You must vary on the volume groups needed by each machine as you define those devices.
10. Customize AIX error notification for your configuration.
11. Verify and test the configuration.

4.2.2 GeoRM installation

4.2.2.1 Prerequisites for GeoRM

Before installing GeoRM, check the following prerequisites:

- Each machine is running at AIX Version 4.3.2, 4.3.3, or later.

- The `bos.data` module has been installed on each machine in case some modules of GeoRM cannot be installed, leaving you with an incomplete installation. To check whether the module is installed, type the command `lslpp -l | grep bos.data`. This should report information similar to the following:

```
bos.data    4.3.0.0COMMITTEDBase Operating System Data
```

- The /usr directory on each machine has 5 MB of free space.

4.2.2.2 Install GeoRM

To install the GeoRM software, perform the following steps:

1. Insert the tape or CD-ROM with the code into the drive and type `smit install_selectable_all`. SMIT displays the Install and Update from LATEST Available Software screen shown in Figure 25.I

```
           Install and Update from LATEST Available Software

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* INPUT device / directory for software                 [] +
```

Figure 25. Install and Update from LATEST Available Software screen

2. In the Entry Field, enter the name of the tape, CD-ROM, or directory path in case you have the code in a directory in a mounted file system.

   If you are not sure of the name of the input device, press F4 for a list of available devices. Select the proper device and press **Enter**. That value is entered into the *INPUT device/ directory for software* field as the valid input device.

3. Press **Enter**. SMIT will show the new screen with the device entered listed first as shown in Figure 26 on page 84.

```
            Install and Update from LATEST Available Software

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INPUT device / directory for software               /dev/cd0
* SOFTWARE to install                                [_all_latest]           +
  PREVIEW only? (install operation will NOT occur)    no                    +
  COMMIT software updates?                           yes                    +
  SAVE replaced files?                               no                     +
  AUTOMATICALLY install requisite software?          yes                    +
  EXTEND file systems if space needed?               yes                    +
  OVERWRITE same or newer versions?                  no                     +
  VERIFY install and check file sizes?               no                     +
  Include corresponding LANGUAGE filesets?           yes                    +
  DETAILED output?                                   no                     +
```

Figure 26. Install and Update from LATEST Available Software screen

4. Enter `geoRM.*` in the Software to Install field, or press **F4** to list the available software.

   If you press **F4**, a pop-up windows appears listing all installable software. Use the arrow keys to locate all software modules associated with the GeoRM software. Press **F7** to select the following modules:

   ```
   geoRM.man.en_US
   geoRM.manage
   geoRM.message
   geoRM.mirror
   ```

   Press **Enter** after making all selections. Your selections appear in this field.

5. Leave the entries for the other fields to the default value and press **Enter**. In the alert window, press **Enter** again to start the installation.

> **Note**
>
> At this time, the only language available for GeoRM is U.S. English.

4.2.3 Handling installation problems

If you have any problems during the install, the install program automatically performs a cleanup process. If, for some reason, the cleanup is not performed after an unsuccessful installation, from the SMIT Software Installation and Maintenance menu, select **Clean Up After a Failed Installation**.

Next, review the SMIT output or examine the /smit.log file to determine why the installation failed. Fix any problems and repeat the installation process.

4.2.3.1 Verifying the installation

If you wish, you can check the installation by performing the following steps:

1. Verify that the GeoMirror device driver and commands are installed.

   The GeoMirror device driver should be in the /etc/drivers directory. It is listed as gmd and gmdpin.

   The GeoMirror methods should be in the /usr/lib/methods directory. These include the following files: cfggmd, chggmd, defgmd, gmddown, startgmd, stopgmd, ucfggmd, and udefgmd.

   Verify that the /usr/sbin/gmd directory exists. The GeoMirror user commands should be listed. The `ls -l` command output is shown in Figure 27.

```
total 1008
-rwxr-xr-x   4 system       9406 Aug 24 18:23 add_geonode
-rwxr-xr-x   4 system       9966 Aug 24 18:23 add_geosite
-rwxr-xr-x   4 system       9406 Aug 24 18:23 change_geonode
-rwxr-xr-x   4 system       9966 Aug 24 18:23 change_geosite
-rwxr-xr-x   4 system       9406 Aug 24 18:23 delete_geonode
-rwxr-xr-x   4 system       9966 Aug 24 18:23 delete_geosite
-rwxr-xr-x   1 system     105210 Aug 24 18:23 geo_verify
-rwxr-xr-x   1 system      57670 Aug 24 18:23 gmd_show_state
-rwxr-xr-x   1 system      57054 Aug 24 18:23 gmd_update_state
-rwxr-xr-x   2 system       2547 Aug 24 18:23 gmdclean
-rwxr-xr-x   2 system       2547 Aug 24 18:23 gmddirty
-rwxr-xr-x   1 system       3235 Aug 24 18:23 gmdsizing
-rwxr-xr-x   1 system      44234 Aug 24 18:23 gmdstat
-rwxr-xr-x   1 system      49988 Aug 24 18:23 gmdswitch
-rwxr-xr-x   1 system       7592 Aug 24 18:23 pipt
-rwxr-xr-x   1 system      65916 Aug 24 18:22 rpc.geod
-rwxr-xr-x   4 system       9406 Aug 24 18:23 show_geonode
-rwxr-xr-x   4 system       9966 Aug 24 18:23 show_geosite
-rwxr-xr-x   1 system       2372 Aug 24 18:23 sync_geo_odm
```

Figure 27. Contents of the /usr/sbin/gmd directory

2. Verify that the /usr/sbin/krpc directory exists. It contains the GeoMessage files and commands. The `ls` command on this directory shows the following information:

```
cfgkrpc             krpc_add_network          krpc_list
initkrpc            krpc_change_interface     krpc_net_chng
krpc                krpc_change_machine       krpc_show_config
krpc_add_interface  krpc_change_network       krpcstat
krpc_add_machine    krpc_dist_config
```

3. When necessary, a line is added to the /etc/inetd.conf file to start the rpc.geod daemon. This daemon is used internally by some of the geo utilities.

```
rpc.geod    sunrpc_tcp tcp     wait    root    /usr/sbin/gmd/rpc.geod
rpc.geod 150005 1
```

4. The following lines are added to the /etc/services file:

```
gmd_port 6755/tcp
gmd_port 6755/udp
```

4.2.3.2 Enabling reception of GeoRM error messages

To receive full information on the GeoRM kernel processes, enable the following options by adding them to the /etc/syslog.conf file, and direct the log output to a file:

- **kern.info** - To receive general information messages
- **kern.err** - To receive error condition messages
- **kern.crit** - To receive critical condition messages, for example, hard error conditions

You can also choose to log only one type of message adding only the relative line. The format of the line to add is, for example

```
kern.info      /usr/syslog
```

So, all the general information messages will be logged in the /usr/syslog file.

4.3 HAGEO installation

The following sections cover the steps for installing the HAGEO software and setting up an HAGEO cluster.

4.3.1 Prerequisites for HAGEO

Before installing the HAGEO Version 2.1 software, check that the following prerequisites have been met:

- Each node is running AIX 4.3.2
- Each node has HACMP for AIX Version 4.3.0 installed
- Have the following PTFs available for HAGEO to apply after the installation:
  - hageo.mirror.utils.2.1.0.7 U462112
  - hageo.mirror.ext.2.1.0.7 U462111

- hageo.msg.en_US.message.2.1.0.4 U449602
- hageo.manage.utils.2.1.0.7 U462113
- hageo.message.ext.2.1.0.7 U462114
- hageo.message.utils.2.1.0.7 U462102
- hageo.msg.en_US.mirror.2.1.0.0 none
- hageo.man.en_US.message.data.2.1.0.0 none
- hageo.man.en_US.mirror.data.2.1.0.0 none

- The /usr directory on each node has 5 MB free space left for installation.

4.3.2 Check disks and memory

If you are installing and configuring HACMP for the first time, as part of the HAGEO installation and configuration process, include the disks required for HAGEO applications and mirrors. These should be documented in your planning sheets.

If you already have HACMP installed and configured at the two sites that will become the HAGEO cluster, you may need to add disks required for the HAGEO applications and mirrors at each site. You may have to add memory for the caching of the state maps (512 KB for each active state map). Since the write behinds are cached in memory, you may also have to add memory if you plan on running asynchronously.

4.3.3 Install the HAGEO networks and adapters

Perform the following steps to install the HAGEO networks and adapters. If you are using a bridge between your sites, the configuration of your geographic network adapters should be straightforward. However, if you are using a router, it is more complex. The following suggestions were arrived at through our testing:

- All geo networks and adapters must be set up in AIX prior to configuration into HAGEO.
- Use netmask 255.255.255.192 and address ranges that create four logical subnets across two sites on the nodes. Netmask the routers at 255.255.255.128 to divide the network into two parts.
- Define one router per site per geo network to talk to the site's service network.
- Define routes that enable the site-specific service interfaces to talk to each other through the routers.

- Verify that you can ping the remote site.
- The language must be set to en_US. The event scripts do not work correctly if the language environment is not set this way.

4.3.4 Install HAGEO Version 2.1

Enter `smitty install_selectable_all`. Fill in the installation device, and, in the field SOFTWARE to install, press **F4** and select all the HAGEO components.

After a successful installation, install all the available PTFs, and reboot the nodes. Upgrade current filesets of HAGEO to the recommended levels listed in Figure 28.

```
 Fileset                      Level  State Description
 ----------------------------------------------------------------------------
hageo.man.en_US.message.data
 2.1.0.0    C    HAGEO GeoMessage Man Pages -
                                          U.S. English
 hageo.man.en_US.mirror.data
 2.1.0.0    C    HAGEO GeoMirror Man Pages - U.S.
                                          English
 hageo.manage.utils         2.1.0.5    C    HAGEO GeoManage utilities
 hageo.message.ext          2.1.0.2    C    GEO Message Device Driver
 hageo.message.utils        2.1.0.5    C    HAGEO GeoMessage Utilities
 hageo.mirror.ext           2.1.0.5    C    HAGEO GeoMirror Device Driver
 hageo.mirror.utils         2.1.0.5    C    HAGEO GeoMirror Utilities
 hageo.msg.en_US.message    2.1.0.4    C    HAGEO GeoMessage Catalogs - U.S.
                                            English
 hageo.msg.en_US.mirror     2.1.0.0    C    HAGEO GeoMirror Message Catalogs
                                            - U.S. English
 State Codes:
  A -- Applied.
  B -- Broken.
  C -- Committed.
  O -- Obsolete.  (partially migrated to newer version)
  ? -- Inconsistent State...Run lppchk -v.
```

Figure 28. Recommended levels for HAGEO software

Make sure that the filesets, hageo.msg.en_US.message and hageo.msg.en_US.mirror, are installed; otherwise, some functions will not behave properly.

4.3.5 Verify HAGEO installation

- Verify that the GeoMirror device driver and commands are installed.The Geomirror device driver should be in the /etc/drivers directory. It is listed as gmd and gmdpin.

- The GeoMirror methods should be in the /usr/lib/methods directory. These include `cfggmd`, `chggmd`, `defgmd`, `gmddown`, `startgmd`, `stopgmd`, `ucfggmd`, and `udefgmd`.
- Verify that the /usr/sbin/gmd directory exists. The GeoMirror user commands should be listed. These include `geo_verify`, `gmdclean`, `gmddirty`, `gmdstat`, `gmdsizing`, `gmd_show_state`.
- Verify that the following GeoManager scripts and commands are installed in the /usr/sbin/gmd/scripts directory:
  - `Geo_local_peer_down`
  - `Geo_network_down`
  - `Geo_notify`
  - `Geo_shared_remote_dev`
  - `Geo_start_server`
  - `Geo_node_down`
  - `amiup`
  - `Geo_remote_peer_down`
  - `Geo_krpc_unload`
  - `configure_events`
  - `dbfs`
  - `Geo_remote_node_up`
  - `Geo_krpc_unload`
  - `Geo_mount_fs`
  - `geo_dbfs`
  - `Geo_prestart_gmds`
  - `Geo_stop_gmds`
  - `pre_event_handler`
  - `geo_ifchk`
  - `Geo_remote_join`
  - `geo_chknode`
  - `geolsnode`
  - `geo_info`
  - `geolssite`

- geo_drop_node
- geo_sh
- post_event_handler
- dbfs.envfile

- Verify that the following line is added to /etc/inetd.conf to start the rpc.geod daemon:

```
rpc.geod   sunrpc_tcp   tcp   wait   root   /usr/sbin/gmd/rpc.geod
    rpc.geod   150005   1
```

- The following lines are added to the /etc/services file:

```
gmd_port   6755/tcp (used for rpc.geod daemon)
gmd_port   6755/udp (used for GeoMessage)
```

- When you run smit and look under Communication Applications and Services, you should see an entry for HAGEO as shown in Figure 29.

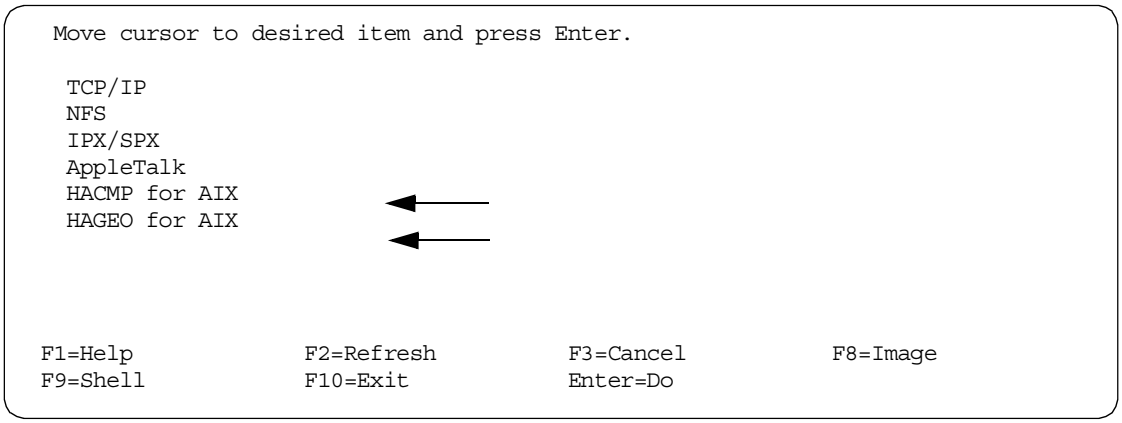

Figure 29. Communications applications and services

This means that you have successfully completed the HAGEO installation.

4.3.6 HAGEO setup guidelines

The diagram in Figure 30 on page 91 shows the various steps required to configure an HAGEO cluster. Much of the required information will come directly from your planning worksheets and the diagrams you created during the planning process.

The manual, *High Availability Geographic Cluster V2.1 for AIX: Planning and Administration Guide*, SC23-1886, gives detailed information on the installation and configuration process.

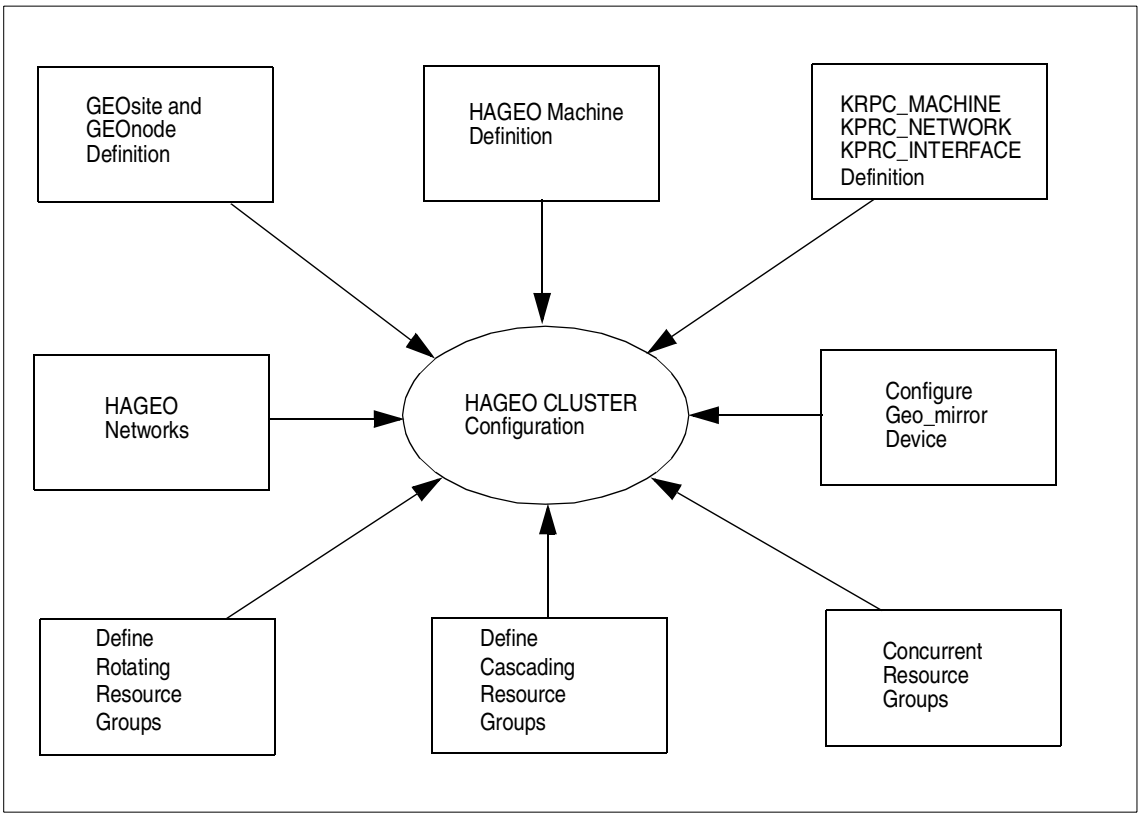

Figure 30. Configuring an HAGEO cluster

Chapter 5. Configuration and implementation examples

This chapter discusses the procedure for setting up GeoRM and HAGEO environments and gives some practical examples of procedures.

5.1 GeoRM configuration definitions and guidelines

A GeoRM configuration can have from two to eight machines (IBM RS/6000 systems). The machines must be distributed between two GeoRM sites. The machines in a site need not be in the same geographical location, although this is the most common configuration.

Machines defined as belonging to the same site may, if so desired, be connected to shared disks if they are actually located near enough to each other. This facilitates keeping the application running in case one machine fails.

Sites may be connected through a variety of local or wide-area TCP/IP networks. In order to protect against problems caused by network failures and possibly improve performance, it is wise to have at least two geographic networks between sites using different routes.

Each machine can have up to 128 GeoMirror devices (GMDs) available. A machine can use any type of GeoMirror (synchronous, asynchronous, or MWC) or a mixture of these types.

Applications reside on top of GeoRM and require no modifications.

The basic GeoRM configurations are as follows:

- GeoRM active-backup. Applications run at the active site and are mirrored to the backup site.
  - One variation of this configuration includes having one or more applications run on machines in disperse geographic locations; these are all configured as belonging to one active site. These applications are mirrored to one or more machines at the backup site.
  - A second variation of this configuration includes having an application running on two or more machines in the same location and sharing a disk to facilitate the handling of local machine failure or maintenance while keeping the application and geo-mirroring running.
- GeoRM mutual backup. Applications running on one site are mirrored at another.

All configurations provide a rapid remote backup method for critical data. The data at the backup site is guaranteed to be up to date with the data entered at the active site until the failure of an active site or one of its components interrupts the geo-mirroring process. Basically, machines at the backup site serve as storage devices.

For complete security, geo-mirrored data at the backup site should be mirrored locally, and the mirrored data can also be backed up and stored periodically on other media.

5.1.1 Two-machine active backup site configuration

To describe the GeoRM components configuration procedure, we use a two-machine GeoRM active-backup configuration. The local machine, called *Alpha*, is at the active site, and the remote machine, called *Beta,* is at the backup site.

Data entered at the site *S1* is sent over a geographic network and immediately mirrored on the disk at site *S2* as shown in Figure 31.

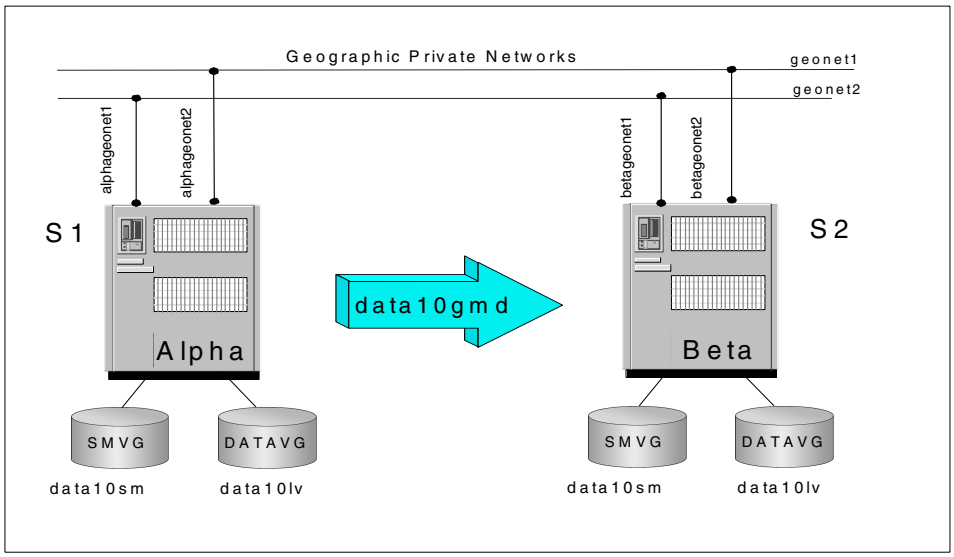

Figure 31. Active backup implementation example

As part of the planning, we have created two volume groups on both machines: One for the data logical volumes, called *datavg*, and another for the state maps, called *smvg.*

The network label settings on the two machine are shown in Table 22.

Table 22. Configuration labels

Machine name	Host name	Interfaces name
Alpha	Alpha	alphageonet1 alphageonet2
Beta	Beta	betageonet1 betageonet2

5.1.2 Configuring GeoMessage component

The first GeoRM component to configure is the GeoMessage. The configuration is done just on one machine and afterwards distributed to the others. To configure this component, it is necessary that all machines be interconnected and the /.rhosts file modified on all machines at each site in case the file does not exist and you have to create it.

We need to enter into this file the list of IP labels of all the GeoRM network interfaces. The GeoRM synchronization and verification functions use the *rsh* command and, thus, require these /.rhosts entries.

The `rsh` command allows you to execute a command on similar foreign hosts. The command requires that the host be authorized. This is accomplished by looking to the entries in the /.rhosts file on the remote machine. These files contain the IP labels and host names authorized to execute commands via rsh on the machine. The IP labels may be removed for security reasons after GeoRM configuration is complete.

Note

- The + character can be used to list the IP labels in the /.rhosts file. It enables any hosts to access the system using rsh.
- /.rhosts file is note required on SP systems running Kerberos.

5.1.2.1 Starting to configure GeoMessage

1. To configure the GeoMessage, we have to start the SMIT menu using the fastpath name for GeoRM: `smit georm`

   This will display the GeoRM for AIX menu shown in Figure 32 on page 96.

```
                                GeoRM for AIX

Move cursor to desired item and press Enter.

  Manage GeoMessage
  Manage GeoRM Sites/Machines
  Manage GeoMirror
  Verify GeoRM Configuration
  GeoRM Utilities
```

Figure 32. GeoRM for AIX main menu

2. Choose the **Manage GeoMessage** option. This displays the Manage GeoMessage menu shown in Figure 33.

```
                            Manage GeoMessage

      Move cursor to desired item and press
      Enter.

       Show GeoMessage Statistics
       Configure GeoMessage
       Start GeoMessage
```

Figure 33. Manage GeoMessage menu

3. Select **Configure GeoMessage**. The Configure GeoMessage menu shown in Figure 34 appears.

```
                        Configure GeoMessage

  Move cursor to desired item and press Enter.

    Show GeoMessage Configuration
   Distribute GeoMessage Definitions to Remote Machines
    Configure GeoMessage Machines
    Configure GeoMessage Networks
     Configure GeoMessage Interfaces
```

Figure 34. Configure GeoMessage

Now, we are in the Configure GeoMessage menu from which the configuration is started.

5.1.2.2 Configure GeoMessage machines

We create the machine definitions in the GeoMessage ODM.

1. From the Configure Message menu, select **Configure GeoMessage Machines**. The Configure GeoMessage machines menu shown in Figure 35 appears.

```
                        Configure GeoMessage machines

Move cursor to desired item and press Enter.

  Create new machine definition
  Change existing machine definition
  Delete existing machine definition
```

Figure 35. Configure GeoMessage machines menu

2. Choose **Create new machine definition**. The Create new machine menu shown in Figure 36 appears.

```
                        Create new machine definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Machine Name                                         [alpha]
* Promote Failure Timeout                              [120] #
```

Figure 36. Create new machine definition menu

3. Enter the values for the following fields, and press **Enter** to create the entry for the machine:

   Machine Name This is the name of the machine to configure. It must be the machine hostname. For example, `alpha` for an Alpha machine.

   Promote Failure Timeout The value here is the amount of time that it will take for a communication failure to be promoted to a *gmddown* condition. A gmddown condition informs a local GeoMirror device that a remote GeoMirror device is down. A value of zero disables the timeout.

4. Press **F3** and repeat steps 2 through 3 to also configure the Beta system.
5. Press **F3** to return to the Configure GeoMessage menu.

5.1.2.3 Configure GeoMessage networks

We have to create the GeoMessage ODM definition for the GeoMessage Networks.

1. From the Configure GeoMessage menu, choose **Configure GeoMessage Networks**. The Configure GeoMessage Networks menu appears as shown in Figure 37.

```
                        Configure GeoMessage Networks

 Move cursor to desired item and press Enter.

   Create new network definition
   Change existing network definition
   Delete existing network definition
```

Figure 37. Configure GeoMessage Networks menu

2. Choose the **Create new network definition** option. The Create new network definition menu shown in Figure 38 appears.

```
                        Create new network definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Network Name                                         [geonet1]
```

Figure 38. Create new network definition menu

3. Enter the network name in the field, and press **Enter** to create the network definition.
4. Press **F3** when the command ends the operation, and repeat steps 2 through 3 for any geographic network that you have planned.
5. Press **F3** to return to the Configure GeoMessage menu.

5.1.2.4 Configure GeoMessage interfaces

In this section, we will described how to add the definitions for the network interfaces to the GeoMessage ODM. Perform the following steps:

1. From the Configure GeoMessage menu, choose **Configure GeoMessage Interfaces**. The Configure GeoMessage Interfaces menu shown in Figure 39 on page 99 appears.

```
  Configure GeoMessage Interfaces

Move cursor to desired item and press Enter.

  Create new interface definition
  Change existing interface definition
  Delete existing interface definition
```

Figure 39. Configure GeoMessage Interfaces menu

2. Choose the **Create new interface definition**. The Create new interface definition menu shown in Figure 40 appears.

```
  Create new interface definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Machine Name                                       [alpha]+
* Network Name                                       [geonet1]+
* Interface Address                                  [9.3.187.184]
* Interface Name                                     [alphageonet1]
```

Figure 40. Create new interface definition menu

3. Insert the value for the fields and press **Enter** to create the interface definition.

   Machine Name The name of the GeoMessage machine. Pressing the F4 key shows you a list of machines to choose from.

   Network Name Name of the GeoMessage network before defined. Pressing the F4 key shows you a list of machines to choose from.

   Interface Address The IP address of the network adapter relative to the Machine Name and Network Name.

   Interface Name The name of the network adapter. The interface name must include the name of the machine to which it connects, such as alfa_geonet1.

4. Press **F3** and repeat Step 3 for any network interface for each machine.
5. Press **F3** to return to the Configuration GeoMessage menu.

5.1.2.5 Checking the GeoMessage configuration

At this point, we have completed the insertion of all the GeoRM configuration definitions. We can list them to check if any errors have been made.

From the Configure GeoMessage menu, choose the **Show GeoMessage Configuration** option. The Show GeoMessage Configuration menu shown in Figure 41 appears.

```
                Show GeoMessage Configuration

Move cursor to desired item and press Enter.

  Show entire configuration
  Show defined networks
  Show defined machines
  Show defined interfaces
```

Figure 41. Show GeoMessage Configuration menu

This menu allows reports on all configuration definitions or only one part. Choosing the `Show entire configuration` option creates a report similar to that in Figure 42.

```
  ---------------------------------------
  Network Name
  ---------------------------------------
  geonet1
  geonet2
  -----------------------------------------------
  Hostname / Promote Failure Timeout:
          Interface Name      Address
  -----------------------------------------------
  alpha / 120:
          alpha_geonet1        9.3.187.183
          alpha_geonet2        192.168.187.184
  beta / 120:
          beta_geonet2        192.168.187.186
          beta_geonet1        9.3.187.185
```

Figure 42. Show entire configuration report

If you look to the report and find any discrepancy between the listed definition and your plan worksheet, you can correct this by going to the Configure GeoMessage menu and choosing the relative submenu.

5.1.2.6 Distributing the GeoMessage configuration

Once the GeoMessage configuration is complete and correct, you have to distribute it to all the machines.

Make sure to update the /.rhosts file as described earlier in this chapter. Failing to do this will result in a permission-denied error from the rshd daemon during the synchronization.

Also, make sure to have the /etc/hosts file updated with all the IP labels for the network interfaces for any machines.

To start to synchronize the configuration between all machines, choose the **Distribute GeoMessage Definitions to Remote Machines** option from the Configure GeoMessage menu.

5.1.3 Configure GeoManager component

The GeoManager component of GeoRM include the GEOsite and GEOnode ODM object classes.

The configuration of both GeoRM sites and all relative GeoRM machines is done on one machines and after distributed and synchronized with the others, so the following procedure have to be executed only once for all machines.

5.1.3.1 Configuring the sites

1. Starting from the main GeoRM smitty menu, you can access it with the fastpath `smit georm`, choose the **Manage GeoRM Sites/Machines** option. The Manage GeoRM Sites/Machines menu appears as shown in Figure 43.

```
                  Manage GeoRM Sites/Machines

Move cursor to desired item and press Enter.

  Configure GeoRM Site
  Configure GeoRM Machine
  Show Sites/Machines
  Sync Site/Machine Definitions to all Machines
```

Figure 43. Manage GeoRM Sites/Machines menu

2. Choose the **Configure GeoRM Site** option. The Configure BeoRM Site menu appears as shown in Figure 44 on page 102.

```
                    Configure GeoRM Site

Move cursor to desired item and press Enter.

  Add GeoRM Site Definition
  Show/Change GeoRM Site Definition
  Remove GeoRM Site Definition
```

Figure 44. Configure GeoRM Site menu

3. Choose the **Add GeoRM Site Definition**. The Add GeoRM Site Definition screen appears as shown in Figure 45.

```
                  Add GeoRM Site Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Site Name                                          [S1]
* Site Machines                                     [alpha]
```

Figure 45. Add GeoRM Site Definition screen

4. To create the site definition for one site, enter the appropriate data in the Site Name and Site Machines fields and press **Enter**.
5. Perform Step 4 a second time.
6. Press **F3** to return to the Configure GeoRM Site menu.
7. Verify the definitions created by using the **Show/Change GeoRM Site Definition**. In the popup window, select the site in which you are interested. Here, you can choose to change the site name or add/remove a machine name.

5.1.3.2 Configure the GeoRM machines

Perform the following steps to configure the GeoRM machines:

1. From the Manage GeoRM Sites/Machines menu, choose the **Configure GeoRM Machines** option. The Configure GeoRM Machines menu shown in Figure 46 on page 103 appears.

```
                    Configure GeoRM Machine

Move cursor to desired item and press Enter.

  Add GeoRM Machine Definition
  Show/Change GeoRM Machine Definition
  Remove GeoRM Machine Definition
```

Figure 46. Configure GeoRM Machine menu

2. Select **Add GeoRM Machine Definition**. The Add GeoRM Machine Definition screen appears as shown in Figure 47.

```
                  Add GeoRM Machine Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Machine Name                                        [alpha]
* Site Name                                            [S1]
```

Figure 47. Add GeoRM Machine Definition screen

3. Here, you will enter the definition for any GeoRM machine for each GeoRM site. Repeat this step until all machines in both sites are inserted.

   Machine Name The name of the machine to define. This must be the same site machine name inserted during the site definition.

   Site Name The name of the site to which the machine belongs. This must be the same name of the site definition.

4. Press **F3** to return to the Configure GeoRM Machine menu.
5. To verify that the GeoRM machine definitions are inserted, choose the **Show/Change GeoRM Machine Definition** option. Choose the machine that you want to verify from the pop-up menu. Check that the machine and site names are correct, and, if they are not, change them.

5.1.3.3 Synchronize the definition on all GeoRM machines

After inserting the definition for the site and the machines on one machine, you have to distribute and synchronize the definition on the other machines by performing the following steps:

1. If you are not already there, return to the Manage GeoRM Sites/Machines menu.

2. Choose the **Sync Site/Machine Definitions to all Machines** and press **Enter**. This will update the ODMs of all the GeoRM machines with the Geonode and Geosite definitions inserted.

5.1.4 Configuring GeoMirror devices (GMDs)

The following sections cover the configuration of GeoMirror devices (GMDs).

5.1.4.1 Preliminary

There are some issues to consider before configuring the GeoMirror device:

- If you have not yet restarted the server after the software installation, now is the right time to do it.
- Start the GeoMessage on the machine on which you are configuring the GeoMirror devices. You can do this from the Manage GeoMessage menu by selecting the **Start Geomessage** option.
- Vary on the volume groups needed by each machine as you define those devices.

A GeoMirror device is a *logical device* that has a local and a remote component. You will configure the device on one machine at each site. On each machine, the name, logical volume, and various other device attributes are all the same.

The application whose data you want to mirror will use the GeoMirror device, which has a normal I/O device associated with it, and the mirror processes will be transparent to the application. On the machine where the application is going to run, the GeoMirror device will be defined as *primary.* Sometimes, it is called the *local* or *active* device*.* On the backup site, the remote peer GeoMirror device defined as *secondary* is receiving the data to mirror.

We need to configure any GeoMirror device individually on each GeoRM machine.

You should already have planned the GeoMirror device configuration and created the raw logical volumes to be used for data and state maps on each site.

5.1.4.2 Adding a GeoMirror device

The following procedure creates the GeoMirror device (GMD) where the application will write the data to mirror.

1. Starting from the GeoRM for AIX menu, choose the **Manage GeoMirror** option to enter the relative menu. The Manage GeoMirror menu appears as shown in Figure 48 on page 105.

```
                        Manage GeoMirror

Move cursor to desired item and press Enter.

  List All GeoMirror Devices
  Configure a GeoMirror Device
  GeoMirror Utilities
  Start a GeoMirror Device
  Stop a GeoMirror Device
```

Figure 48. Manage GeoMirror menu

2. Choose the **Configure a GeoMirror Device** option. The Configure a GeoMirror Device menu shown in Figure 49 appears.

```
                    Configure a GeoMirror Device

  Move cursor to desired item and press Enter.

    Add a GeoMirror Device
    Change/Show a GeoMirror Device
    Remove a GeoMirror Device
```

Figure 49. Configure a GeoMirror Device menu

3. Choose the **Add a GeoMirror Device** option and press **Enter**. SMIT will display the input screen shown in Figure 50 on page 106.

```
                    Add a GeoMirror Device

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
  Device Name                                         [data10gmd]
  Device State                                         stopped            +
* Minor Device Number                                 [10]                #
* State Map Logical Volume                            [/dev/rdata10sm]
* Local Logical Volume                                [/dev/rdata10lv]
* Device Mode                                          sync               +
* Device Role                                          primary            +
  High Water Mark                                     []                    #
  Default Sync Concurrency Rate                       []                   +#
  Remote Machine and Logical Volume                    [beta@/dev/rdata10lv]
  Remote Machine and Logical Volume                   []
  Remote Machine and Logical Volume                   []
  Remote Machine and Logical Volume                   []
  Remote Machine and Logical Volume                   []
  Remote Machine and Logical Volume                   []
  Remote Machine and Logical Volume                   []
```

Figure 50. Add a GeoMirror Device screen

4. Insert the information for each GeoMirror device that needs to be configured.

   Device Name Enter the name of the local GeoMirror device. The same name need to be used on the remote machine. The application wrote the data to geographically mirror on this device.For example `data10gmd`

   Device State The current state of the device. The device should be created in the STOPPED state.

   Minor Device Number This is the GeoMirror minor device number. This must be the same on the remote and local sides for the GeoMirror device. The value range is 0-1023

   State Map LV The name of the raw device relative to the state map logical volume for this GeoMirror device, for example, `/dev/rdata10sm`.

   Local LV The name of the raw device relative to the logical volume to use for the data I/O, for example, `/dev/rdata10lv`.

   Device Mode The mirror mode for the GeoMirror device: *sync*, *async*, or *mwc.* For this example, we use sync.

Device Role It's the role of the device, *primary* on the sending side, *secondary* on the receiving side

High Water Mark Used only on the primary side by the async device. The value indicate the number of 1KB blocks of data should be in pending state on the primary side before transfer them on the secondary side. Default is 128

Sync Concurrency Rate The data blocksize in KB for the sync operation when a side is reintegrated. Valid entries: 32 or 64. Default is 32

Remote Machine and LV The remote peer machine name and remote raw device name in the format <hostname>@<raw device name>, for example `beta@/dev/rdata10lv`. Enter the information for each remote peer for this device.

5. When you have inserted all the information, press **Enter**.
6. Repeat Step 4 for any GeoMirror device in this machine.

5.1.4.3 Checking and changing the GeoMirror devices

To check the GeoMirror devices created on all machines, from the Manage GeoMirror menu, select the **List All GeoMirror Devices** option. The output of the command will list the name, status, location, and description of all the GeoMirror devices defined as shown in Figure 51.

```
                        COMMAND STATUS

Command: OK             stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

name    status  location description

data10gmd Stopped          Geographic Mirror Device
```

Figure 51. Command status screen

To check or change the parameters of a GeoMirror device, from the Configure a GeoMirror Device menu, choose the **Change/Show a GeoMirror Device** option. Select the GeoMirror device name from the pop-up list, and SMIT will display the Change/Show a GeoMirror Device screen. Check the parameters and, if necessary, change them.

5.1.5 Verifying the GeoRM configuration

At this point, we have completed the base portion of the GeoRM configuration. Once we have configured all the components of GeoRm, we have to verify that the definitions are correct and consistent across the machines.

To perform this verification, we will run the geo_verify utility. From the GeoRM for AIX smitty menu, select **Verify GeoRM Configuration**.

From the Verify GeoRM Configuration window that appears, you can choose to verify only some GeoMirror devices and some peer machines. The standard value will verify the devices and all components on each peer machine. The report is written to an output file so that you can print or send it.

Look for any warning messages or error messages at the end of the report. If there are any, take corrective action to resolve them.

5.1.6 Starting the GeoRM configuration

After the GeoRM configuration is complete, we have to start it to allow the application to write on the GeoMirror device.

Be sure to vary on the volume group for the GeoMirror device on both machines.

On the command line of the alpha machine, type the following command:

```
# /usr/sbin/methods/gmddown -l data10gmd beta
```

The `gmddown` command informs the local machine (alpha in our example) that the remote peer (beta in our example) is down. You need to do this if you have already started or stopped the GMD once. For more information about starting a GMD, refer to Chapter 6, “Administration” on page 173.

From the GeoRM smitty men, select **Manage GeoMirror->Start a GeoMirror Device**.

Select the GeoMirror device name from the Select Device Name window and press **Enter**. The Start a GeoMirror Device screen appears as shown in Figure 52 on page 109.

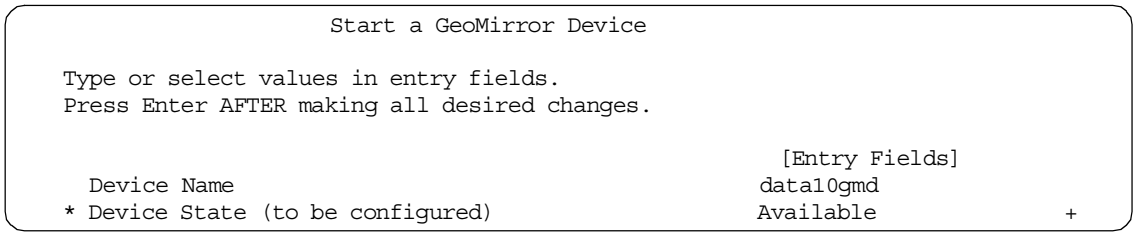

Figure 52. Start a GeoMirror Device screen

Press the **Tab** key to change the Device State value to Available, and press **Enter**.

When the command finishes running, it will report the state of the GMD as Available. We can now start our application on alpha, and the data written on the GMD device will be mirrored on the beta machine.

To start a GMD device you can also use the command line. There are two sets of commands available:

```
# mkdev -l gmd_name
```

OR

```
# /usr/lib/methods/startgmd -l gmd_name
```

where `gmd_name` is the name of the GMD device to start.

If you have more GMD devices, you can start them all using the following command:

```
# /usr/lib/methods/startgmd -A
```

Note

Trying to start a GMD the following error is reported:

```
Method error (/usr/lib/methods/startgmd):
      0514-046 A file containing microcode or adapter software was
               not accessable.
startgmd: Can't start data_10, remote peer is not in available state.
```

because the gmddown command haven't been execute. Run the gmddown command and repeat the operation.

5.1.7 Defining a file system on a GeoMirror device

To mirror a file system, we need to mirror the file system data logical volume and the log jfs logical volume. Unlike the raw logical volume, where we mirror only the data logical volume using a GeoMirror device, with a Journaled File System (JFS), we also need to mirror the JFS log logical volume.

The JFS log logical volume for the GeoMirror device must always be created. A volume group can contain both mirrored and non-mirrored JFS logical volumes. If the GeoMirror device logical volumes are added to an existing volume group, a second JFS log logical volume must be created for the mirrored JFSs. You cannot use the existing volume group JFS log for the non-mirrored data.

For our configuration, we are going to create the device structure shown in Figure 53. To mirror the file system/data, we need to define the two GMDs called data11gmd and log12gmd on both our local machines, Alpha and the remote machine Beta.

- The GMD data11gmd will mirror the file system logical volume called /dev/data11lv using the state map logical volume called /dev/data11sm.
- The GMD log1gmd will mirror the log file system logical volume called /dev/log12lv using the state map logical volume called /dev/log12sm.

  Note that we are using two different minor numbers for the two GMDs.

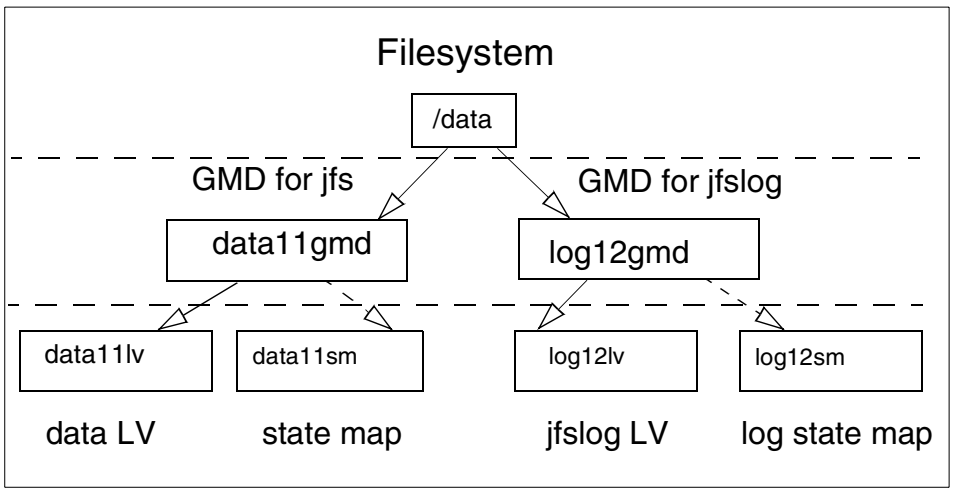

Figure 53. Description example names file system

Perform the following steps to execute to the local and remote machines:

1. Create the logical volumes for the GeoMirror data (/dev/data11lv and /dev/log12lv) in the datavg volume group. You can use the `smitty lv` or `mkdev` commands. You need to format the log logical volume as jfs log:

   ```
   # logform /dev/log12lv.
   ```

2. Create the logical volume for the state map for the two GeoMirror devices, /dev/data11sm and /dev/log12sm.

3. Configure the GeoMirror devices, data11gmd and log12gmd.

   Enter the `smitty gmddev` command to run the Manage GeoMirror smitty menu. Select **Configure a GeoMirror Device->Add a GeoMirror Device**. The screen shown in Figure 50 on page 106 appears.

   Enter the following information in the appropriate fields as indicated, first for data11gmd and then for log12gmd:

   a. **Device Name** - `data11gmd` or `log12gmd`.

   b. **Device State** - Both GMDs must be created in a `stopped` state.

   c. **Minor Device Number** - Insert `11` for data11gmd, insert `12` for log12gmd.

   d. **State Map LV** - Enter `/dev/rdata11sm` for data11gmd and `/dev/rlo12sm` for log12gmd.

   e. **Local LV** - Enter `/dev/rdata11lv` for data11gmd and `/dev/rlog12lv` for log12gmd.

   f. **Device Mode** - For this example, we are using the `sync` mode.

   g. **Device Role** - Enter `primary` on the Alpha machine and `secondary` on the Beta machine; this is valid for both GMDs.

   h. **High Water Mark** - Leave blank to use the default 128.

   i. **Default Sync Concurrent Rate** - Leave blank to use the default 32.

   j. **Remote Machine and Logical Volume** - Enter `<machinename>@/dev/rdata11lv` for data11gmd and `<machinename>@/dev/rlog12lv` for log12gmd, where `<machinename>` is `beta` when you configure the Alpha machine and `alfa` when you configure the Beta machine. See Figure 54 on page 112.

```
                        Add a GeoMirror Device

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
  Device Name                                         [data11gmd]
  Device State                                         stopped                 +
* Minor Device Number                                 [11]                     #
* State Map Logical Volume                            [/dev/rdata11sm]
* Local Logical Volume                                [/dev/rdata11lv]
* Device Mode                                          sync                    +
* Device Role                                          primary                 +
  High Water Mark                                     []                       #
  Default Sync Concurrency Rate                       []                      +#
  Remote Machine and Logical Volume                   [beta@/dev/rdata11lv]
  Remote Machine and Logical Volume                    []
  Remote Machine and Logical Volume                    []
  Remote Machine and Logical Volume                    []
  Remote Machine and Logical Volume                    []
```

Figure 54. Add dat11gmd GeoMirror Device on Alpha

4. After entering the information for each GMD on both machines, press **F3** to exit the menu.
5. Edit the /etc/filesystems file and add an entry for the GeoMirror device file system (/data in our configuration) and set the log parameter to point to the standard log logical volume with the stanzas shown in the following screen:

```
/data:
       dev     = /dev/data11lv
       vfs     = jfs
       log     = /dev/loglv00
       mount   = false
       check   = false
       options = rw
       account = false
```

6. Change the file system JFS log logical volume information to point to the file system log GeoMirror device (log12gmd in our configuration):

   ```
   # chfs -a log="/dev/log12gmd" /data
   ```

7. Change the file systems entry in the /etc/filesystems to point to the GMD.

   In the stanzas for the file system, /data11, change:

   ```
   # dev = /dev/data11lv
   ```

to:

```
# dev = /dev/data11gmd
```

8. From the Manage GeoMirror menu, select **Start a GeoMirror Device** to start these GeoMirror devices, or use the `smitty gmddev` fastpath.
9. Make the directory `data` a mount point:

   ```
   # mkdir data
   ```

10. On the local machine, Alpha, make the file system /data on the GeoMirror Device data11gmd:

    ```
    # mkfs -l data /dev/data11gmd
    ```

11. Mount the file system, /data, on the local machine

At this point, the device should be working and mirrors the I/O operation execute on the file system on the local machine to the remote site.

5.1.8 GeoRM complex scenario example

For our example, we implement a configuration where the two machines in the local site have an independent file system, each mirrored over a geographic network to a machine in a remote site. This can be useful not only for keeping data in a different place far from the production site, but also for allowing the periodic backup of data without affecting the availability of the service. This can be done simply by stopping the remote GeoMirror device (GMD) and executing the backup of the data on the remote site. In the mean time, in the local site, the data will still be available, and the local GeoMirror device will keep track of the modification. When the backup operation is completed and the remote GMD restarted, only the modified data will be copied over to the remote site.

In this configuration, it is possible to implement and show two different scenarios of failure and the procedures to recover from them:

- A local machine failure with takeover at the same site
- A local machine failure with takeover at the remote site

We suppose that the two local machines where the production applications run, Alpha and Beta, are sharing one or more external disk enclosures. This can be in the same room or in a different room or building. The VG(s) with the critical data to mirror are on these external enclosures as the state map VG(s). The configuration is shown in Figure 55 on page 114.

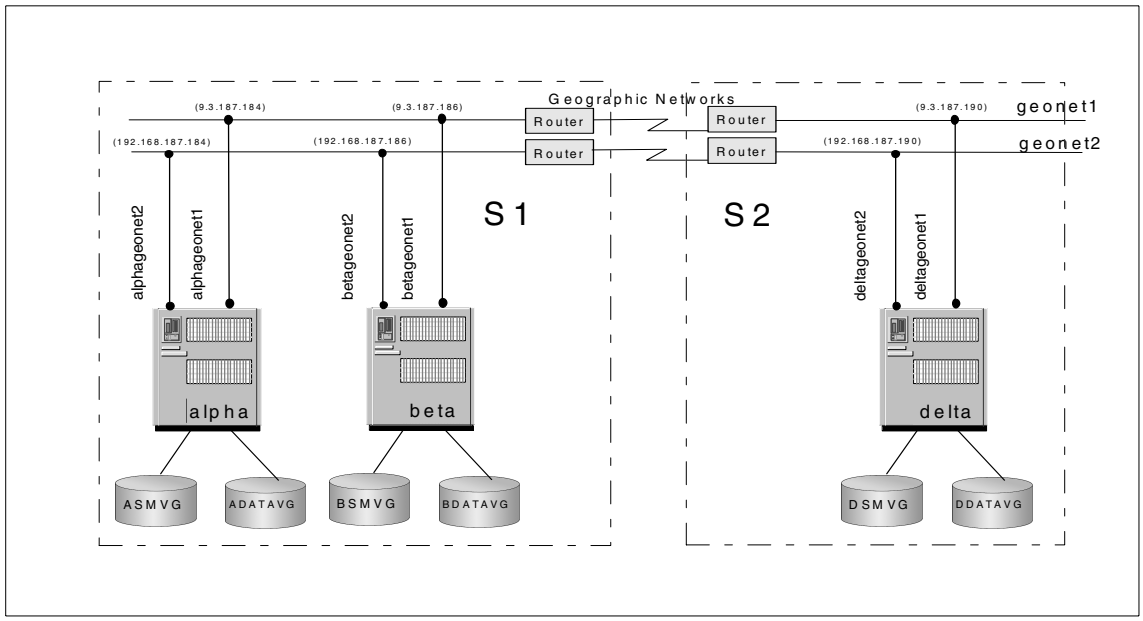

Figure 55. Configuration

We are going to create the device shown in Table 23 on the two local machines and on the remote machine. We call the file systems /alpha for the mirrored file system on Alpha and /beta for the mirrored file system on Beta.

Table 23. GMD and LV dependency for machines

Machine and fs	GMD	LVs	VG
Alpha#/alpha Delta	alpha20gmd	alpha20lv	adatavg
		alpha20sm	asmvg
	log21gmd	log21lv	adatavg
		log21sm	asmvg
Beta#/beta Delta	beta30gmd	beta30lv	bdatavg
		beta30sm	bsmvg
	log31gmd	log31lv	bdatavg
		log31sm	bsmvg

5.1.8.1 Installation and setup of the GeoRM configuration

This section describes the procedure for configuring the machine on the two sites.

On the Alpha machine:

1. Create the two VGs on each of the three machines, one for the data LV and the other for the state map LV.
2. Create the LVs on Alpha and Beta, and then create the associated LVs on the Delta machine. See Table 23 on page 114 for the LV names used. Remember to format the log LV as jfs log. For instance, to format the log21lv LV, enter:

   ```
   logform /dev/rlog21lv
   ```

3. If it is not already done, install the network adapters for GeoRM and configure them; we will use two network adapters in our configuration. Set the IP label for each adapter and check that it is correct in the /etc/hosts file. Remember that the IP labels are used to configure the GeoMessage components. For instance, the /etc/hosts file should appear for the Alpha machine as shown in Figure 56.

```
127.0.0.1              loopback localhost alpha
# Geonet1
9.3.187.184      alpha_geonet1 alpha
9.3.187.186      beta_geonet1 beta
9.3.187.190     delta_geonet1 delta
# Geonet2
192.168.187.184 alpha_geonet2
192.168.187.186 beta_geonet2
192.168.187.190 delta_geonet2
```

Figure 56. The /etc/hosts file

4. If it has not already been done, install GeoRM on all the machines.
5. Edit the /.rhosts file in each machine to add the IP labels. The following lines need to be inserted:

   ```
   alpha_geonet1
   alpha_geonet2
   beta_geonet1
   beta_geonet2
   delta_geonet1
   delta_geonet2
   alpha
   beta
   delta
   ```

6. Configure GeoMessage on one machine, and synchronize the GeoMessage definition to all machines.
7. Configure the GeoManager on one machine, and synchronize the GeoManager definition to all machines.
8. Start GeoMessage on each machine.
9. Configure the GeoMirrors devices on each machine individually. Remember to use the raw device names for the LVs, for instance, /dev/ralpha20lv to refer to the alpha20lv LV.
10. Edit the /etc/filesystems on the local machine, and add a new stanza for the file system that we want to mirror. For instance, on Alpha, the stanzas should appear as shown in Figure 57.

```
/alpha:
      dev     = /dev/alpha20lv
      vfs     = jfs
      log     = /dev/loglv00
      mount   = false
      check   = false
      options = rw
      account = false
```

Figure 57. File system stanzas

11. Change the log LV info to point to the filesytem log GeoMirror device. For instance, on Alpha, the command is:

    ```
    # chfs -a log="/dev/log21gmd" /alpha
    ```

12. Change the file system entry in the /etc/filesystems file to point to the file system GMD. For instance, on Alpha, change `/dev/alpha20lv` to `/dev/alpha20gmd`.
13. Complete steps 10 through 12 on Beta using the relative definitions for LVs and GMD where necessary.
14. On the local machines, Alpha and Beta, run the `gmddown` command to inform the local GMD that the remote GMD on Delta is down.
    - On Alpha:

      ```
      # /usr/lib/methods/gmddown -l alpha20gmd delta
      ```

    - On Beta:

      ```
      # /usr/lib/methods/gmddown -l beta30gmd delta
      ```

    This operation is not necessary if you are sure you have never started the GMDs before.

15. Start the GMDs on Alpha and Beta using the option from the smitty gmddev or executing the `/dev/lib/methods/startgmd -l <gmdname>` command where `<gmdname>` is alpha20gmd or beta20gmd.

> **Note**
>
> The GMD device must be in Stopped state. If it is in Defined state, run the `mkdev -l <gmdname> -S` command to change the device state in Stopped.

16. Start the peer GMDs for Alpha and Beta on Delta. You can start all the Stopped GMDs with the `/usr/lib/methods/startgmd -A` command.
17. Create the directory mounting point on the local machines.
    - On Alpha:

      ```
      # mkdir alpha
      ```

    - On Beta:

      ```
      # mkdir beta
      ```

18. On the local machines, create the file systems on the GMD devices.
    - On Alpha:

      ```
      # mkfs -l alpha /dev/alpha20gmd
      ```

    - On Beta:

      ```
      # mkfs -l beta /dev/beta30gmd
      ```

19. Mount the file systems, /alpha and /beta, on the local machines.

Now, the two file systems on Alpha and Beta are mirrored on Delta.

5.1.9 Local machine failure with takeover at the same site

This section describes the procedure to allow a local takeover at the same site. If a local machine failure occurs, the other local machine can be configured to take over the role of the failed machine.

For example, we show the procedure to allow Beta to take over the file system /alpha in case Alpha is down.

5.1.9.1 Takeover procedure

Perform the following steps to execute a takeover.

1. Stop the GMD(s) on the remote machine to prevent any data diversion and move them to the Defined state.

On Delta:

```
# rmdev -l alpha20gmd
# rmdev -l log21gmd
```

2. Change the remote peer definition for the GMD(s) on the remote machine from alpha@/<rawLVname> to beta@/<rawLVname> where <rawLVname> is the name of the raw logical volume for the GMD device.

   On Delta:

   ```
   # chdev -l alpha20gmd -a remote_peer=beta@/ralpha20lv
   # chdev -l log21gmd -a remote_peer=beta@/rlog21lv
   ```

3. If the Alpha application is not already present on Beta, install it.
4. Create the GMD relative to the /alpha file system on Beta. The definitions are the same ones used during the configuration of GeoRM on Alpha. The GMD(s) to create are alpha20gmd and log21gmd in Stopped state.
5. Add the stanzas in the /etc/filesystems file for the /alpha file system and configure the log logical volume and data logical volume by following steps 10 through 12 in Section 5.1.8.1, “Installation and setup of the GeoRM configuration” on page 115.
6. Import and vary on the Alpha’s VG(s).
7. Start the GMD(s) for /alpha on Beta and then on Delta:

   ```
   mkdev -l alpha20gmd
   mkdev -l log21gmd
   ```

8. Create the mounting directory, and mount /alpha on Beta.
9. Start the Alpha application.

To hasten the takeover procedure, steps 3 through 5 can be planned during the configuration of the local machines in order to have each machine ready to take over the other in case of failure.

5.1.9.2 Restore procedure

At this point, Beta has the same role as Alpha and can perform its service until Alpha is recovered. To restore Alpha, you will need to perform the following steps:

1. Stop the GMD(s) on the remote machine Delta to prevent any data diversion, and move them into the Defined state.
2. Change back the remote definition for the GMD(s) on Delta:

   ```
   # chdev -l alpha20gmd -a remote_peer=alpha@/ralpha20lv
   # chdev -l log21gmd -a remote_peer=alpha@/rlog21lv
   ```

3. Unmount /alpha and stop the GMD(s) on Beta. Set the GMD(s) in Defined state.
4. Vary off and export on Beta the VG(s) with the data and the state maps LV(s) of the /alpha file system, and then import and vary them on Alpha.
5. Start the GMD(s) for the /alpha file system and mount it.
6. Start the application on Alpha.

5.1.10 Local machine failure with takeover at the remote site

This section covers the steps required to take over local machine function on the remote site machine and what to do when the local machine returns to service. This is a consequence of a site disaster where there are no other local machines available.

If it has not been done already, set up the file system definitions for both machines on Delta as follows:

1. Create the stanzas for the two file systems in the /etc/filesystems file using the LV device name for the `dev` parameter. For instance, for the /alpha file system, enter

   ```
   dev = /dev/alpha20lv
   ```

2. Set the definitions of the log logical volume to the log GMDs

   ```
   chfs -a log=/dev/log21gmd /alpha
   chfs -a log=/dev/log31gmd /beta
   ```

3. In the /etc/filesystems file, change the `dev` definition of the file systems to the GMD name. For example, `dev=/dev/beta30lv` was changed to `dev=/dev/beta30gmd`

Finally, in the /etc/filesystems, you should have the definitions listed in Figure 58.

```
/alpha:
        dev             = /dev/alpha20gmd
        vfs             = jfs
        log             = /dev/log21gmd
        mount           = false
        check           = false
        option          = rw
        account         = false
/beta:
        dev             = /dev/beta30gmd
        vfs             = jfs
        log             = /dev/log31gmd
        mount           = false
        check           = false
        option          = rw
        account         = false
```

Figure 58. Definitions

Follow the procedure to configure the GMDs and mount the file system on the Delta for the failing local machine, or repeat the steps for both file systems if both local machines are unavailable.

5.1.10.1 Take-over procedure

Imagine that Beta is down and we need to mount the /beta file system using the mirror copy on Delta. Execute the following steps on Delta:

1. Stop the file system GMDs using the following commands:

   ```
   /usr/lib/methods/stopgmd -l beta30gmd
   /usr/lib/methods/stopgmd -l log31gmd
   ```

2. Remove and set the GMDs in Define state. You can use the following commands:

   ```
   rmdev -l beta30gmd
   rmdev -l log31gmd
   ```

3. Change the role for the GMDs from Secondary to Primary. In this way, it's possible to write on the GMD on this machine:

   ```
   chdev -l beta30gmd -a device_role=primary
   chdev -l log31gmd -a device_role=primary
   ```

4. Configure the GMDs and set them in Stopped state:

   ```
   /usr/lib/methods/cfggmd -l beta30gmd
   /usr/lib/methods/cfggmd -l log31gmd
   ```

5. Mark the remote GMDs as down:

```
/usr/lib/methods/gmddown -l beta30gmd beta
/usr/lib/methods/gmddown -l log31gmd beta
```

6. Start the GMDs:

```
/usr/lib/methods/startgmd -l beta30gmd
/usr/lib/methods/startgmd -l log31gmd
```

7. Create a mounting point and mount the file system:

```
mkdir /beta
mount /beta
```

8. Start the relative application to restore the Beta services.

5.1.10.2 Restore procedure

When the local machine Beta is returning in service, you need to restore the data and the mirror. There are two procedures you can use.

Procedure 1

This procedure is changing the role of the GMD on Beta to Secondary; so, the Delta machine will copy all the modified data over, syncronizing the two copies. Then, the GMD's roles are swapped between Beta and Delta to restore the normal configuration and mount the file systems on Beta.

On the local machine, Beta:

1. Set the GMDs in Defined state:

```
# rmdev -l beta30gmd
# rmdev -l log31gmd
```

2. Change the role for the GMD devices to Secondary:

```
# chdev -l beta39gmd -a device_role=secondary
# chdev -l log31gmd -a device_role=secondary
```

3. Configure and set the GMDs in Stopped state:

```
# mkdev -l beta30gmd -S
# mkdev -l log31gmd -S
```

4. Start the GMDs:

```
# mkdev -l beta30gmd
# mkdev -l log31gmd
```

5. When the system reports the GMD as Available, data synchronization between the two mirrors is complete; so, Beta is now the latest data.

   To mount the file system on Beta, we need to assign the Primary role back to the local GMDs.

On the remote site, Delta, we have to perform the following steps:

1. Stop the application that uses the /beta file system
2. Unmount the file system:

```
# umount /beta
```

3. Stop the GMDs and unconfigure them; set the GMDs to the Defined state:

```
# rmdev -l beta30gmd
# rmdev -l log31gmd
```

4. Change GMDs role to Secondary:

```
# chdev -l beta30gmd -a device_role=secondary
# chdev -l log31gmd -a device_role=secondary
```

On the local machine, Beta, perform the following steps:

1. Stop the GMDs and unconfigure them; set the GMDs to the Defined state:

```
# rmdev -l beta30gmd
# rmdev -l log31gmd
```

2. Change the GMD's role to Primary:

```
# chdev -l beta30gmd -a device_role=primary
# chdev -l log31gmd -a device_role=primary
```

3. Configure and set the GMDs in Stopped state:

```
# mkdev -l beta30gmd -S
# mkdev -l log31gmd -S
```

4. Execute the `gmddown` command to inform the system that the remote GMDs are down:

```
# /usr/lib/methods/gmddown -l beta30gmd delta
# /usr/lib/methods/gmddown -l log31gmd delta
```

5. Start the GMDs:

```
# mkdev -l beta30gmd
# mkdev -l log31gmd
```

6. Mount the file systems to make the /beta file system available on the local site again:

```
# mount /beta
```

7. On Delta, we have to restart the remote mirror devices to mirror the I/O operation on the file system in the local site. Start the devices:

```
# mkdev -l beta30gmd
# mkdev -l log31gmd
```

Procedure 2

This procedure keeps the Primary role for the GMD on Beta and changes back to Secondary on Delta. To synchronize the data, Delta's GMD needs to start before the Beta's GMD. In this way, the modified data will be copied from Delta to Beta.

1. When Beta is again up and running, check that the GMDs are in Defined state:

```
# /usr/sbin/gmd/gmd_show_state -l beta30gmd
# /usr/sbin/gmd/gmd_show_state -l log31gmd
```

On Delta:

2. Stop the applications and unmount the filsystem /beta.
3. Move the GMD(s) into the Defined state.
4. Change the role for GMD(s) from Primary to Secondary:

```
# chdev -l beta30gmd -a device_role=secondary
# chdev -l log31gmd -a device_role=secondary
```

5. Move the GMD(s) in Stopped state:

```
# mkdev -l beta30gmd -S
# mkdev -l log31gmd -S
```

6. Inform the GMD(s) that the remote peer is down:

```
# /usr/lib/methods/gmddown -l beta30gmd beta
# /usr/lib/methods/gmddown -l log31gmd beta
```

7. Start the GMD(s):

```
# mkdev -l beta30gmd
# mkdev -l log31gmd
```

On Beta

8. Start the GMD(s):

```
# mkdev -l beta30gmd
# mkdev -l log31gmd
```

9. Mount the /beta file system and start the application to restore the services

5.2 Dial Back Fail Safe (DBFS)

DBFS is a facility that is used by the GeoManager to distinguish between site isolation and a site disaster when all the primary geographical networks become unavailable. It uses a telephone line and modems to establish a

connection between sites. The non-dominant site always contacts the dominant site, never the other way around.

5.2.1 Steps to Configure DBFS

Perform the following steps to configure DBFS:

1. Configure TTY:

   Connect the modem to a serial port in the RS/6000 using the appropriate straight RS232C cable. Then, configure the tty interface using the `smit mktty` command. The screen in Figure 59 will appear.

```
                                Add a TTY

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                    [Entry Fields]
  TTY type                                          tty
  TTY interface                                     rs232
  Description                                       Asynchronous Terminal
  Parent adapter                                    sa0
* PORT number                                       []                  +
  Enable LOGIN                                      enable              +
  BAUD rate                                         [9600]              +
  PARITY                                            [none]              +
  BITS per character                                [8]                 +
  Number of STOP BITS                               [1]                 +
  TIME before advancing to next port setting        [0]                 +#
  TERMINAL type                                     [dumb]
  FLOW CONTROL to be used                           [xon] +
[MORE...29]

F1=Help             F2=Refresh          F3=Cancel       F4=List
F5=Reset            F6=Command          F7=Edit         F8=Image
```

Figure 59. Adding a TTY for DBFS

   In the Enable LOGIN field, select **enable**, if it is a node in the dominant site, or **disable**, if it is a node in the non-dominant site.

2. Install the bos.net.rte fileset:

   DBFS invokes the `/usr/bin/ate` command; therefore, you need to install this fileset prior to using DBFS.

3. Create a non-root user:

For security reasons, we recommend that you create a non-root user to be used by DBFS. For instance, you can create a user, named DBMS, with the following characteristics:

- Login name dbfs
- Password secretpw
- GECOS DBFS facility login user
- home directory /home/dbfs
- login shell /usr/bin/ksh

After assigning the new password to the user you created, do not forget to log in as the user you created to validate the new password.

4. Test the modem connection:

   To verify that the configuration of the modems is correct, you can use the `/usr/bin/ate` command. First, log in as root on one of the nodes and execute the `ate` command without options. The screen in Figure 60 appears.

```
Node: minnie          UNCONNECTED MAIN MENU
------------------------------------------------------------
     COMMAND      DESCRIPTION
     -------      ----------------------------------------
     Connect      Make a connection
     Directory    Display a dialing directory.

     Help         Get help and instructions.
     Modify       Modify local settings.
     Alter        Alter connection settings.
     Perform      Perform an Operating System command.
     Quit         Quit the program.
------------------------------------------------------------
     The following keys can be used during a connection:
         ctrl b Start or stop recording display output.
         ctrl v Display main menu to issue a command.
         ctrl r Return to a previous screen at any time.
------------------------------------------------------------
Type the first letter of the command and press Enter.
>
```

Figure 60. ATE main menu

First, type `a` to check or alter the current settings for the connection. The screen in Figure 61 appears:

```
Node: minnie          ALTER CONNECTION SETTINGS
------------------------------------------------------------------------
 COMMAND   DESCRIPTION             CURRENT          POSSIBLE CHOICES
---------  ---------------------  --------  ------------------------
Length     Bits per character     8         7,8
Stop       Number of stop bits    1         1,2
Parity     Parity setting         0         0=none, 1=odd, 2=even
Rate       Number of bits/second  1200      50,75,110,134,150,300,600,
                                            1200,2400,4800,9600,19200

Device     /dev name of port      tty0      tty0-tty16
Initial    Modem dialing prefix   ATDT      ATDT, ATDP, etc.
Final      Modem dialing suffix              0 for none, modem suffix
Wait       Wait between redialing 0         seconds between tries
Attempts   Maximum redial tries   0         0 for none, positive integer

Transfer   File transfer method   p         p=pacing, x=xmodem
Character  Pacing char or number  0         0 for none, one char/integer
------------------------------------------------------------------------
To change a current choice, type the first letter of the command followed
by your new choice (example:  r 300) and press Enter.
>
```

Figure 61. Altering the communication using the ATE menu

Alter the parameters to reflect the settings and characteristics of your modem. For instance, to change the rate to 9600 and the device to tty1, you should enter:

```
> r 9600
> d tty1
```

The new values for the parameters will be displayed in a updated screen.

Type `<return>` to return to the main screen, and then try to connect to the remote node by entering:

```
> c <phone_number>
```

If everything is correct, you should receive the prompt login from the remote node. Log in as the user you created to be used for DBFS. Check the value of the PS1 variable and save it. You will need this information later.

5. Configure the dbfs.envfile:

   The /usr/sbin/gmd/scripts/dbfs.envfile needs to be modified. This is what the dbfs.envfile looks like, as delivered:

```
(...)
export MLOGINPROMPT="Username:"
export MLOGIN="guest"
export MPASSWDPROMPT="Password:"
export MPASSWD="guest"
# If modem login requires a node name, this is the prompt string
export TSPROMPT="ts>"
# Login prompt, username, password prompt, and password on remote
system.
export NLOGINPROMPT="login:"
export NLOGIN="root"
export NPASSWDPROMPT="root's Password:"
export NPASSWD="xxxxx"
# Shell prompt on remote system.
export NPROMPT="%"
# Directory in which the dbfs utility resides.
export HAGEO_UTILS="/usr/sbin/gmd/scripts"
(...)
```

   Here is what it looks like after modification. Change the DEVICE variable to the proper tty that you configured to be used by DBFS:

```
(...)
# Device modem is connected to
export DEVICE=tty1
```

   Set the proper baud rate:

```
# Baud rate of modem line
export BAUDRATE=9600

# If the modem being used requires a login, enter its login prompt,
# username, and password here.
# If the modem directly connects to the system login, set the following
# group of variable to blank. example export MLOGINPROMPT="".  (Note
there
# cannot be any white space.)
```

   If the modem being used does not require a login, make sure the `M*` variables are set to null (`""`):

```
export MLOGINPROMPT=""
export MLOGIN=""
export MPASSWDPROMPT=""
export MPASSWD=""
```

```
# If modem login requires a node name, this is the prompt string
export TSPROMPT=""
```

Change the N* variables to reflect the user and its password that you configured to be used by DBFS:

```
# Login prompt, username, password prompt, and password on remote
system.
export NLOGINPROMPT="login:"
export NLOGIN="dbfs"
export NPASSWDPROMPT="dbfs's Password:"
export NPASSWD="dbfs"
```

Change the `NPROMPT` variable to the value, $PS1, for the user you created:

```
# Shell prompt on remote system.
export NPROMPT="$PS1"

# Directory in which the dbfs utility resides.
export HAGEO_UTILS="/usr/sbin/gmd/scripts"
```

After you have configured the dbfs.envfile, you are ready to test your configuration. The next section guides you through a test of your DBFS environment.

5.2.2 DBFS functional test

You should test your DBFS configuration prior to starting HACMP. HACMP uses the `/usr/sbin/gmd/scripts/dbfs` command, which is called from the /usr/sbin/gmd/scripts/geo_dbfs script. The syntax for the geo_dbfs script is as follows:

```
$HAGEO_UTILS/geo_dbfs -e dbfs.envfile -n $node -p $MODEM_NUM
```

where `$node` is the remote node name and `$MODEM_NUM` is the remote node phone number that is set in the script. The `geo_dbfs` script generates an output file, called /tmp/dbfs.out. You can use this file to trace problems related to the configuration of the dbfs.envfile file.

You can also use the `dbfs` command directly:

```
# cd /usr/sbin/gmd/scripts
# . ./dbfs.envfile
# ./dbfs -n mickey -p 86911
```

If DBFS is properly configured, an attempt to log in to a remote node using the `ate` command occurs. If the login is successful, the `/usr/sbin/gmd/scripts/amiup` command is run. This command creates a shared memory segment that has the same key as clinfo. It is used to check

the current status of the node and report it to the remote GeoManager. Make sure that when you start HACMP on the nodes, the clinfo daemon is also started.

Start HACMP by entering `smit clstart`. The screen in Figure 62 appears.

```
                        Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Start now, on system restart or both                now

  BROADCAST message at startup?                       false
  Startup Cluster Lock Services?                      true
  Startup Cluster Information Daemon?                 true



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 62. SMIT HACMP start screen

Change the last two options to *true*.

For some of the more complex failures that HAGEO handles, clinfo needs to be in trap mode. To configure this, run the following command in both machines:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

You can check the current clinfo settings with the following command:

```
# odmget -q subsysname=clinfo SRCsubsys
SRCsubsys:
        subsysname = "clinfo"
        synonym = ""
        cmdargs = "-a"
        path = "/usr/sbin/cluster/clinfo"
        uid = 0
        auditid = 0
        standin = "/dev/null"
        standout = "/dev/null"
```

```
standerr = "/dev/null"
action = 1
multi = 0
contact = 3
svrkey = 0
svrmtype = 0
priority = 20
signorm = 0
sigforce = 0
display = 0
waittime = 15
grpname = "cluster"
```

If clinfo is running on the remote node and HACMP has been started, the `amiup` command will return an IMOK string and exit code of 0. If the node's clinfo daemon is not running, the following message is given:

```
Can't get local cluster ID: Invalid shared memory operation.
```

If this happens, the return code from the command will be `1`.

5.2.3 Security Issues

Unfortunately, there are some security exposures in the DBFS function as it is shipped with the HAGEO product. To minimize these security exposures, we recommend the following actions:

1. Restrict the permissions of the GEOnode ODM class.
   The modem telephone numbers used by DBFS are stored in the HAGEO ODM class, GEOnode, in the $ODMDIR directory. The default permissions value for this file is 0644. With these permissions, the modem numbers can be easily obtained from the ODM by any user that has access to the node. The permissions for this file should be changed to 0600.
2. Create a non-root user.
   The default dbfs.envfile is configured with root as the DBFS user. This is too dangerous, because the file will contain the password of root in ordinary text format. It is highly recommended that you create a non-root user for this purpose. To reduce the risk that the characteristics and environment variables for this user get changed and DBFS does not work properly when needed, the DBFS user should not have write permissions to the DBFS home directory.

   We assumed that the files /.profile and /.kshrc for root have the permission 700.
3. Change the permissions of the dbfs.envfile.
   As stated before, the password for the DBFS user is kept in the dbfs.envfile file in ordinary text format. The default permission for this file

is 0644, which makes the information contained therein easy to read or steal by anyone who can access this node. You should change this file's access permission from 0644 to 0600. This file is read by a process that is forked from the HACMP event manager, which has root user access, and is not read by the DBFS login user process.

4. Accessible tty.
   By default, all users are allowed to access all ttys including /dev/ttyX. /dev/ttyX is the device where the modem is connected. You can change this default behavior by manually editing the /etc/security/user file. Only the dbfs user should be able to access the system from this tty. To accomplish this, you need to edit the /etc/security/user file manually as shown in Figure 63.

```
default:
    ...
    admgroups =
    ttys = ALL,!/dev/tty1
    auth1 = SYSTEM
    ...
dbfs:
    admin = false
    ttys = /dev/tty1
```

Figure 63. Limit Accessible TTY with /etc/security/user

5.3 HAGEO configuration examples

This section illustrates the procedure of setting up an HAGEO environment. Three configuration scenarios are presented along with all the steps required to implement them. For each scenario, a test case is included with a description of the cluster behavior. Remember that the files shown are not the complete version.

In all the scenarios, the machines were running AIX 4.3.2, HACMP 4.3.0, and HAGEO 2.1.0. The fixes described in Section 4.3, "HAGEO installation" on page 86, were applied to HAGEO.

5.3.1 Scenario 1: One node per site

In this scenario, we are going to illustrate the two-node cluster configuration in which each node is located on a different site. This scenario is the basis for illustrating the procedure to configure HAGEO. Figure 64 shows the configuration that will be described.

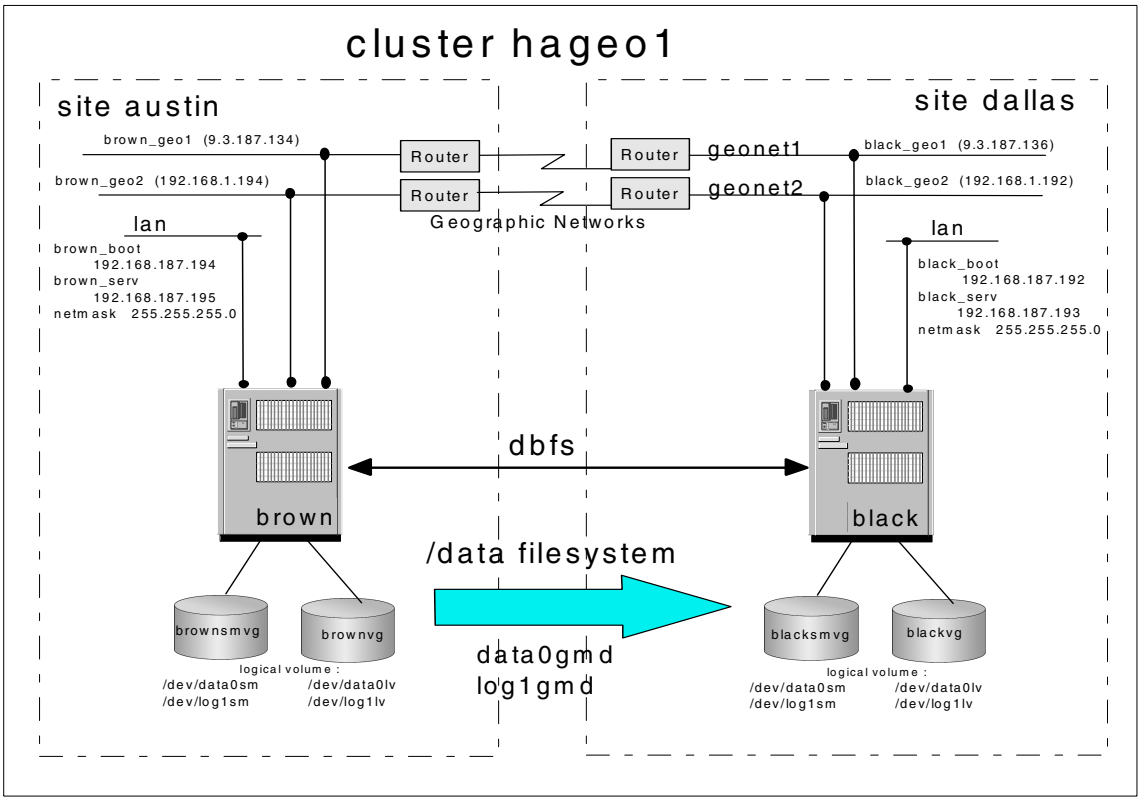

Figure 64. Cluster hageo1 scheme

The two sites are named austin and dallas, and the nodes brown and black are located in each of these sites respectively. The austin site is dominant and the dallas site is non-dominant. Two geographical networks connect the sites, geonet1 and geonet2. These networks are not used simultaneously. HAGEO uses the interface that currently provides the best response time based on historical data. See Table 24.

Table 24. IP labels

Machine name	Adapter name	IP address	Network name
brown	brown_geo1	9.3.187.134	geonet1
	brown_geo2	192.168.1.194	geonet2
	brown_boot	192.168.187.194	token_lan
	brown_serv	192.168.187.195	token_lan

Machine name	Adapter name	IP address	Network name
black	black_geo1	9.3.187.136	geonet1
	black_geo2	192.168.1.192	geonet2
	black_boot	192.168.187.192	token_lan
	black_serv	192.168.187.193	token_lan

Two GeoMirror devices (GMDs) are configured: data0gmd and log1gmd. These allow you to mirror the file system /data between the two sites. See Table 25.

Table 25. GMD and logical volume dependency

GMD name	Logical volume	Brown VG	Black VG
data0gmd	data0lv	brownvg	blackvg
	data0sm	brownsmvg	blacksmvg
log1gmd	log1lv	brownvg	blackvg
	log1sm	brownsmvg	blacksmvg

5.3.1.1 Prerequisites

As part of the planning process, all network adapters and disks should have been installed and configured. Review the following checklist:

- Check that the following filesets are installed:
  - `bos.compat`
  - `bos.net.ate` - This is necessary for DBFS
- Check that the VGs for the data and state map logical volumes on both machines have been created. These VGs can be set to not start automatically at boot.
- Create the data and state map logical volumes on the two machines. Each of the state map LVs is one logical partition size. The size used by the state map is always 640 KB; so, a logical volume of one physical partition will always be enough to house it.
- Install HACMP on the system with the last PTFs available.
- Update the /etc/hosts and /.rhosts files with all the IP labels and addresses used in the cluster. The entries in the /etc/hosts file are shown in Figure 65.

```
9.3.187.134     brown_geo1      brown
9.3.187.136     black_geo1      black
192.168.1.192   black_geo2
192.168.1.194   brown_geo2


192.168.187.194 brown_boot
192.168.187.192 black_boot
192.168.187.193 black_serv
192.168.187.195 brown_serv
```

Figure 65. Entries in the /etc/hosts file

The entries in the /.rhosts file are shown in Figure 66.

```
brown_geo1
black_geo1
black_geo2
brown_geo2
blue_geo1
black_serv
brown_serv
brown
black_boot
brown_boot
```

Figure 66. Entries in the /.rhosts file

- Install HAGEO software and PTFs on each machine, and reboot the system.

5.3.1.2 HACMP cluster configuration

In order to work properly, HAGEO needs the HACMP software; therefore, you need to configure HACMP before you configure HAGEO. The cluster is configured in the same way as if the nodes were on the same sites.

The following procedure describes the steps necessary to configure the cluster topology:

1. Open the smitty HACMP for AIX menu using the `smitty hacmp` fastpath, and select ***Cluster Configuration***.
2. From the Cluster Configuration menu, choose **Cluster Topology**.
3. To add a new cluster definition, select **Configure Cluster->Add a Cluster Definition**. The Add a Cluster Definition screen appears as shown in Figure 67. Fill in the cluster ID with a unique number and name.

```
                    Add a Cluster Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  **NOTE: Cluster Manager MUST BE RESTARTED
          in order for changes to be acknowledged.**

* Cluster ID                                          [1]                    #
* Cluster Name                                         [hageo1]
```

Figure 67. Add a Cluster Definition screen

4. To add the nodes to the cluster, select **Configure Nodes->Add Cluster Nodes**. The Add Cluster Nodes screen shown in Figure 68 appears. Fill in the Node Names entry field with the names of the two nodes.

```
                         Add Cluster Nodes

  Type or select values in entry fields.
  Press Enter AFTER making all desired changes.

                                                       [Entry Fields]
  * Node Names                                        [brown black]
```

Figure 68. Add Cluster Nodes screen

5. To configure the network adapters, select **Configure Adapters->Add an Adapter**. The Add an Adapter screen shown in Figure 69 on page 136 appears. From there, define all the adapters that will be used in the cluster. The network type chosen for the geo adapters was Geo_Primary, which has a larger timeout value that is more suitable for WANs. WANs can come with many different kinds of behaviors, speeds, and latencies. You can alter the ODM class HACMPnim to reflect the kind of WAN you are using, but you can also choose the standard network types from HACMP, such as FDDI or Token Ring. Another important issue is that all the geographical adapters must be configured as private. Refer to Table 24 on page 132 for the adapter definitions.

```
                        Add an Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Adapter IP Label                                   [black_geo1]
* Network Type                                       [Geo_Primary]       +
* Network Name                                       [geonet1]           +
* Network Attribute                                   private            +
* Adapter Function                                    service            +
  Adapter Identifier                                 [9.3.187.136]
  Adapter Hardware Address                           []
  Node Name                                          [black]                +
```

Figure 69. Add an Adapter screen

6. After configuring the adapters, the cluster topology configuration is complete; so, you need to synchronize the configuration. Select **Synchronize Cluster Topology**.
7. Verify that the synchronization is using the `/usr/sbin/cluster/diag/clverify` utility.

At this point, the cluster topology configuration is complete. For further information about configuring HACMP, refer to the *HACMP V4.3 AIX: Install Guide*, SC23-4278.

5.3.1.3 HACMP resource group configuration

The resource group creation for the HAGEO critical resources is somewhat different than a normal HACMP configuration. HAGEO resource groups within HACMP may include:

- Volume groups and hdisks on which GMDs will be created
- File systems mounted on GMDs
- Applications using (writing/reading) GMDs and GMD file systems

No other resources are handled by the HAGEO integration, and any resources using non-HAGEO resources cannot be defined in the same resource groups as HAGEO resources. To optimize the performance of the HAGEO scripts and integration, Async GMDs should not be in the same

resource groups as synchronous GMDs.

Note

Within the resource groups, MWC and sync GMDs may be grouped together and are handled in the same manner.

The resource group configuration is created on one node and, afterwards, is synchronized to the other node. To create a new resource group definition from the Cluster Configuration menu for HACMP, select **Cluster Resources->Define Resource Groups->Add a Resource Group**. The Add a Resource Group screen shown in Figure 70 appears.

```
                              Add a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Resource Group Name                                [res_black]
* Node Relationship                                   cascading             +
* Participating Node Names                           [black]                +
```

Figure 70. Add a Resource Group screen

We have to define a resource group for each node. We will call res_black the resource group for the black machine, and res_brown will be the resource group for the brown machine.

Then, we have to add the resources that are part of the resource group. From the Cluster Resources menu, choose **Change/Show Resources for a Resource Group**. Select the resource group to define, and the screen shown in Figure 71 on page 138 appears.

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
  Resource Group Name                                  res_black
  Node Relationship                                    cascading
  Participating Node Names                             black

  Service IP label                                    [black_serv]            +
  Filesystems                                         []                      +
  Filesystems Consistency Check                        fsck                   +
  Filesystems Recovery Method                          sequential             +
  Filesystems to Export                               []                      +
  Filesystems to NFS mount                            []                      +
  Volume Groups                                       [blacksmvg blackvg]     +
  Concurrent Volume groups                            []                      +
  AIX Connections Services                            []                      +
  AIX Fast Connect Services                           []                      +
  Application Servers                                 []                      +
  Highly Available Communication Links                []                      +
  Miscellaneous Data                                  []
  Inactive Takeover Activated                          false                  +
  9333 Disk Fencing Activated                          false                  +
  SSA Disk Fencing Activated                           false                  +
  Filesystems mounted before IP configured             false                  +
  Raw Disk PVIDs                                      []                      +
```

Figure 71. Configure Resources for a Resource Group screen

For each node resource group, the resources to define are:

- **Service IP label field** - This is the IP label for the service adapter address. For the res_black resource group, this is black_serv. For the res_brown resource group, it is brown_serv.
- **Volume Groups field** - The volume groups, in which the GMD logical volumes and data logical volumes are located, must be inserted here. For the res_black resource group, this will be blackvg and blacksmvg. For the res_brown resource group, this will be brownvg and brownsmvg.

After you have configured the two resource groups on one machine, you have to synchronize it. Go back to the Cluster Resources menu and select **Synchronize Cluster Resources**. Leave the default and press **Enter** to execute it.

5.3.1.4 GeoMessage configuration

The GeoMessage subsystem is controlled by its three configuration ODM classes: KRPC_MACHINE, KRPC_NETWORK and KRPC_INTERFACE. These classes look much like the HACMP for AIX classes for cluster/node configuration.

In order to configure the KRPC classes, go back to the starting menu for HAGEO and select **Manage GeoMessage** and next **Configure GeoMessage**. The screen shown in Figure 72 appears.

```
                    Configure GeoMessage

Move cursor to desired item and press Enter.

  Show GeoMessage Configuration
  Import HACMP Definitions
  Distribute GeoMessage Definitions to Remote Machines
  Configure GeoMessage Machines
  Configure GeoMessage Networks
  Configure GeoMessage Interfaces
```

Figure 72. Configure GeoMessage screen

There are two ways to configure GeoMessage. You can import the definitions from HACMP and remove the ones not related to HAGEO, or you can use the three last options of the Configure GeoMessage menu to define the machines, networks, and interfaces.

In this example, we are going to import the HACMP definitions. Select **Import HACMP Definitions**. In this scenario, there is no information to remove. All the definitions created in HACMP are going to be used by HAGEO. Select **Show GeoMessage Configuration** to verify the configuration that was created.

For instance, select **Show GeoMessage Configuration**, and then select **Show defined interfaces**. The output for this scenario is as follows:

```
------------------------------------------------------------
Interface Name      Address         Host      Network Link
------------------------------------------------------------
black_geo1          9.3.187.136     black     geonet1
black_geo2          192.168.1.192   black     geonet2
brown_geo1          9.3.187.134     brown     geonet1
brown_geo2          192.168.1.194   brown     geonet2
```

Next, go to the Configure GeoMessage menu and select **Distribute GeoMessage Definitions to Remote Machines**.

Note

The names of all the machines in the cluster must be resolved into an IP address (using /etc/hosts or DNS) associated with one of the geographical network interfaces, and the /.rhosts file must be configured for the synchronization operation to work properly.

5.3.1.5 GeoManager configuration

In this step, we configure the GEOnode and GEOsite ODM classes. To do this, we will use the SMIT menu for HAGEO. To access it, issue the `smit hageo` command. The HAGEO for AIX screen shown in Figure 73 appears.

```
                        HAGEO for AIX

Move cursor to desired item and press Enter.

  Manage GeoMessage
  Manage GeoMirror
  Manage HAGEO Sites/Machines
  Verify HAGEO Configuration
  HAGEO Utilities
```

Figure 73. HAGEO for AIX screen

Select **Manage HAGEO Sites/Machines**. The screen shown in Figure 74 appears.

```
                  Manage HAGEO Sites/Machines

Move cursor to desired item and press Enter.

  Configure HAGEO Site
  Configure HAGEO Machine
  Show Sites/Machines
  Sync Site/Machine Definitions to all Machines
```

Figure 74. Manage HAGEO Sites/Machines screen

First, we need to configure the sites. Select **Configure HAGEO Site**, and then select **Add HAGEO Site Definition**. The screen shown in Figure 75 on page 141 appears.

```
                    Add HAGEO Site Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Site Name                                          [austin]
* Dominance                                          [yes]             +
* Backup Communications                              [DBFS]            +
* Site Machines                                      [brown]
```

Figure 75. Add HAGEO Site Definition screen

In the Site Name field, fill in the name you want for the site. It cannot be greater than 31 characters long.

Next, select **yes** if you want this to be the dominant site or **no** if it is going to be the non-dominant site. The dominance defines which site will be halted when site isolation occurs. Site isolation happens when all the geographical networks are down, but at least one node at each site is still up. To prevent data divergence, the non-dominant site is halted.

You should reflect on some issues before deciding which site will be the dominant one. For instance:

- Is one site more prone to disaster than the other?
- Does one site have more critical data than the other?
- Is one site accessed by more clients than the other?

Next you should select the type of backup communication you will use, if you are going to use backup communication at all. Select **dbfs** if you are using a telephone line or **sgn** if it is a Geo_Secondary network. The selected method will be used by HAGEO to distinguish between a site isolation or a site failure.

You should complete the Site Machines field with the node names that belong to the site being configured.

After configuring sites austin and dallas, go back to the Manage HAGEO Sites/Machines menu. Select **Configure HAGEO Machine**, and then select **Add HAGEO Machine Definition**. The screen shown in Figure 76 on page 142 appears.

```
                    Add HAGEO Machine Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Machine Name                                       [brown]
* Site Name                                          [brown]
  System administrator ID                            [root]
  Modem NUMBER                                       [8386911]         #



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 76. Add HAGEO Machine Definition screen

In the field Machine Name, enter the name of the node, and, in Site Name, enter the name of one of the sites defined previously. The System administrator ID field defines the login ID or e-mail ID of the user who should receive e-mail that HAGEO sends when a failure occurs. This ID can be a single one, or it can be an alias representing a group of administrators. The Modem NUMBER field only applies if you are using DBFS. If that is the case, fill it out with the telephone number of the modem on that machine.

You should repeat this configuration for machine black. The site and machine definition for all nodes in the cluster can be done from one node. You will want to choose one of the nodes on which to do the definitions and then synchronize the definitions to all the machines. Go back to the Manage HAGEO Sites/Machines menu. Select **Sync Site/Machine Definitions to all Machines**.

5.3.1.6 Integrating HACMP and HAGEO and configuring events

Execute the following command on each node:

```
# /usr/sbin/gmd/scripts/configure_events <machine_name>
```

This command will plug the GMD events into HACMP. The next time HACMP is started, the GMD devices will come on automatically.

5.3.1.7 GeoMirror device configuration

You now need to start GeoMessage on all nodes prior to configuring the GeoMirror devices. Go back to the Manage GeoMessage menu, and select

Start GeoMessage. You can also use the `/usr/sbin/krpc/cfgkrpc -ci` command.

In this scenario, the file system, /data, is going to be mirrored across the geography. To do so, two GeoMirror devices (GMDs) need to be configured. The data0gmd GMD is associated with the logical volume for /data, data0lv. The log1gmd is associated with the jfslog logical volume from /data, log1lv. You first need to create two logical volumes in each site. Notice that the logical volume names must be the same in both nodes and must also have the same size.

Two other logical volumes also need to be created in each node to be used as state maps: data0sm and log1sm on both nodes. Each of them is one logical partition in size. The size of a state map is always 640 KB; so, a logical volume of one physical partition will always be enough to house it. Refer to Table 25 on page 133.

After creating the logical volumes, you need to go back to SMIT to configure the GMD devices. Enter `smit hageo`. Then, select **Manage GeoMirror** followed by **Configure a GeoMirror Device**. Next, choose **Add a GeoMirror Device**. The screen shown in Figure 77 on page 144 appears.

```
                    Add a GeoMirror Device

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Device Name                                        [data0gmd]
  Device State                                       stopped               +
* Minor Device Number                                [0]                   #
* State Map Logical Volume                           [/dev/rdata0sm]
* Local Logical Volume                               [/dev/rdata0lv]
* Device Mode                                        async                 +
* Device Role                                        primary               +
  High Water Mark                                    []                    #
  Default Sync Concurrency Rate                      []                   +#
  Remote Machine and Logical Volume                  [black@/dev/rdata0lv]
  Remote Machine and Logical Volume                  []
  Remote Machine and Logical Volume                  []
  Remote Machine and Logical Volume                  []
  Remote Machine and Logical Volume                  []
  Remote Machine and Logical Volume                  []
  Remote Machine and Logical Volume                  []
  Local Peer and state map device                    []
  Local Peer and state map device                    []
  Local Peer and state map device                    []
  Local Peer and state map device                    []
  Local Peer and state map device                    []
  Local Peer and state map device                    []
```

Figure 77. Add a GeoMirror Device screen

The Device name field should be filled in with the GMD name being configured. Leave the Device name stopped for now. The Minor Device Number defines the GeoMirror minor device number. A minor number identifies a particular device among the several devices handled by the device driver associated with the major number. This number as well as the GMD name must be the same on both local and remote parts of the device.

Next, in the State Map Logical Volume field, enter the name of the logical volume you created for the state map. Notice that the full path to the logical volume device must be given, and the `r` indicates that this is a raw LV. You should always use the raw form of the device name when defining GMDs. The state map name must be unique within the HAGEO cluster, and it can have up to 15 characters.

Now you need to supply the local logical volume name. Once again, specify the full path and prefix the LV name with `r`. This name must be the same across the geography and can have up to 15 characters.

You should choose async, sync, or mwc for the Device Mode. If you choose async, you need to define the device role. If the device mode is sync or mwc, leave the default value (none).

Note

Both the file system and the jfslog must have the same the Device Mode.

When you assign the roles of an async device, you define one side of the GeoMirror device as primary (where I/O originates) and the other side as secondary (a backup device that only receives data from the primary). Only the primary side allows open requests.

In most cases, it is logical to assign the primary role to the devices that reside at the site you define as the dominant site so that, if a problem occurs with the HAGEO networks, users can still write to the application. When the situation is corrected, in order to update the mirrors, the devices at the non-dominant site will be synchronized when they start up or when they join the cluster.

The High Water Mark attribute controls how many 1 KB blocks of data are allowed to be in flight asynchronously at any given moment. This attribute is only used by async devices on the primary side. The default value is 128.

The Default Sync Concurrency Rate specifies the maximum total amount of data that is transferred at one time during a sync operation. This is the data being read from the remote site by the local site when the local site is reintegrating. This number only affects the sync rate when the local node is reintegrating. Speeding up the sync_rate makes data available sooner on the resyncing side, but it degrades performance on the remote site. The default value is 32 KB, and it can be changed to 64 KB.

The Remote Machine and Logical Volume specify the name of the remote peer node and the logical volume name on that node for the GMD being created. Remember that the logical volume name needs to be the same for both sides of the GMD device.

The Local Peer and state map device as well as the remaining Remote machine and LV fields do not apply in the case of an async device. An async device can only be taken over by one remote peer. It cannot be taken over by a local peer.

Now you need to configure the log1gmd device in node brown and repeat the definition of the GMDs on node black. The GeoMirror device configuration is

not synchronized, but, rather, these definitions must be made on each node describing that node's point of view. Also, the local node does not have all the information to create the GMD on the remote node. It does not know the name of the state map on this node.

After creating all the GMD devices, you are able to create the /data file system. First, start all the GMD devices in both nodes. This can be accomplished by selecting **Start a GeoMirror Device** from the Manage GeoMirror menu.

If it is not known whether the remote peer is down, the GMD's devices may take a while to start. To prevent this from happening, you can use the commands below instead of starting the GMDs through SMIT. On node brown, enter:

```
# /etc/methods/cfggmd -A
```

This command will put the GMD in the stopped state, which is necessary for the next command:

```
# /etc/methods/gmddown -A black
```

This command will inform the node on which you enter the command that the remote GMD is down, and the GMD being started will not have to wait for the network time-out period. The next command actually makes the GMD available:

```
# /etc/methods/startgmd -A
```

On node black, you should not execute the `gmddown` command because the GMD is already started on node brown. The `gmddown` command informs a local GMD that a remote device is down. The GMD will not send data to or accept requests from that device. Service resumes when the remote device is restarted. Note that, if a user issues the `gmddown` command and the remote device is still available, the remote device is stopped to protect against data divergency.

If the `gmddown` command is issued erroneously (that is, if the remote device is not down) and you use SMIT or the `lsdev -Cc geo_mirror` command to display the device status, the device status will be listed as available. The `lsdev` command displays device information from the Device Configuration database. The Device Configuration database is not updated when the device is brought off-line internally. If this command is issued by mistake, you must first stop and then restart the device on the remote node to return it to service.

After the GMD is started on node brown, issue the following commands on node black to start the GMDs in this node:

```
# /etc/methods/cfggmd -A
# /etc/methods/startgmd -A
```

Note

If one of the nodes that can take over the GMD is powered off, or is inaccessible through TCP/IP, the GMDs will take a long time to start, even if you issue the `gmddown` command as explained earlier.

Next, edit the /etc/filesystems file on both nodes and add the following entry:

```
/data:
    dev         = /dev/data0gmd
    vfs         = jfs
    log         = /dev/log1gmd
    mount       = false
    check       = false
    options     = rw
    account     = false
```

Notice that the dev and log fields point to the GeoMirror device names.

Next, create the mount point for the file system /data in both nodes with the `mkdir /data` command. Then, on the local machine, execute the command:

```
# mkfs -l data /dev/data0gmd
```

At this point, writes will occur at both sites because the file system will be created on both physical devices in `/dev/data0gmd`. You may want to execute the `fsck` command on the newly-created file system to verify the file system. Then, mount the file system on node brown. You will not be able to mount the file system on node black, whose GMD's device role is secondary. If node black takes over /data, the GMDs will change role from secondary to primary in order to mount /data on black.

5.3.1.8 HACMP shared resources configuration

Now, we have to create the shared resource group in HACMP to handle the critical file system /data. To do this, we need to go back to the HACMP's Cluster Resources smit menu, execute smit hacmp, and select **Cluster Configuration->Cluster Resources**.

Select **Define a Resource Group**, and, as done before, create the resource group called hageo. In the Participating Node Name field, type the site names `austin` and `dallas`. Press **Enter**, and go back to the Cluster Resources menu,

select **Change/Show Resources for a Resource Group**, and choose the resource group, **hageo**, from the popup list.

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                [Entry Fields]
  Resource Group Name                                hageo
  Node Relationship                                  cascading
  Participating Node Names                           austin dallas

  Service IP label                                  []                      +
  Filesystems                                       [/data]                 +
  Filesystems Consistency Check                      fsck                   +
  Filesystems Recovery Method                        sequential             +
  Filesystems to Export                             []                      +
  Filesystems to NFS mount                          []                      +
  Volume Groups                                     []                      +
  Concurrent Volume groups                          []                      +
  Raw Disk PVIDs                                    []                      +
  AIX Connections Services                          []                      +
  AIX Fast Connect Services                         []                      +
  Application Servers                               []                      +
  Highly Available Communication Links              []                      +
  Miscellaneous Data                                 [data0gmd log1gmd]

  Inactive Takeover Activated                        false                  +
  9333 Disk Fencing Activated                        false                  +
  SSA Disk Fencing Activated                         false                  +
  Filesystems mounted before IP configured           false                  +
```

Figure 78. Configure Resources for a Resource Group screen

The only resource to add to this group is the file system /data, and, in the Miscellaneous Data field, you specify the GMD(s) name(s) associated with the file system.

Note that, for file systems, you do not specify node names. Instead, the site names should be given. This same procedure should be used to implement application servers across the geography.

When defining an application server or a file system resource group, you must include the related GMD names in the Miscellaneous Data field. This is required for all GMD types.

If you execute the `clverify` utility, it will return an error because it does not know how to handle site names. You may ignore this error. Once again, do not forget to synchronize the configuration to the other node.

After adding this resource group on one machine, you have to synchronize the cluster resource group with the other node again. Go back to the Cluster Resources menu and select **Synchronize Cluster Resources**. This screen is shown in Figure 79.

```
                    Synchronize Cluster Resources

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
  Ignore Cluster Verification Errors?                 [Yes]                    +
  Un/Configure Cluster Resources?                      [Yes]                    +
* Emulate or Actual?                                   [Actual]                 +
```

Figure 79. Synchronize Cluster Resources screen

Change the Ignore Cluster Verification Errors field to *Yes*. You need to change it or else the synchronization will be stopped because it is reporting the errors shown in Figure 80.

```
Verifying Configured Resources...
ERROR: Node austin, participating in resource group hageo, is not configured
in the Cluster Topology.

ERROR: Node dallas, participating in resource group hageo, is not configured
in the Cluster Topology.
```

Figure 80. Synchronization errors

These errors are normal because the clverify utility does not know how to handle site names; they are part of HAGEO site configuration. You may ignore these errors. Also, if the command status reports *Failed*, the synchronization with the other node is done.

5.3.1.9 DBFS customization

To use the DBFS feature, you need to customize the /usr/sbin/gmd/scripts/dbfs.envfiles file and customize the login information. Refer to Chapter 6, “Administration” on page 173, for more information about this.

5.3.1.10 Verifying the configuration

Now, you need to verify the HAGEO configuration to ensure that it is correct and consistent across all machines in your HAGEO environment. Enter `smit hageo`, and select **Verify HAGEO Configuration**.

You can also verify the HAGEO configuration by executing the following command:

```
# /usr/sbin/gmd/geo_verify
```

The `geo_verify` utility checks for many possible configuration errors that could cause serious problems at run time. For example, if you mistakenly listed a machine at the local site as a remote peer for a GeoMirror device, some writes would not be sent to the remote site, causing a loss of data integrity. Running the `geo_verify` utility after the configuration is initially set up, and any time it is changed, will greatly reduce the occurrence of such problems.

5.3.1.11 Starting HACMP

For some of the more complex failures that HAGEO handles, `clinfo` needs to be in trap mode. To achieve this, you need to run the following command on all nodes:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

You also need to make sure that all the resources that are handled by HACMP are released. Unmount the /data file system if it is still mounted. Then, stop the GMDs with the following commands:

```
# /etc/methods/stopgmd -A
# /etc/methods/ucfggmd -A
```

The first command changes the state of the GMD from *available* to *stopped*, and the second alters it to *defined*. You can also stop the GMDs through SMIT. Enter `smit hageo`; select **Manage GeoMirror**, and then select **Stop a GeoMirror Device**. Next, unload GeoMessage with the following command:

```
# /usr/sbin/krpc/cfgkrpc -u
```

Finally, vary off the volume group geovg.

Now, you are able start HACMP. Execute `smit clstart`. The Start Cluster Services screen appears as shown in Figure 81 on page 151.

```
                    Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Start now, on system restart or both               now              +

  BROADCAST message at startup?                      true             +
  Startup Cluster Lock Services?                     true             +
  Startup Cluster Information Daemon?                true             +
```

Figure 81. Start Cluster Services screen

Make sure you change the Startup Cluster Lock Services and Startup Cluster Information Daemon options to *yes*.

You need to start HACMP on both nodes.

After HACMP startup is finished, the GeoMirror devices should be started in both nodes and the file system, /data, mounted on node brown.

5.3.1.12 Testing the cluster

Besides running clverify and geo_verify utilities, you also need to run various tests in the cluster to make sure that the configuration behaves as desired. First, make sure that, after starting HACMP in both nodes, all the GMDs were started and the file system, /data, was mounted. At this time, you are able to begin testing the cluster. Some suggested tests for this scenario and our results follow.

Site isolation testing

A site isolation scenario is shown in Figure 82 on page 152.

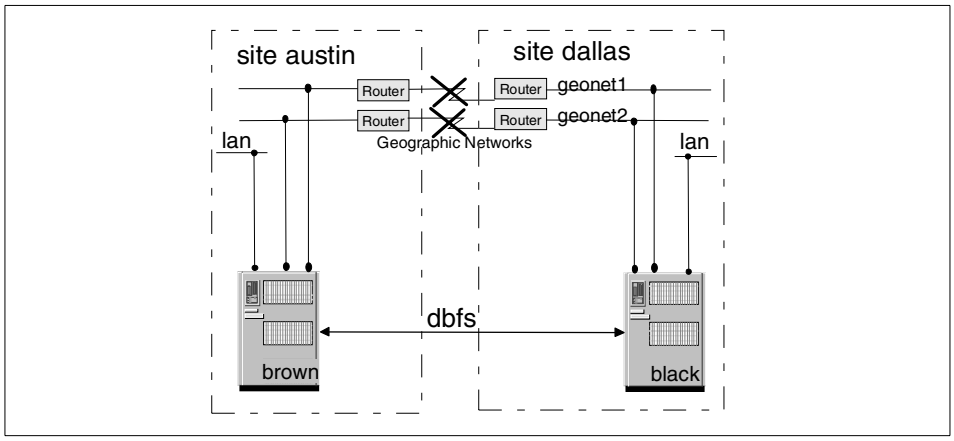

Figure 82. Cluster hageo1: Site isolation scenario

First, make all the geographical networks unavailable. This means that you should execute the `ifconfig <interface> down` command in brown or pull out the cables from the interfaces that belong to brown. You can also issue the `ifconfig <interface> down` command or pull out the cables from black. The behavior of the cluster is the same in all cases.

Brown realizes that the communication with black has stopped, and the node_down event takes place. Since brown is dominant, it does not need to know if it was a site disaster or site isolation. Brown executes the `gmddown` command against the available GMDs to stop the mirroring across the geography.

Black detects that the communication with brown is down, but, since black is in the non-dominant site, it needs to check if it was only a site isolation or an actual site failure. It uses DBFS to call brown. When black realizes that brown is still up, it calls geo_drop_node to exit the cluster. HACMP is stopped with the takeover option. Since it did not stop in 60 seconds, black is halted.

Site failure on Austin

A site failure scenario is shown in Figure 83 on page 153.

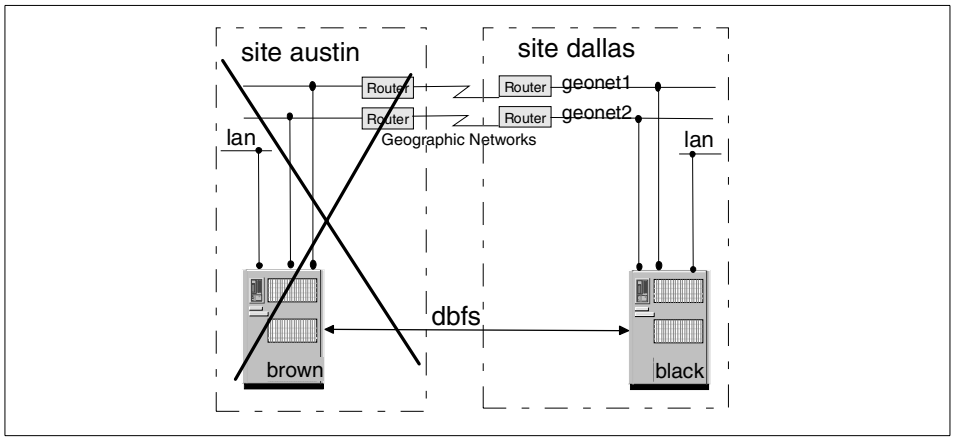

Figure 83. Cluster hageo1: Site failure scenario

To simulate this case, power-off brown or stop HACMP on brown with the takeover option.

Node black detects that communication with brown is down. It uses DBFS to confirm that it was really a site failure. It tries to call three times. At this point, the config_too_long event starts. Next, black switches the GMD to primary and starts them again. The GMDs take a long time to start because brown is powered-off. The /data file systems are mounted, and the node_down brown event ends successfully. Then, the event config_too_long stops.

Reintegrating the austin site
Power on brown, and start HACMP to bring it back to the cluster.

When black detects that brown is reentering the cluster, it first unmounts the /data file system. Then, the GMDs are stopped and changed to Defined state so that HACMP can change their device roles back to secondary. Finally, black restarts the GMDs.

Brown varies on the volume groups browndatavg and brownsmvg, starts GeoMessage, starts the GMDs, and mounts the /data file system.

5.3.2 Scenario 2: Three node configuration

A three-node configuration scenario is presented in Figure 84 on page 154.

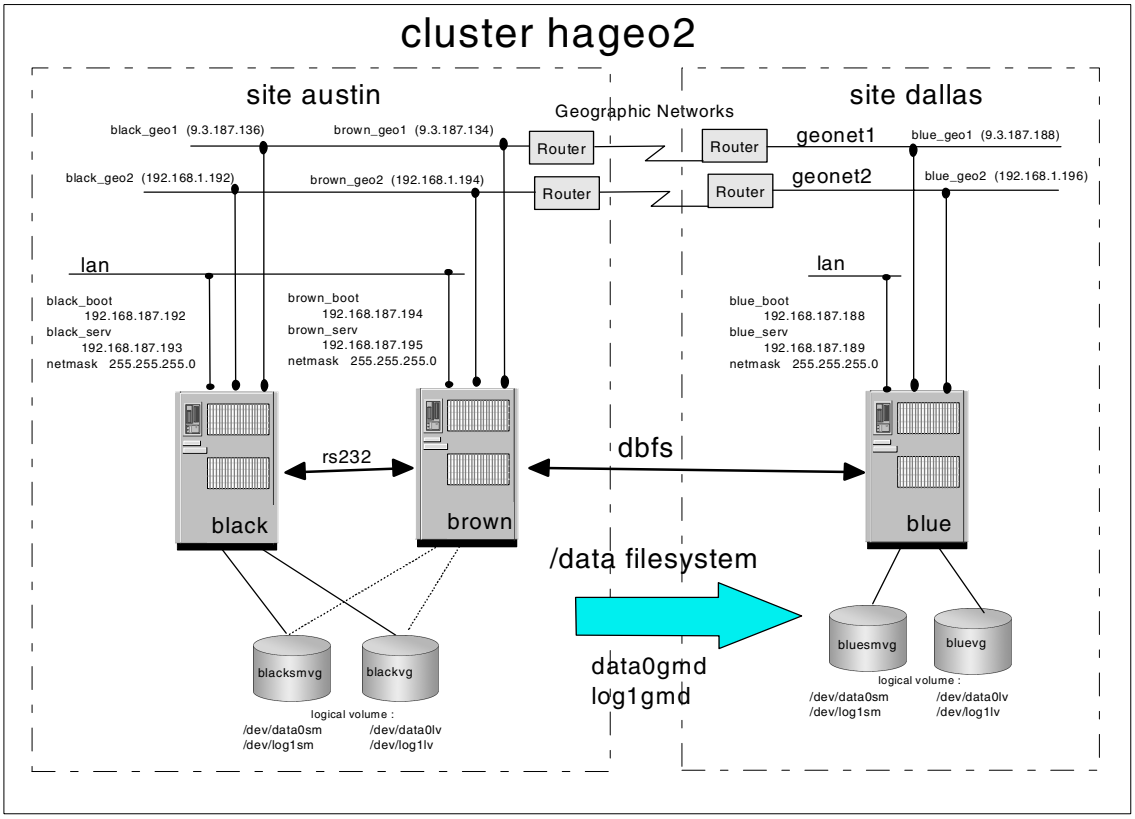

Figure 84. Cluster hageo2 scheme

In this scenario, we will have two cluster nodes in the local site and one node in the remote site. The site austin is the production site; so, it is dominant. The two sites are connected using two geographics networks: geonet1 and geonet2. The IP labels are listed in Table 26.

Table 26. IP labels

Machine name	Adapter name	IP address	Network name
brown	brown_geo1	9.3.187.134	geonet1
	brown_geo2	192.168.1.194	geonet2
	brown_boot	192.168.187.194	token_lan
	brown_serv	192.168.187.195	token_lan

Machine name	Adapter name	IP address	Network name
black	black_geo1	9.3.187.136	geonet1
	black_geo2	192.168.1.192	geonet2
	black_boot	192.168.187.192	token_lan
	black_serv	192.168.187.193	token_lan
blue	blue_geo1	9.3.187.188	blue_geo1
	blue_geo2	192.168.1.196	blue_geo2
	blue_boot	192.168.187.188	token_lan
	blue_serv	192.168.187.189	token_lan

Two GeoMirror devices (GMD) are configured: data0gmd and log1gmd. These allow mirroring of the file system /data between the two sites. GMDs and logical volume dependencies are listed in Table 27.

Table 27. GMD and logical volume dependency

GMD name	Logical volume	Brown VG	Black VG
data0gmd	data0lv	brownvg	blackvg
	data0sm	brownsmvg	blacksmvg
log1gmd	log1lv	brownvg	blackvg
	log1sm	brownsmvg	blacksmvg

5.4 Three nodes in mutual takeover configuration

In this scenario, we show a three-node cluster example in a mutual takeover configuration. The local site has two nodes, and the remote site has one node. Figure 85 on page 156 shows the sample configuration.

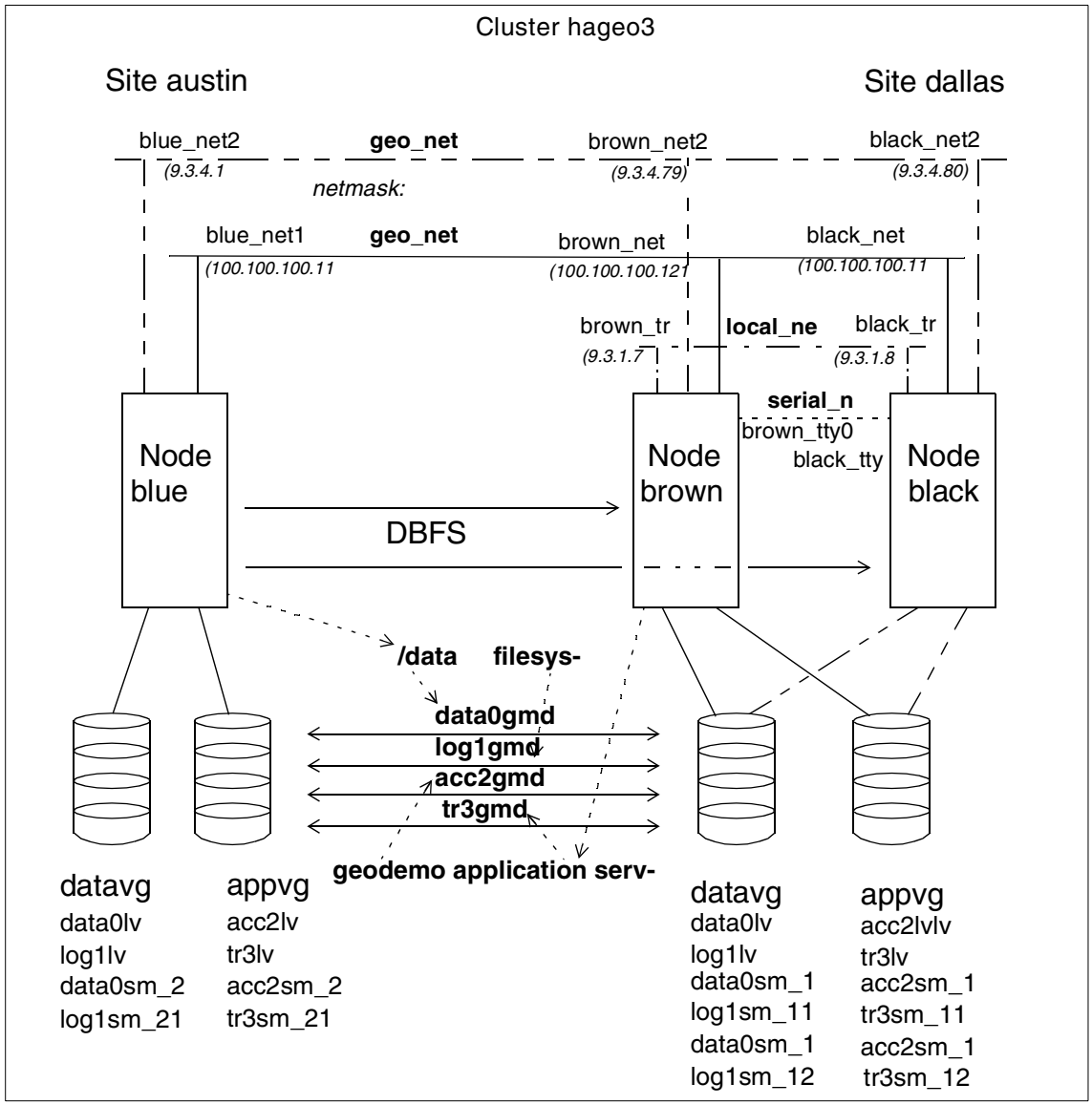

Figure 85. Scenario 3 Configuration

In Figure 85, you can see that the nodes, brown and black, are located in the site, dallas, and node blue is in site austin. Site dallas is dominant, and, consequently, austin is nondominant. Two geographical networks connect the sites, geo_net1 and geo_net2. There are also the local networks, local_net and serial_net. The GMDs, data0gmd and log1gmd, are associated with the

/data filesystem. This resource is shared between the sites, but site austin has the highest priority. The application server, geodemo, requires two GMDs, acc2gmd and tr3gmd. It can also be taken over by both sites, but site dallas has the highest priority.

5.4.1 Configuring the Cluster

This section shows the steps needed to configure this cluster.

5.4.1.1 Cluster Topology Configuration

As in the previous scenarios, you need to configure HACMP prior to configuring HAGEO. The cluster is configured in the same way as if the nodes were on the same site.

Cluster hageo3 was configured with three nodes: Brown, black, and blue. Six geographical adapters and four local adapters were defined. All the geographical adapters were configured with the Network Type as Geo_Primary, Network Attribute as private and Adapter Function as service.

You also need to configure brown_tr0 and black_tr0 local adapters. They should be configured as public, service and the local network type should be defined properly. The serial network must be configured. Adapters brown_tty0 and black_tty0 should be configured with Network Type as rs232, Network Attribute as serial and Adapter Function as service.

In this way you have created four networks: geo_net1, geo_net2, local_net and serial_net.

Now you should synchronize the configuration to the other nodes and verify it using `smit` or the `/usr/sbin/cluster/diag/clverify` utility.

For more information about configuring HACMP, refer to the *HACMP for AIX Installation Guide*, SC23-4278.

5.4.1.2 Configuring Sites and Machines

The configuration that will be described in this item only needs to be done in one of the nodes. At the proper time it will be synchronized to the other nodes.

You can use SMIT to configure sites austin and dallas and then the machines brown, black and blue. Refer to Section 5.1.2, “Configuring GeoMessage component” on page 95 for a detailed description of how to complete this configuration.

5.4.1.3 Configuring GeoMessage

Go back to the main menu for HAGEO and select **Manage GeoMessage** and next **Configure GeoMessage**. As in the previous scenarios, we are going to import the HACMP definitions. Select **Import HACMP Definitions**. Then you need to remove the information related only to the local site dallas. Select **Configure GeoMessage Networks** and next **Delete existing network definition**. Select the **local_net** option. Removing the network automatically removes all its adapters. The serial_net network is not imported from HACMP to HAGEO.

You should now use the Show GeoMessage Configuration to verify the configuration that was created. Select **Show GeoMessage Configuration** and then **Show defined interfaces**. The output for this scenario would be:

```
------------------------------------------------------------
Interface Name      Address         Host      Network Link
------------------------------------------------------------
brown_net2        9.3.4.79         brown    geo_net2
brown_net1        100.100.100.111  brown    geo_net1
black_net2        9.3.4.80         black    geo_net2
black_net1        100.100.100.112  black    geo_net1
blue_net2         9.3.4.16         blue     geo_net2
blue_net1         100.100.100.121  blue     geo_net1
```

5.4.1.4 Synchronizing the Configuration

Go back to the Manage HAGEO Sites/Machines menu. Select **Sync Site/Machine Definitions to all Machines**. Remember that all the names of the machines in the cluster must be resolved into an IP address (through /etc/hosts or DNS) associated to one of the geographical network interface and the /.rhosts file must be configured for the synchronization operation to work properly. Next go to the Configure GeoMessage menu and select Distribute GeoMessage Definitions to Remote Machines.

You now need to start GeoMessage in all nodes prior to configuring the GeoMirror devices. Go back to the Manage GeoMessage menu and select **Start GeoMessage**. You can also use the command:

```
# /usr/sbin/krpc/cfgkrpc -ci
```

5.4.1.5 Configuring GeoMirror Devices

In this scenario, the filesystem /data is going to be mirrored across the geography. Also the application server geodemo will be shared between the sites. Four GeoMirror devices need to be configured. At the dallas site, all the logical volumes needed for the GMDs must be located on shared disks if the filesystem and/or the application server are to be taken over locally.

It was chosen to have two volume groups per site: datavg and appvg. The former contains the logical volumes data0lv, log1lv, data0sm_21 and log1sm_21 in site austin and in site dallas it holds data0lv, log1lv, data0sm_11, log1sm_11, data0sm_12 and log1sm_12. The volume group appvg contains the logical volumes acc2lv, tr3lv, acc2sm_21 and tr3sm_21 in site austin and acc2lv, tr3lv, acc2sm_11, tr3sm_11, acc2sm_12 and tr3sm_12 in site dallas. The volume groups names do not need to be the same across the geography. They were configured in this way only for naming convention purposes. In site austin the logical volumes could even be located in rootvg volume group.

First vary on the volume groups and create the logical volumes in blue. In site dallas the logical volumes configuration only needs to be done in one node. It will later be imported to the other node. Vary on the volume groups in node brown for instance, and create the logical volumes. Then vary on the volume groups in blue and define the logical volumes. Notice that the logical volume sizes must be the same. For the state maps, the names need to be different and the remaining logical volumes should have the same name.

After creating the logical volumes, you need to go back to SMIT to configure the GMD devices. Enter `smit hageo`. Then select **Manage GeoMirror** followed by **Configure a GeoMirror Device**. Next choose **Add a GeoMirror Device**. Then add the GMDs data0gmd, log1gmd, acc2gmd and tr3gmd in nodes brown and blue. For a detailed description of this configuration, refer to Section 5.1.4, “Configuring GeoMirror devices (GMDs)” on page 104.

After creating these GMD devices, you are able to create the /data filesystem. First start data0gmd and log1gmd in node brown with the commands:

```
# /etc/methods/cfggmd -A
# /etc/methods/gmddown -A blue
# /etc/methods/startgmd -A
Then start the GMDs in node blue:
# /etc/methods/cfggmd -A
# /etc/methods/startgmd -A
```

If you try to start the GMDs and one of the nodes is powered-off or inaccessible through TCP/IP, the GMDs are going to take a long time to start.

Then edit /etc/filesystems file in all the nodes (including black) and add the following entry:

```
/data:
    dev         = /dev/data0gmd
```

```
vfs         = jfs
log         = /dev/log1gmd
mount       = false
check       = false
options     = rw
account     = false
```

Notice that the dev and log fields point to the GeoMirror device names.

Next create the mount point for the filesystem /data in all the nodes (including black) with the command `mkdir /data`. Then only on the local machine execute the command:

mkfs -l data /dev/data0gmd

At this point, writes will occur at both sites to make the filesystem. You may want to execute the `fsck` command on the newly created filesystem. Then mount the filesystem on the local node.

After creating the filesystem you are ready to import the configuration to node black. First unmount /data in node brown, stop the GMDs and vary off the volume groups. Then import both volume groups to node black using the `importvg` command. Remember that the volume groups should not be automatically varied on at boot time. This is the default option when you use `importvg` and you will need to alter this parameter to no .

Next create data0gmd, log1gmd, acc2gmd and tr3gmd in node black. Start the GMDs. You could use the following commands:

```
# /etc/methods/cfggmd -A
# /etc/methods/gmddown -A blue
# /etc/methods/startgmd -A
```

After the GMDs are started, try to mount the /data filesystem. At this point you are all set to create the application server.

5.4.1.6 Creating the Application Server

HAGEO comes with a demo application that simulates a banking application, where account data is mirrored across geographic locations to provide data redundancy in the event of a catastrophic failure that causes one site to fail and the other site to takeover. The demo consists of a server and client portion and resides in the /usr/sbin/gmd/demos directory. It runs by directly using raw GMDs as I/O devices, not through a database. We are going to use this demo as our application server.

The server should be run on nodes with GMDs that are available. All cluster events and I/O to the GMDs are processed by the server. The server should be started on all nodes that will access the GMDs when the GMDs are available. The application server will start the server in one of

the nodes. In the other node, the server has to be started manually. The objective of starting the server on both nodes is to show the online mirroring that occurs between the local and remote GMDs.

For asynchronous GMDs, only the primary node should start the server since the secondary node will be unable to access the data. The data will continue to be mirrored, but it will not be readable. When a failover occurs, the backup site should start the server when roles have been switched and the GMDs are available.

You can run the Geo Demo client in any Motif environment and on any machine serving as a cluster client, on any node running the Geo Demo Server, or on any geo_node in general. Do not, however, run the demo client on a node being failed. A node that uses HACMP to manage the GMDs will work as long as the node is not crashed or re-booted. Also, remember that only one client can write to a GMD at a time, but all clients can read from both sites.

The client application allows you to attach to a node at each site. When attached, you will see the account data for the application. You can then select to view the transaction data for any account by clicking on that account. I/O can be started to either node that has its GMDs available, but I/O can occur on only one node at a time. During the specified time interval for I/O, you can query the server for the current data at any time.

Important

Do not run this demo on a GMD that contains real data you want to maintain or preserve; the demo overwrites all data in the GMD as it runs.

The following is the start script for the application server geodemo:

```
#! /bin/sh
LOG=/tmp/geodemo.out
if [[ -f /tmp/geoinit ]]
   then /usr/sbin/gmd/demos/demo_server /dev/racc2gmd /dev/rtr3gmd
1>>$LOG &
   else /usr/sbin/gmd/demos/demo_server -i /dev/racc2gmd
/dev/rtr3gmd 1>>$LOG &
fi
touch /tmp/geoinit
exit
Alter the script to reflect the names of the GMDs you created. The
following is the stop script for the application server geodemo:
#! /bin/sh
usage()
{
```

```
    /bin/echo "Usage: stop_imagedemo [ -a Service_Address ]"
    exit 2
}
PROGNAME="$0"
SERVICE_ADDRESS=""
# Turn on debugging
if [ "$VERBOSE_LOGGING" = "high" ]
then
    set -x
fi
# Get command line options
set -- `getopt "a:" $*`
if [ $? -ne 0 ]
then
    usage
fi
# Parse command line.
while [ $1 != -- ]
do
    case $1 in
    -a)
        IP_LABEL="$2"
        #
        # Convert ip label to dot address
        # Doesn't matter if the address is already in dot format
        #
        SERVICE_ADDRESS=`host $IP_LABEL | cut -d' ' -f3 | sed s/,//g`
        shift ; shift
        ;;
    *)
        usage
        ;;
    esac
done
shift   # lose the --
set -u
#  Get the pid (with the given Service Address)
if [ "$SERVICE_ADDRESS" = "" ]
then
    PID=`ps auxww | egrep -e "demo_server" | grep -v egrep | awk -F' '
'{print $2}'`
else
    PID=`ps auxww | egrep -e "demo_server" | grep -v egrep | grep
$SERVICE_ADDRESS | awk -F' ' '{print $2}'`
fi
if [ -n "$PID" ]
then
```

```
     kill -9 $PID
fi
exit 0
```

You need to define the geodemo application server to HACMP, using the start and stop scripts given above.

You can also use these scripts to start and stop the Geo Demo server on the other nodes. However, the first time you run the script, it reinitializes the data. So before manually starting the server in one of the nodes, run the command:

touch /tmp/geoinit

This command will prevent the data from being reinitialized when you use the start script in other nodes. This command only needs to be run once, unless you remove the /tmp/geoninit file.

To start a demo client, enter the following command:

/usr/sbin/gmd/demos/demo_client

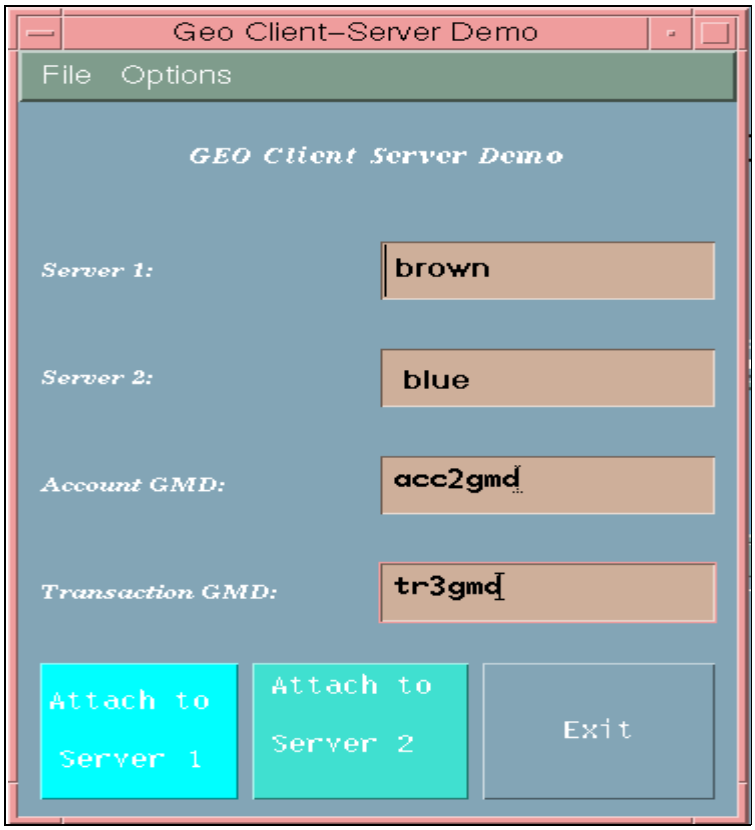

Figure 86. Geo Client/Server Demo screen

Figure 86 shows the window that is presented after you start the Geo Demo client. This window contains several fields that require you to define the servers in the geographic cluster. The servers contain the GMDs and should have the Geo Demo server started. The Account GMD and Transaction GMD fields in this window also require that you enter the GMD names matching the ones you used when starting the server.

The Geo Client-Server Demo window also lets you define how the demo operates. By selecting **Change Options** from the Options menu, you can define in the pop-up Demo Options window time intervals (in seconds) for performing I/O, how long the demo waits between I/Os, and the number of queries to be performed on the GMDs.

Note

If a value other than zero is specified in the Query time field, access to updated data is restricted on the server that did not initiate the I/O, and the window is updated on the initiating server only after pressing the **Query** Server button in the Geo Client-Server Accounts window.

To attach to servers at the local and remote site, click on the **Attach to Server 1** and **Attach to Server 2** button at the bottom of the Geo Client-Server Accounts window. Clicking on these buttons causes the demo to attach to the defined server and to display account information in pop-up Geo Demo - Accounts windows. These windows contains several buttons that let you query the server for updated account information, start writes to a server and close the Geo Demo - Accounts window. Clicking on the **Start I/O** button lets you view GMD writes as they occur.

To start I/O to a GMD, click on the **Start I/O** button in the bottom of the Geo Client-Server Accounts window. A Geo Demo - Activity window appears, requiring that you click on the Start I/O button in this window to start sending writes to the server and begin viewing them. Writes to the GMD continue until the specified time interval for I/O elapses. Click on the **Close Window** button to return to the Geo Demo - Accounts window.

Note

When viewing writes to synchronous GMDs, a minimal lag time occurs as updates are made across geographic sites, allowing you to view changes in data almost instantaneously. Viewing writes to asynchronous GMDs at the local site, however, restricts you from viewing data updates at the remote site.

To see updates to account information, click on the **Query Server** button in the Geo Demo - Accounts window. If account data has changed, the window is refreshed to indicate the current dollar values for all accounts. If you want to view data for a specific account, click on the **person's name**; a Geo Demo - Transactions window appears showing the most recent deposits to and withdrawals from the account.

5.4.1.7 Creating Resource Groups

For the proposed configuration, four resource groups are needed. First create the resource groups associated with the volume groups. In this way HACMP will be able to start the GMDs that are located in volume groups other than rootvg. The resource groups configuration for volume groups is

the same as in scenario 2. The only difference is that the volume groups are now named datavg and appvg instead of smvg and geovg.

Next configure the resource group that will handle the critical filesystem /data. The definition is almost the same as the fsrg resource group in scenario 2. What differs is that this time site austin has a higher priority over the /data filesystem. In the participating node names field you should complete with austin followed by dallas.

Then configure the resource group that will handle the application server geodemo. Follows its description:

```
            Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Resource Group Name                                 fsrg
  Node Relationship                                   cascading
  Participating Node Names                            dallas austin

  Service IP label                                    []                +
  Filesystems                                         []                +
  Filesystems to Export                               []                +
  Filesystems to NFS mount                            []                +
  Volume Groups                                       []                +
  Concurrent Volume groups                            []                +
  Raw Disk PVIDs                                      []                +
  Application Servers                                 [geodemo]         +
  Miscellaneous Data                                  [acc2gmd tr3gmd]

  Inactive Takeover Activated                         false             +
  9333 Disk Fencing Activated                         false             +
  SSA Disk Fencing Activated                          false             +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

As stated before, for resource groups that include application servers or filesystems, the site names should be specified and the GMDs used by the application server should be given in the Miscellaneous Data field. For the application server geodemo, site dallas has the highest priority.

Remember to synchronize the resource group configuration to all nodes.

After adding the last two resource groups, the `clverify` utility will give an error because it does not know how to handle sites. You may ignore this error.

5.4.1.8 Configuring DBFS

Configure the dbfs.envfile in the nondominant site as explained in Section 5.2, “Dial Back Fail Safe (DBFS)” on page 123.

5.4.1.9 Verifying the configuration

Now you need to verify the HAGEO configuration. Enter `smit hageo` and select **Verify HAGEO Configuration**.

You can also verify HAGEO configuration executing the following command:

```
# /usr/sbin/gmd/geo_verify
```

5.4.1.10 Integrating HACMP and HAGEO

Execute in each node the following command:

```
# /usr/sbin/gmd/scripts/configure_events <machine_name>
```

This command will plug the GMD events into HACMP. The next time HACMP is started, the GMD devices will come on automatically. You will need to reboot the nodes at this time.

5.4.1.11 Starting HACMP

For some of the more complex failures HAGEO handles, `clinfo` needs to be in trap mode. To achieve this you need to run in both machines the following command:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

Prior to starting HACMP, you need to release the resources that are handled by HACMP. Unmount the /data filesystem and stop all instances of the Geo Demo Server that you might have started. Then stop the GMDs with the commands:

```
# /etc/methods/stopgmd -A
# /etc/methods/ucfggmd -A
```

Then unload GeoMessage:

```
# /usr/sbin/krpc/cfgkrpc -u
```

Finally vary off the volume groups datavg and appvg.

Now start HACMP by executing `smit clstart`. Change the Startup Cluster Lock Services and Startup Cluster Information Daemon options to yes.

You need to start HACMP in all the nodes. After HACMP startup is finished, the GeoMirror devices should have been started in nodes brown and blue and the filesystem /data mounted in node brown.

5.4.2 Testing the Cluster

Besides running `clverify` and `geo_verify` utilities, you also need to run tests in the cluster to make sure that the configuration will behave as desired. First make sure that after starting HACMP in both nodes, the GMDs were started in nodes brown and blue, that the geodemo application server was initialized in brown and the filesystem /data was mounted in blue. At this time you are able to begin testing the cluster.

The same tests performed in scenario 2, were utilized to test the scenario 3 cluster. The behavior of the cluster in most cases is similar to scenario 2. The tests brown unavailable, site dallas disaster and reintegrating brown have exactly the same sequence of events. The only difference is that instead of mounting/mounting /data, HACMP starts/stops the geodemo application server.

The test site isolation from brown using `ifconfig` has the same sequence of events as in scenario 2. blue is halted again, but in scenario 3 this node currently owns the resource /data filesystem. In the previous scenario blue did not own any resource. brown does not mount this filesystem when blue is halted. When brown is running the node_down blue event, it uses the `geo_info` command to check the status of the remote GMDs. The following error occurs:

```
: RPC: Port mapper failure - RPC: Unable to send
: RPC: Port mapper failure - RPC: Unable to send
GMD status: No route to host
```

The command is aborted with no route to host and exit status `0` and the node_down event is completed without brown taking over the resources. There is a HAGEO problem when you `ifconfig down` the interfaces. HAGEO uses these adapters to communicate with some local daemons. So when the interfaces are down, problems with some utilities that need to communicate with local daemons will occur. This is why `geo_info` fails.

Testing site isolation from brown by pulling out the cables has almost the same sequence of events as in scenario 2. brown stops geo_demo server instead of unmounting /data. But this time brown is also able to stop the GMDs and start the node_down_local event. The volume groups are varied off. At this point black start the node_down brown event and takes over the critical resources. This event never happened in the same test in scenario 2.

The site isolation from blue has a behavior very similar to scenario 2. The difference is that now blue owns the /data filesystem. During the execution of

the event node_down blue in brown, it issues the `gmddown` command in the GMDs for the remote node blue and mounts the filesystem /data.

For this scenario two more tests were necessary. As site austin in the current case owns a resource, there is a need to test the behavior of the cluster when this site becomes unavailable and also what happens when it is reintegrated. The following is a description of these tests:

5.4.2.1 Austin site disaster

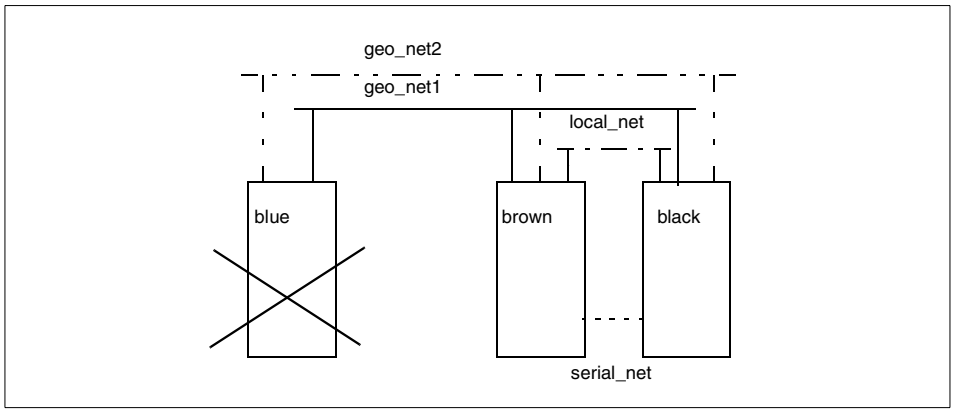

Figure 87. Austin site configuration

In order to simulate a site austin disaster, you can power-off blue or stop HACMP with the takeover option. The following list details the sequence of events in brown when blue becomes unavailable:

```
EVENT START: node_down blue
Starting Geo_stop_gmds ...
Starting Geo_node_down ...
Starting Geo_remote_peer_down ...
Starting Geo_notify ...
/etc/methods/gmddown -l acc2gmd blue
/etc/methods/gmddown -l data0gmd blue
/etc/methods/gmddown -l log1gmd blue
/etc/methods/gmddown -l tr3gmd blue
Starting Geo_mount_fs ...
fsck -f /data
mount /data
GEO Filesystem /data mounted
EVENT START: start_server geodemo
EVENT COMPLETED: start_server geodemo
Geo_node_down blue exiting with STATUS 0
```

```
EVENT COMPLETED: node_down blue
EVENT START: node_down_complete blue
EVENT COMPLETED: node_down_complete blue
```

brown issues the `gmddown` command on the GMDs to which blue is a remote peer, in order to stop the mirroring across the geography. Then brown mounts the /data filesystem.

5.4.2.2 Reintegrating austin

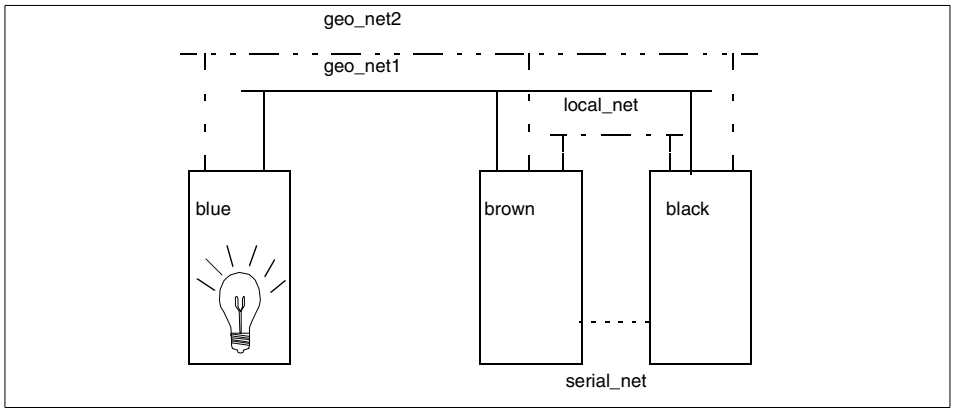

Figure 88. Reintegrating site austin

Now start HACMP in blue to bring it back to the cluster. The sequence of events in brown follows:

```
EVENT START: node_up blue
Starting Geo_remote_node_up
Geo_remote_node_up complete
Starting Geo_start_gmds
Starting Geo_start_server...
Starting Geo_remote_join ...
sync
sync
umount /data
GEO Filesystem /data unmounted
Geo_start_gmds complete
EVENT COMPLETED: node_up blue
EVENT START: node_up_complete blue
Starting Geo_shared_remote_dev...
Geo_shared_remote_dev complete
EVENT COMPLETED: node_up_complete blue
```

brown detects that blue is back in the cluster, so it unmounts /data filesystem. This is what happens in blue:

```
EVENT START: node_up blue
Starting Geo_remote_node_up
Geo_remote_node_up complete
EVENT START: node_up_local
EVENT START: get_disk_vg_fs appvg datavg
varyonvg -n appvg
varyonvg -n datavg
EVENT COMPLETED: get_disk_vg_fs appvg datavg
EVENT COMPLETED: node_up_local
Starting Geo_start_gmds
Starting Geo_start_server...
/etc/methods/cfggmd -l acc2gmd
cfggmd: gmd, kmid = 28606324 loaded.
/etc/methods/gmddown -l acc2gmd black
/etc/methods/startgmd -l acc2gmd
startgmd: acc2gmd started successfully.
/etc/methods/cfggmd -l data0gmd
/etc/methods/gmddown -l data0gmd black
/etc/methods/startgmd -l data0gmd
startgmd: data0gmd started successfully.
/etc/methods/cfggmd -l log1gmd
/etc/methods/startgmd -l log1gmd
startgmd: log1gmd started successfully.
/etc/methods/cfggmd -l tr3gmd
/etc/methods/gmddown -l tr3gmd black
/etc/methods/startgmd -l tr3gmd
startgmd: tr3gmd started successfully.
Starting Geo_mount_fs ...
fsck -f /data
mount /data
GEO Filesystem /data mounted
Geo_start_gmds complete
EVENT COMPLETED: node_up blue
EVENT START: node_up_complete blue
Starting Geo_shared_remote_dev...
Geo_shared_remote_dev complete
EVENT START: node_up_local_complete
EVENT COMPLETED: node_up_local_complete
EVENT COMPLETED: node_up_complete blue
```

blue varies on the volume groups datavg and appvg, starts the GMDs, and mounts the filesystem /data.

Chapter 6. Administration

In this chapter, we describe the methods used to stop and start the different components of GeoRM/HAGEO software, and we offer some procedures to help you administrate a cluster when file system size is increasing, verify a configuration, and so on.

You will find information about how to back up or upgrade your GeoRM/HAGEO configuration, and we describe the various daily tasks required to monitor a GeoRM/HAGEO configuration.

6.1 Starting and stopping a GeoRM configuration

To upgrade the operating system or modify your GeoRM/HAGEO configuration, you must start and stop the individual GeoRM components in a specific order on each machine to start and stop the entire GeoRM configuration.

6.1.1 Starting the GeoRM configuration

Starting the GeoRM configuration includes starting GeoMessage, starting the GeoMirror Devices, and starting the application(s) on each machine. You must start GeoMessage prior to starting the GeoMirror devices.

The local and remote parts of a GeoMirror device cannot be started simultaneously.Therefore, you must determine the order for starting the GeoRM machines. We advise that you start the local/remote machines in the reverse order that they were stopped. For example, if GMD1 was stopped on machine b and then a, it should be started on machine a and then b. If there are any staleness markings (use the `gmd_show_state` command to determine this) on either the local or remote machine, that machine should be started before the other.

For a specific GMD, you must wait for it to start completely (AVAILABLE state) on either the local or remote side, and then start it on the other. However, you can start different GMDs simultaneously on different machines. For example, you can start GMD1 on machine a and GMD2 on machine b at the same time.

Once you determine the starting order of your machines, perform the following procedures on each machine to start the GeoRM configuration:

1. If the volume groups are offline, vary them on for the GeoMirror devices.
2. Start GeoMessage.

3. Start each GeoMirror device.
4. Mount any file systems that reside on GeoMirror devices.
5. Start the GeoMirror application(s).

6.1.2 Stopping the GeoRM Configuration

You must stop the GeoRM configuration when doing software upgrades, hardware changes (controllers, interfaces, disks), or when making local and remote configuration changes to GeoMessage, GeoManager, or a GeoMirror device. You do not have to stop the entire configuration when making configuration changes to individual GeoMirror devices. For example, devices mirroring application A can remain running while you change devices used by application B.

You must stop the GeoMirror application(s) prior to stopping the GeoMirror devices. You cannot stop a GeoMirror device if it is in use. GeoMessage is stopped last.

You should stop the machine owning the primary GeoMirror device(s) first. This will ensure that the primary statemaps are clean prior to stopping. You must wait until one side of the GMD is completely stopped (DEFINED state) before stopping the other side. Follow these procedures on each machine to stop the GeoRM configuration.

1. Stop the GeoMirror application(s).
2. Unmount any file systems that reside on GeoMirror devices.
3. Stop each GeoMirror device. When the last GeoMirror device is unconfigured, the GMD driver is unloaded.
4. Stop GeoMessage.
5. Vary off the volume groups for the GeoMirror devices.

6.2 Starting and stopping an HAGEO cluster with HACMP

When your HAGEO software is integrated with HACMP, you normally use the regular HACMP commands for starting and stopping an HAGEO cluster. When you execute the HACMP `clstart` command, it starts the HACMP daemons and calls the Geo_start_server pre-event script. This script starts the GeoMessage and GeoMirror components (in that order). Then, the HACMP start_server script starts the applications.

In HAGEO configuration, you must first start the machine on the dominant site. If you have more than one machine on the dominant site, you must start the machine having the highest priority first.

For some of the more complex failures that HAGEO handles, `clinfo` needs to be in trap mode. To achieve this, you must run the following command on all nodes:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

We advise that you follow the steps for starting the `/tmp/hacmp.out` HACMP log file with the following command:

```
# tail -f /tmp/hacmp.out
```

Do not start HACMP simultaneously on different machines; before you start the second machine, you must wait until the first machine has completely finished starting. You must obtain the following message in the `/tmp/hacmp.out` HACMP log file:

```
EVENT COMPLETED: node_up_complete machine_name
```

6.2.1 Stopping the HAGEO configuration

When you execute the HACMP `clstop` command, the HACMP stop_servers script calls the Geo_stop_servers post-event script to stop the GeoMirror and GeoMessage components (in that order). You cannot stop a GeoMirror device if it is in use. You must stop the machines in the reverse order that they were started.

Note

The HACMP Cluster Manager will process the config_too_long event if the process of starting several GeoMirror devices exceeds six minutes. You may see the message displayed, but you can ignore it. The Cluster Manager will continue the start process.

Once the configuration is set and HACMP is started, the HAGEO components should only be stopped and started by the HACMP software. Stopping HACMP on the node will automatically stop the GeoMirror devices. Restarting HACMP restarts and synchronizes the GeoMirror devices.

We advise you to follow the different steps of stopping in the `/tmp/hacmp.out` HACMP log file with the following command:

```
# tail -f /tmp/hacmp.out
```

Do not stop HACMP simultaneously on different machines. You must wait until the first machine is completely finished its stopping before to stop the second machine. You must obtain the following message in the `/tmp/hacmp.out` HACMP log file:

```
EVENT COMPLETED: node_down_complete machine_name
```

6.3 Starting and stopping individual GeoRM/HAGEO components

This section describes the different ways of starting and stopping the GeoMessage and GeoMirror devices.

You can use the SMIT interface or the GeoRM commands to start and stop GeoMessage.

You can use the SMIT interface, GeoRM commands, or AIX commands to start and stop the GeoMirror devices. This section describes administrative procedures for:

- Starting and stopping GeoMessage
- Starting and stopping a GeoMirror device.

6.3.1 Starting and stopping GeoMessage

The GeoMessage subsystem must be started before you start a GeoMirror device.

You can use `smit` to start GeoMessages:

```
smit georm or smit hageo
```

The Manage GeoMessage menu shown in Figure 89 on page 177 appears.

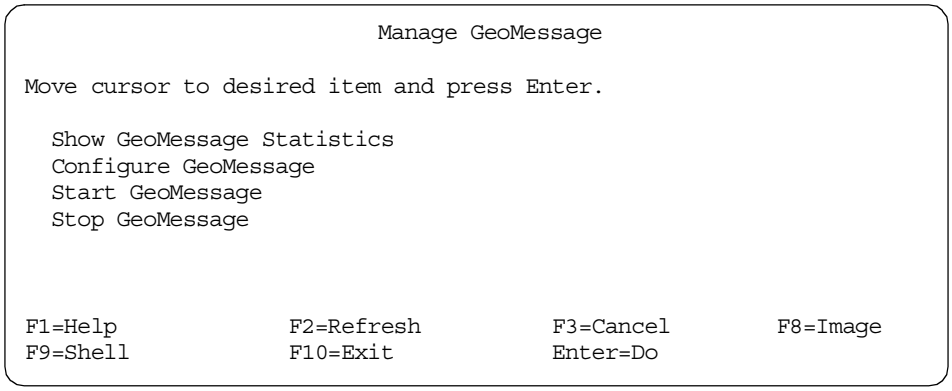

Figure 89. Manage Geomessage menu

Choose the **Start GeoMessage** option, and press **Enter**.

You can also use the command line by entering the following command:

```
#/usr/sbin/krpc/cfgkrpc -ci
cfgkrpc: loadit: extension=/usr/sbin/krpc/krpc kmid=19107104
```

If the above message is returned, GeoMessage has been started. You may, however, receive the following message:

```
cfgkrpc: extension=/usr/sbin/krpc/krpc already loaded
kmid=18950176initkrpc: krpc_kext_init: failed to initialize extension,
unloading: Operation already in progress

cfgkrpc: KRPC initializing failed.
```

This message means the GeoMessage is already started; so the KRPC kernel extension has already been loaded into the kernel.

You can check whether the KRPC kernel extension is loaded into the kernel with the following command:

```
#genkex|grep krpc|grep sbin
          11f7820            264e8 /usr/sbin/krpc/krpc
```

If the return from the command is null, it means that GeoMessage is not started. This command is included in the *perfagent.tools* fileset, which is part of the Performance Toolbox for AIX.

6.3.1.1 Stopping GeoMessage

You cannot stop GeoMessage if an application is currently using it. All the GeoMirror devices on this machine must be in the DEFINED state. See Section 6.3.2.2, “Stopping a GeoMirror device” on page 182.

Note

The GeoMessage kernel extension may still appear to be loaded even after being stopped when viewing the AIX loaded kernel extensions. This will occur if other kernel extensions are loaded after GeoMessage.

You can use smit to start GeoMessages. Enter `smit georm` or `smit hageo`.

The Manage GeoMessage menu shown in Figure 90 appears.

```
                        Manage GeoMessage

Move cursor to desired item and press Enter.

  Show GeoMessage Statistics
  Configure GeoMessage
  Start GeoMessage
  Stop GeoMessage



F1=Help             F2=Refresh          F3=Cancel        F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 90. Manage GeoMessage smit screen

Choose the **Stop GeoMessage** option, and press **Enter**.

You can also use the command line by entering the following command:

```
#/usr/sbin/krpc/cfgkrpc -u
initkrpc: kmid = 18950176 unloaded
```

If, instead of the above message, you receive the message

```
initkrpc: 1 sessions still active; can't unload.
cfgkrpc: KRPC unload failed.
```

there is at least one available (running) GMD in the node and, therefore, the kernel extension cannot be unloaded.

6.3.2 GeoMirror device

GeoMirror devices can be in one of three states: DEFINED, STOPPED, or AVAILABLE. In the DEFINED state, the device exists in the ODM but is neither configured nor available for use in the system. In the STOPPED state, the device is configured, but not available for use by applications. The device is AVAILABLE when it is configured and available for use by applications.

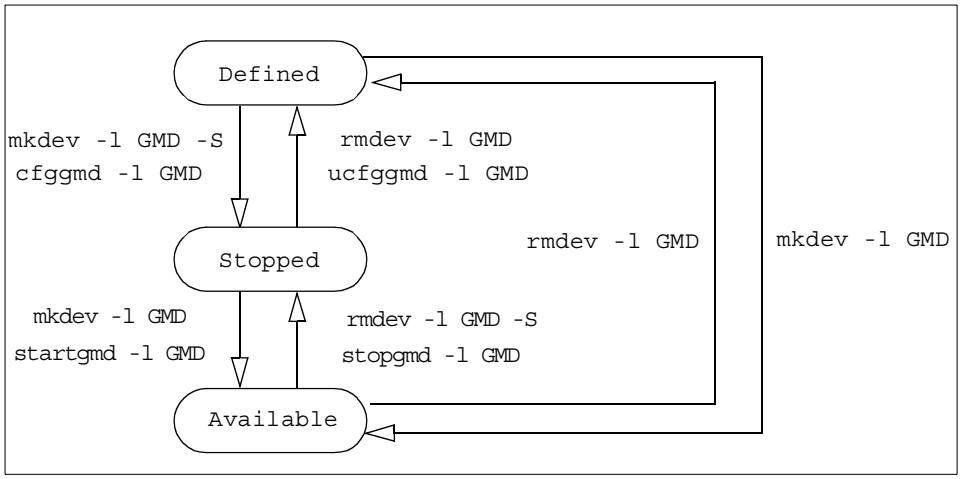

Figure 91. State transition diagram for GMDs

When you create a GeoMirror device, it is created in the STOPPED state.

The device must be in the DEFINED state in order to make any configuration changes to it. You may need to stop and restart a GeoMirror device while making local configuration changes to a database subsystem.

To obtain the current status of all GMDs, you can use the `lsdev` command as follows:

```
#lsdev -Cc geo_mirror
alpha20gmd Defined  Geographic Mirror Device
log21gmd Defined  Geographic Mirror Device
```

Since we have two GMDs, we will use two different methods to start and stop the GMD. We will see that the messages returned by the commands are different.

You can also use the SMIT interface. Enter the following:

```
smit georm or smit hageo
```

The Manage GeoMirror menu is shown in Figure .

```
                    Manage GeoMirror

Move cursor to desired item and press Enter.

  List All GeoMirror Devices
  Configure a GeoMirror Device
  GeoMirror Utilities
  Start a GeoMirror Device
  Stop a GeoMirror Device



F1=Help             F2=Refresh          F3=Cancel        F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 92. Manage GeoMirror menu

Choose the **List All GeoMirror Devices** option, and press **Enter**.

6.3.2.1 Starting a GeoMirror device

With GeoRM and HAGEO (mode asynchronous only), one side of the device must be primary, and the other must be secondary. If the role is set locally to primary and the remote device is set to primary and currently available, the device will not start, and an error will be returned.

If you start the secondary before the primary and the primary has previously been doing staleness marking, you will be unable to start the primary.

Only one of the remote peers can be active at one time for any GeoMirror device. An error will be returned if more than one remote peer is active when starting a GeoMirror device.

A local device cannot start if a remote peer is in the process of starting. In this case, an error will be returned.

The starting of a GeoMirror device is executed in two steps:

1. To configure a GMD, that is, change the status from DEFINED to STOPPED, you can use the `mkdev` command or the `cfggmd` command as follows:

   ```
   #mkdev -l alpha20gmd -S
   alpha20gmd Stopped
   or
   #/etc/methods/cfggmd -l log21gmd
   ```

The -A option of the cfggmd command can be used in place of -l <gmd_name> to configure all GMDs.

When the first GMD is configured, the KRPC kernel extension is automatically loaded from the kernel.

The command checks the mode (async, mwc, or sync) of a device before it configures it into the kernel. The command then requests the mode of the remote peer. If the modes do not match, the command fails.

You can verify the state of GMDs with the lsdev command as follows:

```
#lsdev -Cc geo_mirror
alpha20gmd Stopped  Geographic Mirror Device
log21gmd Stopped  Geographic Mirror Device
```

2. To start a GMD, that is, to change the status from STOPPED to AVAILABLE, you can use the mkdev command or the startgmd command as follows:

```
#mkdev -l alpha20gmd
alpha20gmd Available
```

or

```
#/etc/methods/startgmd -l log21gmd
startgmd: No remote peers currently available for log21gmd.
startgmd: log21gmd started successfully.
```

The above message returned by the startgmd command informs you that the remote peer is not available and that you must use the gmddown command.

The -A option of the startgmd command can be used in place of -l <gmd_name> to start all GMDs.

You can verify that the GMDs are started with the lsdev command:

```
#lsdev -Cc geo_mirror
alpha20gmd Available  Geographic Mirror Device
log21gmd Available  Geographic Mirror Device
```

If the remote peer is not available, when you start a GMD, you obtain the following message:

```
#mkdev -l alpha20gmd
Method error (/usr/lib/methods/startgmd):
        0514-046 A file containing microcode or adapter software was
                 not accessable.
startgmd: Can't start alpha20gmd, remote peer is not in available state.
```

This message means that the remote peer for the GMD you are starting is down or not reachable. In this case, you must use the `gmddown` command to inform the local GMD that the remote device is down:

```
#/etc/methods/gmddown -l alpha20gmd delta
```

The device must be in the STOPPED state to execute `gmddown`.

- A GMD will not start if the `gmddown` command is not executed on the starting machine when the remote machine is up but the remote GMD is not AVAILABLE.
- A GMD will not start if the `gmddown` command is not executed and the remote machine is not reachable due to a geographic network error.

You can use the -A option in place of `<-l gmd>` if all defined remote devices are down.

If you are defined more than one remote peer, you must execute the `gmddown` command for each remote peer before starting GMD.

Later, when you start the GMDs on the remote node, you should not issue the `gmddown` command to the local node. The process of starting the GMDs on the remote node will be fast anyway because the GMDs on the local node are already started and ready for communication.

If the `gmddown` command is issued and the remote device is still available, the remote device is stopped to protect against data divergence. If the `gmddown` command was issued erroneously (the remote device was not down) and you use `smit` or the `lsdev -Cc geo_mirror` command to display the device status, the device status will be listed as Available. The `lsdev -C` command displays device information from the Device Configuration Database. The Device Configuration Database is not updated when the device is brought off line internally. If the command is issued by mistake, you must first stop and then restart the device on the remote node to return the device to service.

If one of the nodes that can take over the GMD is powered off or is completely inaccessible through TCP/IP, the GMDs will take a long time to start anyway, even if you issue the `gmddown` command.

6.3.2.2 Stopping a GeoMirror device

To stop a GeoMirror device, perform the following steps:

1. Stop the application that resides on the file systems defined on the GMD that you want stop.

2. Unmount the file systems. If you have some problems unmounting them, you can use the `fuser` command to see and kill the process using the file system.

   - To see the process using a file system, issue: `fuser -u /dev/hd8`
   - To kill the process using a file system, issue: `fuser -k /dev/hd8`

   With the `fuser` command, you can specify the name of the file system or logical volume.

3. To stop a GMD, that is, to change the status from AVAILABLE to STOPPED, you can use the `rmdev` command or the `stopgmd` command as follows:

   ```
   #rmdev -l alpha20gmd -S
   alpha20gmd Stopped
   ```

   or

   ```
   #/etc/methods/stopgmd -l log21gmd
   ```

   The `-A` option of the `stopgmd` command can be used in place of `-l <gmd_name>` to stop all GMDs.

   You can verify the state of GMDs with the `lsdev` command as follows:

   ```
   #lsdev -Cc geo_mirror
   alpha20gmd Stopped  Geographic Mirror Device
   log21gmd Stopped  Geographic Mirror Device
   ```

4. To unconfigure a GMD, that is, change the status from AVAILABLE to DEFINED, you can use the `rmdev` command or the `cfggmd` command as follows:

   ```
   #rmdev -l alpha20gmd
   alpha20gmd Defined
   ```

   or

   ```
   #/etc/methods/ucfggmd -l log21gmd
   ```

   The `-A` option of the `ucfggmd` command can be used in place of `-l <gmd_name>` to unconfigure all GMDs.

   When the last GMD is unconfigured, the KRPC kernel extension is automatically unloaded from the kernel.

```
                    Manage GeoMirror

Move cursor to desired item and press Enter.

  List All GeoMirror Devices
  Configure a GeoMirror Device
  GeoMirror Utilities
  Start a GeoMirror Device
  Stop a GeoMirror Device



F1=Help             F2=Refresh          F3=Cancel        F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 93. Starting the GeoMirror device

Perform the following steps:

1. From the Manage GeoMirror menu shown in Figure on page 180, choose the **Start GeoMirror Device** option, or choose **Stop a GeoMirror Device**.

   SMIT displays the Select Device Name screen.

2. Select the GeoMirror device to be started or stopped, and press **Enter**. A new screen appears.

3. Press the **Tab** key or **F4** to select the desired Device State, and press **Enter**.

6.4 Verifying the GeoRM or HAGEO configuration

The geo_verify utility checks for many possible configuration errors that can cause serious problems at run time. We recommend that you execute the geo_verify utility after any modification of the configuration. You can execute this utility even if the GeoRM configuration is running. This utility checks more information with the HAGEO software than HACMP information.

For each component of the GeoRM and HAGEO software, we give the following information checked by the `geo_verify` utility:

GeoMessage information

- GeoMessage machines are properly listed on machines that share GeoMirror devices.
- GeoMessage interface entries are identical for all machines in the configuration.

- GeoMessage network configuration information is identical on all machines.

GeoManager information

- GEOnode and GEOsite ODM entries exist on all machines that have GeoMirror devices.
- GEOsite entries are identical on all machines.
- All remote peers listed for any GeoMirror device have GEOnode configuration entries.

The following information is only for HAGEO software:

- A backup to the primary HAGEO network exists. This can be another network or Dial-Back-Fail-Safe.

  If Dial-Back-Fail-Safe is configured, the utility invokes the DBFS routine and verifies that a response is received from the remote site.

  If the backup is a serial (or any point-to-point) network, the utility checks that all machines have a connection to every machine that is a remote peer of one of its GeoMirror devices.

Geographic mirror device information

- Each GeoMirror device has the same minor number on the local and remote machines.
- Each GeoMirror device is the same size on the local and remote machines.
- Each GeoMirror device has the same role on the local and remote machines.
- Each GeoMirror device has a state map device.
- State map devices are large enough to support the GeoMirror devices assigned to them.
- The local GeoMirror and state map logical volumes are raw devices.
- Remote peers for any GeoMirror device are all at the same site.

The following information is only for HAGEO software:

- Local peers can access state map devices on their sites.
- If local peers exist on a site, GeoMirror devices are on shared volume groups.

HACMP information

This section only covers HAGEO software:

- The network type, based on the HACMP Network Interface Module, is defined for the HAGEO networks.
- Geo_Primary and Geo_Secondary HACMPnim entries are present in the configuration database.
- HAGEO networks are defined to HACMP as service interfaces.
- IP address takeover is not configured for any HAGEO network. HACMP events that require action by HAGEO call the HAGEO utilities. For all of the following events, the HACMP events call a generic HAGEO pre-event and post-event handler, which takes appropriate action based upon which event is occurring:
  - node_down
  - node_up_local_complete
  - network_down_complete
  - network_up_complete
  - node_down_complete
  - node_up
- GeoMirror devices are defined in volume groups within resource groups.

Geo_verify utility

You can use the SMIT interface or the GeoRM/HAGEO commands to verify the configuration.

Enter `smit georm` or `smit hageo`, and select **Verify GeoRM Configuration**. The Verify GeoRM Configuration menu appears as shown in Figure 94 on page 187.

```
                        Verify GeoRM Configuration

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  GeoMirror device name(s)                              []          +
  Peer name(s)                                          []          +
  Module level                                           all        +
  Output level                                           Full       +
  Output file name                                       []



F1=Help             F2=Refresh          F3=Cancel       F4=List
Esc+5=Reset         F6=Command          F7=Edit         F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 94. Verify GeoRM configuration

If you press Enter, all configuration information will be verified, but, if you want to verify a specific GMD and its remote peer, press **F4** to select them.

You can also use the command line by entering the following command:

```
#/usr/sbin/gmd/geo_verify
```

The command returns a significant amount of information, but, if the configuration is correct, the last line returned must be:

```
No ERRORS:.
```

You can use the `-l` option of the `geo_verify` command to send the output in the file. We advise you to print the output of this command because it contains all the information of the GeoRM/HAGEO configuration, and it can be very useful for restoring the configuration after a system crash.

6.5 Administration of GeoRM and HAGEO

With HAGEO, you need to stop HACMP to modify the different components of the configuration.

6.5.1 GeoMessage machine

For all operations concerning a GeoMessage machine, you must stop applications, GeoMirrors, and GeoMessages on all machines defined on the GeoRm configuration.

6.5.1.1 Adding a new machine

The prerequisite operations are as follows:

- You must have defined the host name and configured the network interfaces.
- The GeoRM software must be installed with the last PTFs.
- The /etc/hosts and /.rhost files must be updated on local and all remote machines. You must execute the `rsh` command with the other remote machines.
- The volume groups for the GMD must be configured.

Perform the following steps:

1. GeoMessage
   a. Stop GeoMirrors and Geomessage components on all machines included in the GeoRM/HAGEO configuration.
   b. Add the GeoMessage machine definition.
   c. Add the GeoMessage network interface(s) definition(s) for this machine.
   d. Distribute the new definition to all machines in the GeoRM configuration.
2. GeoManager
   a. Add this new machine to a GeoRM site definition.
   b. Add this new machine to a GeoRM machine definition.
   c. Synchronize with the other GeoRM machines.
3. Verify the new configuration with the `geo_verify` utility.
4. Create the GeoMirror devices. See Chapter 4, “Installation and configuration” on page 81, for more information.

6.5.1.2 Changing the hostname of a GeoRM machine

If you change the hostname of a GeoRM machine, you must change the GeoRM configuration by performing the following steps:

Modify the hostname of the machine with the `smit hostname` command.

1. GeoMessage
   a. Modify the name of the GeoMessage machine.
   b. Modify the name of the machine for each GeoMessage interface defined for this machine.

c. If the label IP of these interfaces contains the hostname, we advise you to change the label IP as described in Section 6.5.2.1, “Changing the GeoMessage network” on page 190.

d. Distribute the new definitions to all GeoMessage machines in the GeoRM configuration.

2. GeoRM

   a. Change the name of the machine in the GeoRM site where it is defined.

   b. Change the name in the GeoRM machine definition.

   c. Synchronize with the other GeoRM machines.

3. GeoMirror

   If some GMDs have defined this machine as remote peer, you must change the Remote Machine attribute of these GMDs. The GMDs must be stopped to change this attribute.

4. Verify the new configuration with the `geo_verify` utility.

If no errors are reported, we can first start GeoMessage and then the GMDs on all machines.

6.5.1.3 Removing a GeoRM machine

To remove a GeoRM machine, perform the following steps:

1. GeoMirror

   Delete the GMDs on this machine and on all remote peers. For more information, see Section 6.5.3.2, “Removing a GMD” on page 191.

2. GeoManager

   a. Delete the machine definition.

   b. Delete this machine in the GeoRM site definition.

   c. Synchronize Site/Machine to all machines.

3. GeoMessage

   a. Delete the GeoMessage interface(s) of this machine.

   b. Delete the GeoMessage machine.

   c. Distribute the GeoMessage definitions to the remote machines.

4. Verify the new configuration with the `geo_verify` utility.

If no errors are reported, you can first start GeoMessage and then the GMDs on all machines.

You can delete this machine's information in the /etc/hosts and /.rhosts files on all remote machines.

6.5.2 GeoMessage network

The following sections deal with changing and deleting the GeoMessage Network.

6.5.2.1 Changing the GeoMessage network

For all modifications concerning a GeoMessage network, you must stop applications, GeoMirrors, and GeoMessages on all machines defined on the GeoRm configuration before adding this new GeoMessage network.

If you want to change the IP address or the IP label of a network adapter defined in the GeoRM configuration, you must perform the following steps:

Modify the /etc/hosts file on the local machine and all remote machines.

If you change the IP label, you must change the /.rhosts file to add this new IP label to the local machine and all remote machines.

If you change the IP address, use the `smit chginet` command to change the IP address.

1. GeoMessage
   a. Change the label IP or the address IP of the GeoMessage interface using the old IP label or the old IP address.
   b. Distribute the GeoMessage definitions to all remote machines.
2. Verify the new configuration with the `geo_verify` utility.

If no errors are reported, you can first start GeoMessage and then the GMDs on all machines.

6.5.2.2 Delete a GeoMessage network

To delete a GeoMessage network, you must delete all GeoRM machines and GeoMessage interfaces using this GeoMessage network.

6.5.3 GeoMirror device

This section describes the maintenance of GeoMirror devices.

6.5.3.1 Changing a GMD

You can make changes (except to the mode of operation) without stopping the device, but the changes will not take effect until the next time the device is configured.

1. Stop the application and unmount the file system that can reside on the GMD.
2. Stop and unconfigure the GMD. The GMD must be in DEFINED state on the local and remote machines.
3. Change the attributes of the GMD on the local machines and all remote machines where this GMD is defined (in case you have more than one remote peer for this GMD).
4. Verify the new configuration with the `geo_verify` utility.

If no errors are reported, you can start the GMD.

6.5.3.2 Removing a GMD

If a file system is defined on this GMD, you must stop the application and unmount the file system first. For more information about stopping a GMD, see Section 6.3.2.2, “Stopping a GeoMirror device” on page 182. Perform the following steps to remove a GMD:

1. Enter `smit georm` or `smit hageo`
2. Select **Manage GeoMirror -> Configure a GeoMirror Device ->Remove a Geographic Mirror Device**.
3. A pop-up menu appears with the list of GMDs. Select the GMD to delete and press **Enter**. The Remove a GeoMirror Device screen appears as shown in Figure 95 on page 192.

```
                          Remove a GeoMirror Device

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Device Name                                        alpha20gmd
* KEEP definition in database                        yes                  +



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 95. Remove a GeoMirror Device menu

4. If you have chosen to remove the GMD from the system, you can now remove the logical volumes for the data and state map from AIX if so desired.
5. Verify that the new configuration is correct with the `geo_verify` utility.

6.5.4 Increasing the size of a logical volume

If your GMDs only use logical volumes without file systems, the following procedure enables you to increase the size of the logical volumes:

1. On the local machine, stop the GeoRM/HAGEO application that uses this logical volume.
2. On both the local and remote site nodes, stop the GMD on which the file system resides, and put it in a DEFINED state.
3. On the machines on which the logical volumes are defined, use the following command to increase the size of all logical volumes in the GMD:

   ```
   # smit chlv
   ```

4. Verify the new configuration with the `geo_verify` utility.
5. Mark all the cells in the state map of the GMD to *stale* on the local node where it is defined as the primary role (GeoRM or HAGEO mode asynchronous) or the dominant site (HAGEO):

   ```
   # /usr/sbin/gmd/gmddirty -l alpha20gmd
   ```

   You can use the `gmd_show_state` command to verify whether the cells are marked stale:

   ```
   #/usr/sbin/gmd/gmd_show_state -l alpha20gmd
   ```

```
Point of View: Node alpha
-------------
Point of View GMD List:
----------------------
Name: alpha20gmd
Status: STOPPED
State Map:
Cell     Value
-------   -----
      0   0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf
*
1048576
```

6. Configure the GMD on the local node:

```
# /etc/methods/cfggmd -l alpha20gmd
```

7. On the local node, run `gmddown` on the GMD for the remote node(s):

```
# /etc/methods/gmddown -l alpha20gmd delta
```

8. Start the GMD on the local node:

```
# /etc/methods/startgmd -l alpha20gmd
```

9. On the remote node, configure and start the GMD. This will cause the GMD data to synchronize from the local node:

```
# /etc/methods/cfggmd -l alpha20gmd
# /etc/methods/startgmd -l alpha20gmd
```

6.5.5 Increase the size of a file system

If some file systems are defined on your GMDs, the following procedure enables you to increase the size of a file system on which a GeoRM or HAGEO application resides. The application on which the file system resides is offline during this procedure. However, other applications that do not use this file system can still be available.

With HAGEO, you do not need to stop HACMP. You only have to prevent the application from using this file system.

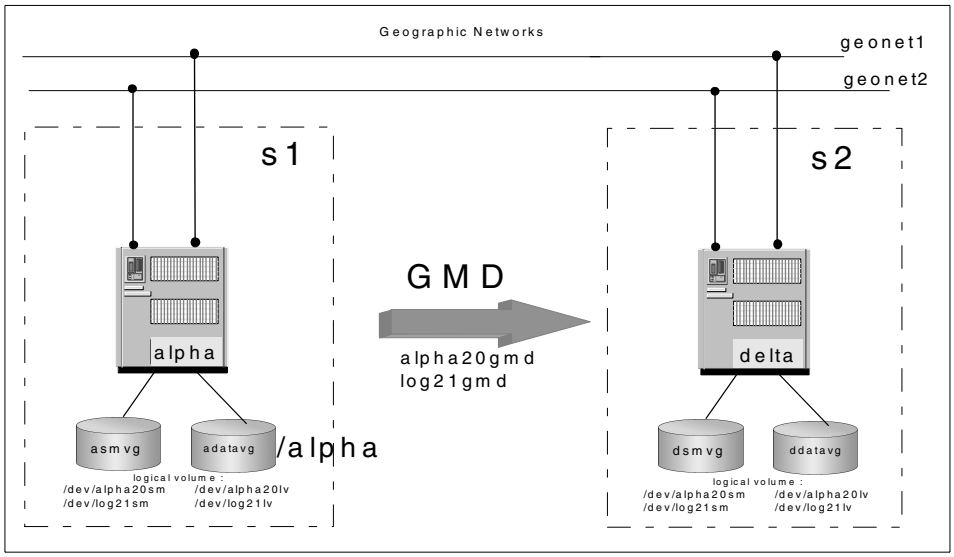

Figure 96. GeoRM configuration

In our case, we want to increase the **/**alpha file system, which resides on alpha machines of 20 pps. The two GMDs are defined as primary on the alpha machine and secondary on the delta machine. Perform the following steps:

1. On the local machine, stop the GeoRM/HAGEO application, which resides on the file system that you want to change.
2. On the local machines, unmount the file system where the GeoRM/HAGEO application resides. If you experience problems unmounting the file systems, you can use the `fuser` command as described in Section 6.3.2.2, “Stopping a GeoMirror device” on page 182.
3. On both the local and remote site nodes, stop the GMD on which the file system resides, and put it in a Defined state:

   ```
   # /etc/methods/stopgmd -l alpha20gmd
   # /etc/methods/ucfggmd -l alpha20gmd
   ```

   The file system log GMD can remain in the available state during this procedure enabling other file systems that use it to remain in service.

   Complete steps 4 through 8 on the local machine first, then the remote machine delta.
4. Change the file system entry in `/etc/filesystems` to point to the non-GMD file system logical volume:

```
# vi /etc/filesystems
/alpha:
        dev             = /dev/alpha20gmd
        vfs             = jfs
        log             = /dev/log21gmd
        mount           = false
        check           = false
        option          = rw
        account         = false
```

In our example, we changed the entry of the GMD to point to the logical volume data.

```
/alpha:
        dev             = /dev/alpha20lv
        vfs             = jfs
        log             = /dev/log21gmd
        mount           = false
        check           = false
        option          = rw
        account         = false
```

5. Change the file system log logical volume info to point to the non-GMD file system log logical volume.

   a. If a no-GMD file system log logical volume does not exist, create one through `smit mklv`. In our example, we have defined `jfslogtmp` as the log logical volume.

   b. Use `logform` to initialize the new logical volume as a jfs log:

      ```
      # /usr/sbin/logform /dev/jfslogtmp
      logform: destroy /dev/jfslogtmp (y)?
      ```

      Answer yes to when it asks you whether to format the new log logical volume.

   c. Change the file system log entry in /etc/filesystems to point to the non-GMD file system log logical volume using `chfs`. This will ensure that the lvcb log entry points to the correct log logical volume.

      ```
      # chfs -a log="/dev/jfslogtmp" /alpha
      ```

6. Increase the size of the file system through `smit`:

   ```
   # smit chfs
   ```

7. Change the file system log logical volume info to point to the GMD file system log logical volume:

```
# chfs -a log="/dev/log21gmd" /alpha
```

8. Change the file system entries in /etc/filesystems to point to the GMD file system logical volume.

```
# vi /etc/filesystems
/alpha:
        dev             = /dev/alpha20gmd
        vfs             = jfs
        log             = /dev/log21gmd
        mount           = false
        check           = false
        option          = rw
        account         = false
```

9. After completing steps 4 through 8, mark all the cells in the state map of the GMD to *stale* on the local node where it is defined as the primary role (GeoRM or HAGEO mode asynchronous) or dominant site (HAGEO).

```
# /usr/sbin/gmd/gmddirty -l alpha20gmd
```

You can use the `gmd_show_state` command to verify that the cells are marked stale.

```
#/usr/sbin/gmd/gmd_show_state -l alpha20gmd
Point of View: Node alpha
--------------
Point of View GMD List:
-----------------------
Name: alpha20gmd
Status: STOPPED
State Map:
Cell     Value
-------   -----
      0   0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf 0xf
*
1048576
```

10. Configure the file system GMD on the local node:

```
# cfggmd -l alpha20gmd
```

11. On the local node, run `gmddown` on the file system GMD for the remote node(s):

```
# /etc/methods/gmddown -l alpha20gmd delta
```

12. Start the file system GMD on the local node:

```
# /etc/methods/startgmd -l alpha20gmd
```

13. On the remote node, configure and start the file system GMD. This will cause the GMD data to synchronize from the local node:

```
# /etc/methods/cfggmd -l alpha20gmd
# /etc/methods/startgmd -l alpha20gmd
```

This operation must take a long time to synchronize the data.

14. If the file system GMD is relatively small, you may wish to verify data integrity on both sites.You must verify that you have enough available space in the /tmp file systems.

    a. Save a copy of the data on the GMDs on the local node:

    ```
    # dd if=/dev/filesystem_gmdname of=/tmp/fsgmd.local
    ```

    b. Save a copy of the data on the GMDs on the remote node:

    ```
    # dd if=/dev/filesystem_gmdname of=/tmp/fsgmd.remote
    ```

    c. Copy the remote file on the local machine.

    d. Verify that the data on the GMDs is identical:

    ```
    # cmp /tmp/fsgmd.local /tmp/fsgmd.remote
    ```

Note

In the case of a three-node HAGEO configuration in which two nodes on a site use a shared disk, you must increase the size on only one of the two machines, and the ODM of the second machine will be updated automatically on the next takeover by an importvg of the volume group that contains this file system.

6.5.6 Adding a new application

This section describes how to add a new application in the GeoRM configuration.

If the application will reside on a new machine not defined in the GeoRM configuration, you must add this new machine as described in Section 6.5.1.1, “Adding a new machine” on page 188, and, afterwards, you can follow these steps:

1. Reboot the system
2. Start GeoManager
3. Create the logical volume for the data and the state map for each GMD. If the application uses file systems, you must create a GMD for the log logical volume. This operation must be executed on the local and remote machines.

4. Create the GMD on the local and remote machines.
5. Verify the GMD with the `geo_verify` utility.
6. Start the GMD.
7. If the application uses file systems, create the mount point on both machines and create the file system. For more information see Chapter 4, “Installation and configuration” on page 81.
8. Start the application.

6.6 Monitoring a GeoRM/HAGEO configuration

The GeoRM/HAGEO and AIX monitoring tools help you design a GeoRM/HAGEO configuration, as well as maintain an existing system.

The following methods are available to monitor a GeoRM/HAGEO configuration:

- Monitoring utilities
- User commands
- Log files

See the AIX documentation for information about utilities, commands, and log files supplied with that product. AIX tools include utilities, such as iostat, the `ls-` commands, such as `lsdev`, and various log files.

The following sections describe the GeoRM/HAGEO utilities, commands, and log files.

It is important to monitor the configuration after it is installed and active. It is critical to know as soon as possible when a failure has occurred. It is best to keep the window of non-mirroring time as short as possible. Recovery time can be very lengthy if there is much data to recover.

It is critical that you monitor all logs at all times. Specifically, the geographic network should be monitored. It cannot be assumed that, because the machines are up at both sites, the data is consistent. For example, an intermittent network error can cause the primary GMD to execute the `gmddown` command for the remote peer on that network and make an entry in the AIX log. Staleness would then begin on the primary GMD. Unless specific action is taken, no mirroring will occur, even when the network comes back up. During this time, the remote machine is still seen as AVAILABLE. If you look at the machine state only and not the AIX error log, you can be misled. In

general, it is recommended that there be procedures in place to monitor and log all machines, networks, GeoMirror devices, and log files.

The GeoRM/HAGEO monitoring utilities work like UNIX statistics monitoring programs (for example, iostat, vmstat, netstat). These utilities provide information that helps you tune your system. Monitor GeoNetworks

6.6.1 Monitor the GeoMessage performance

You can use the `krpcstast` to evaluate Geographic Messaging performance, in the following case, we monitor the performance of beta:

```
# /usr/sbin/krpc/krpcstat -s -m beta

 Machines                          KRPC Statistics
---------- ------------------------------------------------------------
              rpcs       kb        rp    rpcs fragd    frags   % frag
beta             0      0.0         0             0        0     0.00


 Networks                          KRPC Statistics
----------
           ------------------------------------------------------------
           -
              pkts       kb    rsnds    touts     fail     dups      oth
geonet1         57    295.0        0        0        0        0    23242

              pkts       kb    rsnds    touts     fail     dups      oth
geonet2          1      0.3        0        0        0        0        0

rpcs = remote procedure calls, kb = kilobytes of data written, rp =
replies, frag = fragment or fragmented, pkts = packets, rsnds = resends,
touts = timeouts, fail = failures, dups = duplicates, other = pings, acks
```

The preceding output is the result of the `gmdstat` command on a GeoRM configuration. On a HAGEO configuration, the output also contains HAGEO-specific information.

The `krpcstat` command produces the following information:

- The amount of data sent and received per network since the last time interval measured in KB, KBps, RPCs and RPCs/sec. The number of RPCs reported includes all fragmented RPCs but excludes resends.
- The degree of RPC fragmentation since boot time reported as the number of original requests, the number of fragmented requests, the number of

fragments and the percentage of fragmentation. The percentage of fragmentation is defined as the total number of original requests versus the number of those requests that were fragmented.

- The degree of fragmentation since the last time interval reported as the number of original requests, the number of fragmented requests, the number of fragments, and the percentage of fragmentation (defined by the total number of original requests versus the number of those requests that were fragmented).
- The number of failures and time-outs per network since boot time and the last time interval.
- The number of resends per network since boot time and the last time interval.
- The number of duplicate packets received per network since boot time and the last time interval.
- The current configured priority per interface.
- The request dispatch latency (if applicable).
- The request round-trip times per network.

6.6.2 Check the GeoMirrors

You can use the AIX commands to check the status or usage of GeoMirrors devices and the GeoRM command to see

- The `lsdev` command to check the state of GMDs:

```
#lsdev -Cc geo_mirror
alpha20gmd Defined  Geographic Mirror Device
log21gmd Defined  Geographic Mirror Device
```

- The `lsattr` command to show the different attributes of GMDs:

```
#lsattr -El alpha20gmd
local_device   /dev/ralpha20lv       Local Logical Volume          True
state_map_dev  /dev/ralpha20sm       State_map Logical Volume      True
state_map_off  256                  State_map Starting Position   False
state_map_size 1024                 State_map Size in Blocks      False
remote_device  delta@/dev/ralpha20lv Remote Logical Volume         True
remote_device1                       Remote Logical Volume         True
remote_device2                       Remote Logical Volume         True
remote_device3                       Remote Logical Volume         True
remote_device4                       Remote Logical Volume         True
remote_device5                       Remote Logical Volume         True
remote_device6                       Remote Logical Volume         True
minor_num      20                    Minor Device Number           True
device_mode    sync                  Device Mode                   True
```

```
device_role    primary              Device Role                    True
hwm            128                  Device High Water Mark         True
sync_rate      32                   Sync Concurrency Rate          True
cell_size      4                   State_map Cell Size in Bits    False
region_size    32768               State_map Region Size in Bytes False
```

In this output, we can see the different attributes of the alpha20gmd GMD.

- The `gmd_show_state` command to show the state of a state map for the specified devices.

```
#/usr/sbin/gmd/gmd_show_state -l alpha20gmd
Point of View: Node alpha
--------------
Point of View GMD List:
-----------------------
Name: alpha20gmd
Status: DEFINED
State Map:
Cell     Value
-------   -----
      0   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
*
1048576
```

See Chapter 2, “GeoRM and HAGEO” on page 17, for more information about the meaning of the cell values.

You can also use the SMIT interface to get the information with the smit command:

`smit georm` or `smit hageo`

Next, select **GeoRM Utilities -> GeoMirror Utilities -> Show State Map**.

The Show State Map menu appears as shown in Figure 97 on page 202.

```
                          Show State Map

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Device Name                                          []          +
  Machine name                                         []


F1=Help             F2=Refresh          F3=Cancel        F4=List
Esc+5=Reset         F6=Command          F7=Edit          F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 97. Show State Map menu

The `gmd_show_state` command reads the state map from different sources depending on the GeoRm execution environment:

- All GeoMirror devices in DEFINED state, GMD driver not loaded: State map read from disk.
- Local mirrored device on line, but unavailable to applications (STOPPED state): State map read from Kernel.
- Resync process in operation: Static map of all regions being resynchronized.
- Active mirroring in progress: Snapshot of state map from kernel.

To print the GeoMirror usage statistics with the `gmdstat` command:

```
#/usr/sbin/gmd/gmdstat -l alpha20gmd
GMDs                              Device Statistics
------  ------------------------------------------------------------
               lx       lx/s        lw        lw/s       rx       rw
alpha20gmd     17     101.52       4.0       23.89        0      0.0

         Hosts                            KRPC Statistics
        -------   ------------------------------------------------------
                     xm       xm/s   rt   rp       kw     kw/s       kr
               delta  2      11.94    0    4      4.0    23.89      0.0

lx = local transfers, lx/s = local transfers/sec, lw = local writes
(kb), lw/s = local writes (kb)/sec, rx = remote transfers, rw = remote
writes (kb), xm = krpc transmits, xm/s = krpc transmits/sec, rt =
```

```
retries, rp = replies, kw = krpc writes (kb), kw/s = krpc writes
(kb)/sec, kr = krpc reads
```

This output is the result of the `gmdstat` command on a GeoRM configuration. The output of this command on HAGEO configuration is more complete.

This `gmdstat` command produces the following information:

- The amount of time the device was active since the last time interval
- The amount of data written locally and remotely since boot time measured in KB, KB/sec, requests, and requests/sec
- The amount of data written locally and remotely since the last time interval measured in KB, KB/sec, requests, and requests/sec
- The amount of time spent satisfying GeoMessage requests since boot time
- The amount of time spent satisfying GeoMessage requests since the last time interval
- The amount of data sent to and received from GeoMessage for each remote host since boot time measured in KB, KB/sec, requests, and request/sec
- The amount of data sent to and received from GeoMessage for each remote host since the last time interval measured in KB, KB/sec, requests, and request/sec

You can also use the SMIT interface: `smit georm` or `smit hageo`

GeoRM Utilities -> GeoMirror Utilities ->Show GeoMirror Device Statistics

The Show GeoMirror Device Statistics menu is shown in Figure 98 on page 204.

```
                        Show GeoMirror Device Statistics

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Device Name                                            []       +
  SECONDS between samples                                []
  NUMBER of samples                                      []        #



F1=Help              F2=Refresh          F3=Cancel       F4=List
Esc+5=Reset          F6=Command          F7=Edit         F8=Image
F9=Shell             F10=Exit            Enter=Do
```

Figure 98. Show GeoMirror Device Statistics menu

6.6.3 GeoRm logfiles

GeoRm uses the following files to log information about error occurrences during operation.

- AIX error log

  GeoRM makes entries in the AIX error log when:

  - A Promote Failure Timeout has been exceeded for a particular host.
  - A remote peer joins while another remote peer is active. In this case, the GMD device driver will automatically perform a gmddown for the remote peer that is marked active at the time the other remote peer joins.

  You can use the AIX `errpt` command to see these errors.

- GeoRM/HACMP custom log files

  To receive complete information on the GeoRM kernel processes, you can enable the following options in the /etc/syslogd.conf file:

  - *kern.info* - To receive general information messages
  - *kern.err* - To receive error condition messages
  - *kern.crit* - To receive critical condition messages

  You can send the log output in the file of your choice. In our case, we have added the following entries in /etc/syslogd.conf, and the log files are created in the /var/adm directory.

- kern.info /var/adm/geo_info
- kern.err /var/adm/geo_err
- kern.crit /var/adm/geo_crit

Create the files with the `touch` command:

- touch /var/adm/geo_info
- touch /var/adm/geo_err
- touch /var/adm/geo_crit

Afterwards, you must refresh the `syslogd` daemon:

- refresh -s syslogd

You should also monitor the size of the log files after starting this function since they can grow quite large.

The size of this log file depends on the activity of GeoMirror devices and on the log retention period. You need to ensure there is enough space in the file system for the log to grow. Furthermore, we suggest that you use the following method to keep log files in separate files for each month. This method uses cron to rename the syslog output file from the last month to `/var/adm/kern.log.YYMM` (Where `YYMM` stands for the year and month in two digits. For example, if this script was invoked on July 1, 1997, it would rename the log file to `/var/adm/kern.log.9706`).

To invoke the file renaming, perform the following steps:

1. Edit the cron table for the root user.

   Add the following lines to the cron table for the root user by using the `crontab -e` command:

   ```
   # HAGEO kernel proc message management tool.
   5 0 1 * * /admin/bin/rename-kern-log.ksh
   ```

   This invokes the shell script `/admin/bin/rename-kern-log.ksh` at 0:05 on the first day of each month.

2. Create the shell script called `/admin/bin/rename-kern-log.ksh`.

   Create the following shell script and change the file permission mode to 0755 (-rwxr-xr-x).

   ```
   #!/bin/ksh
   #
   # rename-kern-log.ksh -- rename "kern.info" syslog file.
   # Note: This shell script assume to invoke from cron once a month
   #       (at first day of each month).
   #
   ```

```
ORIGINAL_LOGFILE="/var/adm/kern.log"

CURRENT_YEAR=`date +"%y"`
CURRENT_MONTH=`date +"%m"`

typeset -Z2 YEAR
typeset -Z2 MONTH

if [ ${CURRENT_MONTH} = "01" ]; then
    MONTH="12"
    ((CURRENT_YEAR = ${CURRENT_YEAR} - 1))
    YEAR=${CURRENT_YEAR}
else
    ((CURRENT_MONTH = ${CURRENT_MONTH} -1))
    MONTH=${CURRENT_MONTH}
    YEAR=${CURRENT_YEAR}
fi

SAVED_LOGFILE="${ORIGINAL_LOGFILE}.${YEAR}${MONTH}"

mv ${ORIGINAL_LOGFILE} ${SAVED_LOGFILE}
touch ${ORIGINAL_LOGFILE}

exit 0

#
# end of rename-kern-log.ksh
```

6.7 Testing the DBFS configuration

This section describes the test procedure for your DBFS configuration. HACMP must be stopped first.

HACMP uses the `/usr/sbin/gmd/scripts/dbfs` command, which is called from the `/usr/sbin/gmd/scripts/geo_dbfs` script. The syntax for the `geo_dbfs` script is as follows:

```
$HAGEO_UTILS/geo_dbfs -e dbfs.envfile -n $node -p $MODEM_NUM
```

where `$node` is the remote node name and `$MODEM_NUM` is the remote node phone number that is set in the script. The `geo_dbfs` script generates an output file named `/tmp/dbfs.out`. You can use this file to trace problems related to the configuration of the dbfs.envfile file.

You can also use the `dbfs` command directly:

```
# cd /usr/sbin/gmd/scripts
```

```
# . ./dbfs.envfile
# ./dbfs -n brown -p 86911
```

If DBFS is properly configured, an attempt to log in to a remote node using the `ate` command occurs. If the login is successful, the `/usr/sbin/gmd/scripts/amiup` command is run. This command creates a shared memory segment that has the same key as clinfo (HACMP Cluster Information Daemon). It is used to check the current status of the node and report it to the remote GeoManager. Make sure that when you start HACMP in the nodes, the clinfo daemon is also started.

Start HACMP by entering `smit clstart`. The screen shown in Figure 99 appears.

```
                        Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
* Start now, on system restart or both                now        +

  BROADCAST message at startup?                       false       +
  Startup Cluster Lock Services?                      true        +
  Startup Cluster Information Daemon?                    true     +



F1=Help             F2=Refresh          F3=Cancel        F4=List
F5=Reset            F6=Command          F7=Edit          F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 99. Start Cluster Services menu

Change the last two options to `true`.

For some of the more complex failures HAGEO handles, clinfo needs to be in trap mode. To achieve this, run the following command in both machines:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

You can check the current clinfo settings with the following command:

```
# odmget -q subsysname=clinfo SRCsubsys
SRCsubsys:
        subsysname = "clinfo"
        synonym = ""
```

```
cmdargs = "-a"
path = "/usr/sbin/cluster/clinfo"
uid = 0
auditid = 0
standin = "/dev/null"
standout = "/dev/null"
standerr = "/dev/null"
action = 1
multi = 0
contact = 3
svrkey = 0
svrmtype = 0
priority = 20
signorm = 0
sigforce = 0
display = 0
waittime = 15
grpname = "cluster"
```

If clinfo is running on the remote node and HACMP has been started, the `amiup` command will return an IMOK string and exit code `0`. If the node's clinfo daemon is not running, the following message appears:

```
Can't get local cluster ID: Invalid shared memory operation.
```

If this happens, the return code from the command will be 1.

6.8 Backing up a GeoRM/HAGEO configuration

In general, a backup of user and system data is kept in case data is accidentally removed or in case of a disk failure. A formal backup process is really an insurance policy. You invest in the technology and time to back up systems so that, in the event of a problem, you can quickly rebuild them.

Since system and application backups are preferably done during periods of no usage (for instance, in the middle of the night), many installations implement an automated backup procedure using the AIX cron facility. While this is a very good procedure, the HACMP cluster environment presents some special challenges. The problem is that you never know which machine has your application data on line; so, you need to ensure that exactly the node that has a resource online initiates the backup of data.

It isn't actually important which of the several backup commands you use; what is important is the strategy you use. For the features and/or restrictions

of backup commands, such as `tar`, `cpio`, `dd`, or `backup`, refer to the *AIX Commands Reference V4.3*, SBOF-1877.

The basic GeoRM/HAGEO configuration should be backed up in case of a catastrophic failure. Both the GeoRM/HAGEO configuration and certain AIX files modified by GeoRM/HAGEO should be backed up.

Two standard methods for backing up a GeoRM/HAGEO configuration are available:

• Standard application methods. Some products provide online backup methods that can be applied in the GeoRM/HAGEO cluster environment with no change.

• Bring down one site and do a backup; then, bring it back up. It takes some time for the resynchronization process to complete because the first machine in the downed site rejoins the configuration.

If you are using a serial disk subsystem with the LVM, you may also back up mirrored data to tape and then reintegrate the mirror. This method requires customizing for your site.

If you are using a RAID device, consult the documentation for the RAID device for suggestions on handling backups.

AIX files modified by GeoRM/HAGEO

The following AIX files are modified by GeoRM/HAGEO and should be backed up:

- /etc/inetd.conf - The following lines are added to start the rpc.geod daemon:

```
rpc.geod        sunrpc_tcp      tcp     wait    root
/usr/sbin/gmd/rpc.geod rpc.geod 150005 1
```

- /etc/services - The following entries are added for the GeoMessage and rpc.geod daemons:

```
gmd_port                6755/tcp
gmd_port                6755/udp
```

Using geo_verify to Document your Configuration

The `geo_verify` command can be used with the `-l` option. If *full* is chosen for the output level, the `geo_verify` command will give a complete report of the GeoRM/HAGEO configuration along with any warnings and/or errors.

6.9 Upgrading a GeoRM/HAGEO cluster

In this section, we explain how to upgrade your GeoRM/HAGEO cluster. We advise you to create a full backup of the system with the `mksysb` command before the upgrade.

6.9.1 Upgrading a GeoRM cluster

If you want to install software maintenance called Program Temporary Fixes (PTFs) in your GeoRM cluster, you must stop the various components of the GeoRM software on the machine on which you are applying the PTF.

The normal steps for applying AIX fixes are the following:

1. Stop the different components of the GeoRM cluster as described in Section 6.1.2, “Stopping the GeoRM Configuration” on page 174.
2. Apply the software maintenance to this machine using the procedure described in the documentation distributed with the PTF.
3. Reboot the node to reload any GeoRM for AIX kernel extensions that may have changed as a result of the PTF being applied.
4. Restart the different components of the GeoRM configuration as described in Section 6.1.1, “Starting the GeoRM configuration” on page 173.
5. Repeat Steps 1 through 5 on the remaining machines.

- The machines should be kept at the same AIX maintenance levels wherever possible. This will, of course, not be true while the update is being applied, but it should be true at all other times.
- The machines should be running the same GeoRM maintenance levels.

6.9.2 Upgrading an HAGEO cluster

An HAGEO cluster is like an HACMP cluster: To install software maintenance, called Program Temporary Fixes (PTFs), to your HAGEO cluster, you must stop cluster services on the node on which you are applying a PTF. As with everything else in a cluster, applying software fixes should be done in a controlled fashion. With the method we are about to describe, you might even be able to keep your mission-critical application up and running during the update process, provided that the takeover node is designed to carry its own load as well as the takeover load. The normal method of applying AIX fixes is as follows:

1. Use the `smit clstop` fastpath to stop HAGEO on the node on which the PTF is to be applied. If you would like the resources provided by this node

to remain available to users, stop the cluster with the `takeover` option. This will allow the takeover node to continue serving users.

2. Apply the software maintenance to this node using the procedure described in the documentation distributed with the PTF.
3. Run the `/usr/sbin/cluster/diag/clverify` utility to ensure that no errors exist after installing the PTF. Test the fix as thoroughly as possible.
4. Reboot the node to reload any HACMP or HAGEO for AIX kernel extensions that may have changed as a result of the PTF being applied.
5. Restart HAGEO on the node using the `smit clstart` fastpath, and verify that the node successfully joined the cluster.
6. Repeat Steps 1 through 5 on the remaining cluster nodes.

Along with the normal rules for applying updates, the following general points should be observed for your HAGEO cluster:

- The machines should be kept at the same AIX maintenance levels wherever possible. This will, of course, not be true while the update is being applied, but it should be true at all other times.
- The machines should be running the same HACMP/HAGEO maintenance levels. There might be incompatibilities between various maintenance levels of HACMP; so, you must ensure that consistent levels are maintained across all cluster nodes. The cluster must be taken down to update the maintenance levels.

6.10 Security

In this section, we provide information about the security problems that the ./rhosts file or the DBFS configuration can generate.

6.10.1 /.rhosts file

In Chapter 4, “Installation and configuration” on page 81, we saw that we need the /.rhosts file to distribute or synchronize the configuration with all machines of the GeoRM/HAGEO configuration.

Enter the IP labels of all GeoRM network interfaces in the /.rhosts file on all machines at each site. The GeoRM/HAGEO synchronization and verification functions use rsh and, thus, require these /.rhosts entries. These IP labels may be removed for security reasons after GeoRM/HAGEO configuration is complete, or you can rename the file.

This file will be necessary if you change the GeoRM/HAGEO configuration and want to distribute the new configuration on all remote machines.

Note

The /.rhosts file is not required on SP systems running Kerberos.

6.10.2 Security of DBFS

Unfortunately, there are some security exposures in the DBFS function as it is shipped with the HAGEO product. To minimize these security exposures, we recommend the following actions:

1. Restrict the permissions of the GEOnode ODM class.

   The modem telephone numbers used by DBFS are stored in the HAGEO ODM class GEOnode in the $ODMDIR directory. The default permissions value for this file is 0644. With these permissions, the modem numbers can be easily obtained from the ODM by any user that has access to the node. The permissions for this file should be changed to 0600.

2. Create a non-root user.

   The default, dbfs.envfile, is configured with root as the DBFS user. This is too dangerous because the file will contain the password of root in ordinary text format. It is highly recommended that you create a non-root user for this purpose. For instance, you can create a user named DBMS with the following characteristics:

   - login name: dbfs
   - password: secretpw
   - GECOS: DBFS facility login user
   - home directory: /home/dbfs
   - login shell: /usr/bin/ksh

   After assigning the new password to the user you created, do not forget to log in as the user you created to validate the new password.

   To reduce the risk that the characteristics and environment variables for this user get changed and DBFS does not work properly when needed, the DBFS user should not have write permissions to the DBFS home directory.

   We assumed that the files /.profile and /.kshrc for root have the permission 700.

3. Change the DBFS user's password after each utilization of DBFS.
4. Change the permissions of dbfs.envfile.

As stated before, the password for the DBFS user is kept in the dbfs.envfile file in ordinary text format. The default permission for this file is 0644, which makes the information contained therein easily read or stolen by anyone who can access this node. You should change this file's access permission from 0644 to 0600. This file is read by a process that is forked from the HACMP event manager (it has root user access), and is not read by the DBFS login user process.

5. Monitor the Verify the `/etc/security/failedlogin` file to be sure that nobody tries to connect to the system using the DBFS user.
6. Accessible tty

   By default, all users are allowed to access all ttys, including /dev/ttyX, where /dev/ttyX is the modem-connected tty device name. You can change this default behavior by manually editing the /etc/security/user file. Only the dbfs user should be able to access the system from this tty. To accomplish this, you need to edit /etc/security/user manually as follows:

```
defaults:
...
admgroups =
ttys = ALL,!/dev/tty1
auth1 = SYSTEM
...
dbfs:
admin=false
ttys = /dev/tty1
```

6.11 Daily administration

In this section, we describe the different operations used to monitor your GeoRM/HAGEO configuration:

1. Verify the logs
   - As with an HACMP configuration, with HAGEO, we advise that you monitor the /tmp/hacmp.out HACMP log file for errors.
   - You can verify the AIX error log with the `errpt` command and see if any entries about the network or disks have been created.
   - If you have modified the /etc/syslog.conf file as described in Section 6.6.3, "GeoRm logfiles" on page 204, to receive full information on the GeoRM kernel processes, you can verify the contents of these files.
2. Verify the state map of each GMD

It is very important to verify the state map of each GMD. If a state map has some cells in a consistent and stale state, perhaps the GMD of the remote peer is stopped or the remote machine is down.

To verify the state map of all GMDs, you can use the following shell script:

```
#!/usr/bin/ksh

echo "\n\t\t Verification the GMDs of the machine "$(hostname) "\n"
for GMD in $(lsdev -Cc geo_mirror|awk ' { print $1 } ')
do
        if (/usr/sbin/gmd/gmd_show_state -l $GMD|grep "0xf" >/dev/null)
        then
                printf "\t %-20s %-10s \n" $GMD "Dirty cells"
        else
                printf "\t %-20s %-10s \n" $GMD "OK"
        fi
done
```

You can easily include this shell script in the crontab of the root user and send the output in a file or e-mail. In this way, you will have a detailed report of the state of your GMDs every day.

Chapter 7. Migrating an existing application to HAGEO

In this chapter, we describe the steps to take and point out some things to consider when migrating an existing HACMP cluster to HAGEO. Some scripts that can be used to automate the migration process are provided. We will also introduce scripts to automatically stop mirroring across sites, to remove the GMDs, and to restore the initial logical volume configuration.

7.1 Scenario description

The following sections are based on the clusters shown below. The procedure and considerations that we describe are generic, but we will refer to these clusters in order to make the explanations more clear. Figure 100 shows the existing cluster that will be migrated to HAGEO.

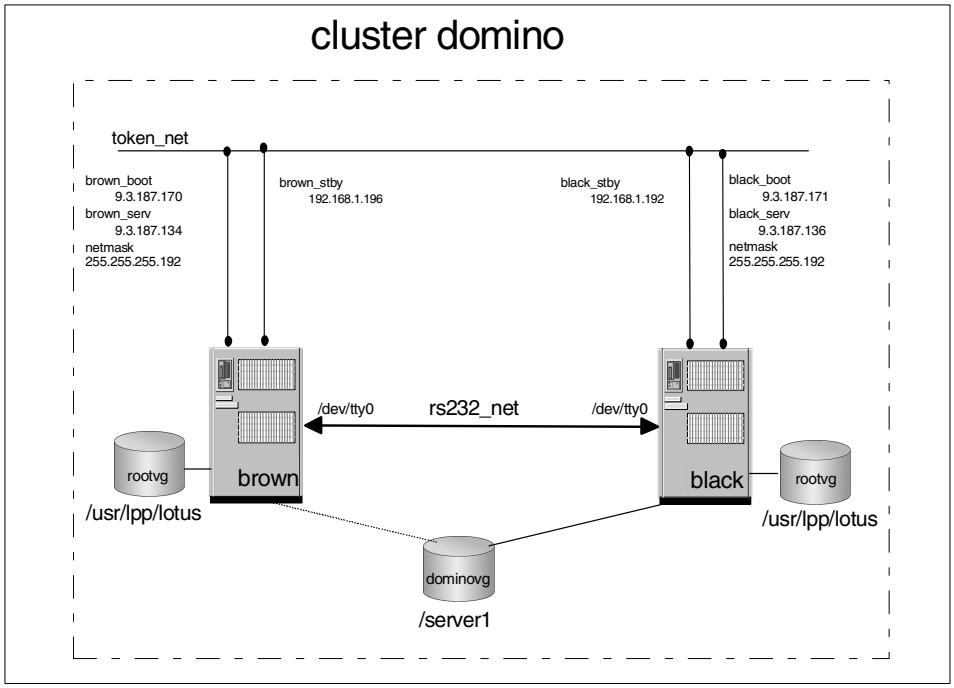

Figure 100. Cluster domino before the Migration to HAGEO

Cluster domino has two nodes, black and brown, that share the domino application server and the volume dominovg group in a Hot-Standby configuration. This volume group contains the file systems /server1 and two logical volumes lvserver1 and logdominovg, which are accessed by the

domino application server. Each node connects to the local token network (token_net) through two interfaces, one for service and the other for standby. A boot interface is defined for both nodes. There is also a serial connection between the machines: The local rs232 network (rs232_net).

Two resource groups are defined on the cluster :

- res_domino which contains the /server1 file system and the black_serv IP address. This resource is shared between the two systems but black has the high priority.
- res_brown which contains the brown_serv IP address. This resource has only the brown machine as a member.

The migration plan is to expand the cluster, adding another machine for hot backup in another site. Figure 101 shows the cluster after the migration.

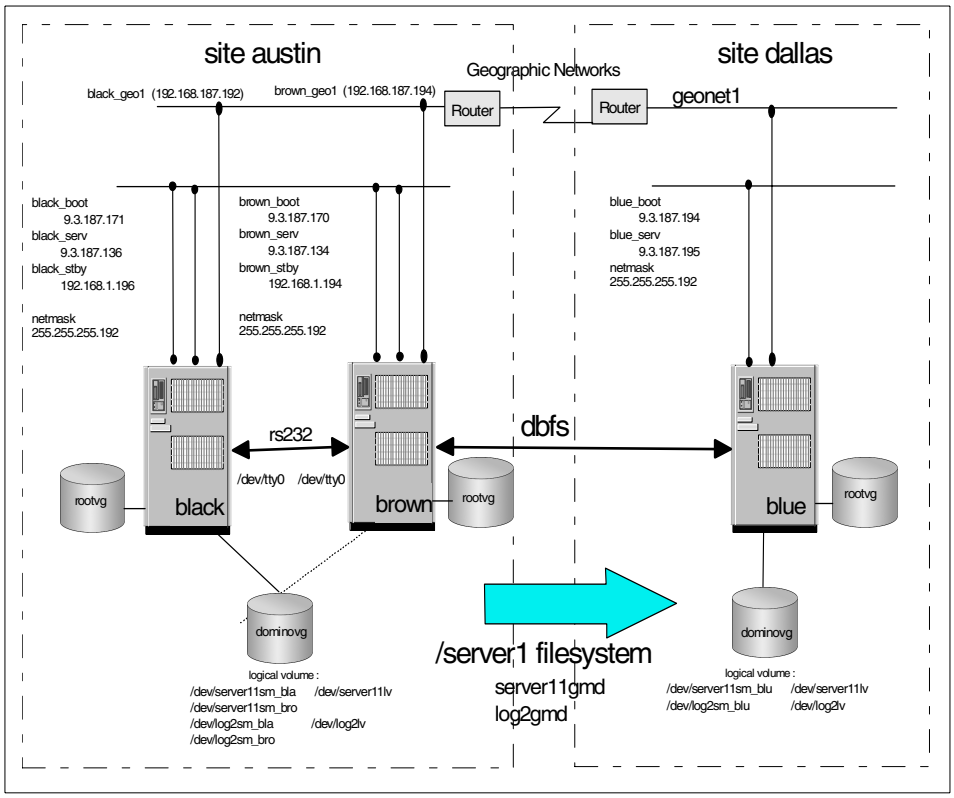

Figure 101. Cluster domino after the Migration to HAGEO

Cluster domino now has three nodes: Black and brown in site austin and blue in site dallas. The geo_net network has been added for communication between the sites. Each machine connects to this network through one adapter. DBFS was configured as a secondary path to distinguish between site isolation and site failure. The local networks in site austin were not altered.

In order to make the domino application server available across the geography, two GMDs, server11gmd and log2gmd, were added in each node. Notice that the GMDs have the same names as their logical volumes did before the migration. This is done to mirror the data across the geography. For the application server, the change is transparent. It does not know that it is accessing the GMDs instead of the logical volumes.

The migration was chosen to be done in this way; otherwise, you would probably have to alter the application.

The necessary state maps for this configuration were created in the respective volume groups for each site, in agreement with the suggested naming convention.

7.2 Naming convention

We suggest that you follow the naming convention introduced in Chapter 4. Table 28 provides a description of the naming convention that we use throughout the book.

Table 28. GMD naming conventions

Device description	Device name	LV type
Raw data LVs	<XXX><minor#>lv	gmd
GMD name for data LVs	<XXX><minor#>gmd	N/A
File systems	<XXX>	N/A
State Maps for data LVs	<XXX><minor#>sm_<YYY>	sm
Raw LV for jfslog	log<minor#>lv	gmdlog
GMD name for jfslog	log<minor#>gmd	N/A
State Maps for jfslog	log<minor#>sm_<YYY>	sm

<XXX>The name for the mount point of the file system or data logical volume (LV). Since the limit for the LV name is 15 characters, try to make this name at most 9 characters. Otherwise you will not be able to follow the naming convention when creating the state maps.

<YYY>The three first letters of the machine where is located the State Map logical volume.

<minor#> The GMD device minor number, which ranges from 0 to 1023.

Note

Actually, the LV type field is optional for the LVM. The logical volume types, gmd, gmdlog, and sm, are used as naming conventions only.

As can be seen in Table 28, the original logical volumes that will be mirrored across the geography should be renamed. We decided to add to the logical volume name the minor number associated with the GMD and the letters lv. In cluster domino (introduced in Section 7.2, “Naming convention” on page 217), nodes black and brown originally shared two logical volumes, lvdomino and logdominovg. These were renamed to server11lv and log2lv.

The GeoMirror devices were created with the minor number one for the server11gmd GMD and with minor number two for the log2gmd GMD. Site austin was chosen to be dominant. Since the State Map device logical volume must be unique within the cluster, for the State Maps, we add the three first letters of the machine where the Stat Map resides; so, the state maps in black, for example, were named server11sm_bla and log2sm_bla. The file system names do not need to be altered, nor do their configurations. The LV type field is optional. However, it can be used to easily distinguish the role of the logical volumes.

7.3 Migration procedure and considerations

This section covers the procedure to migrate an existing HACMP cluster to an HAGEO cluster. All the steps and considerations are discussed in detail. Finally, some scripts will be given to help automate the migration process.

7.3.1 Manual procedure

The following steps are necessary to migrate an HACMP cluster to HAGEO:

1. Stop HACMP.

   To configure HAGEO, some changes to the cluster topology will be needed. Since the HACMP version being used in our example is 4.3.1, it needs to be stopped prior to continuing the migration. Two options can be used in this case:

   - forced

     This option will stop all the HACMP daemons and processes without running any local procedures. In other words, the application servers will not be stopped, the shared resources will not be released and

service IP addresses will not be swapped to their respective boot addresses.
This option is very useful if you have other applications running in the cluster that are not going to be made highly available across the geography. This way, users can still access the other applications normally while you proceed with the migration.

- graceful

  This option will stop the application servers, release all the shared resources and return the service interfaces to their respective boot addresses.
  In cluster domino, the only application server defined is domino, which has to be stopped to alter the logical volume names. So, in this case, HACMP should be stopped gracefully.

After deciding which option is suitable for your environment, you can stop HACMP by entering `smit clstop`. The Stop Cluster Services screen shown in Figure 102 is displayed.

```
                              Stop Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Stop now, on system restart or both                  now                      +

  BROADCAST cluster shutdown?                         true                     +
* Shutdown mode                                       graceful                 +
      (graceful or graceful with takeover, forced)


F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 102. Stop Cluster Services screen

We advise that you stop the HACMP cluster with the *graceful* option, because you will need to reboot the system after the installation of the HAGEO software anyway.

2. Stop applications.

   If you stopped HACMP with the *forced* option, the application must always be active; then, you must stop the applications that use the logical volumes or those that are going to be mirrored across the geography. If

the logical volumes are open, you will not be able to rename them. Verify the state of the logical volumes with the `lsvg` command. For example:

```
# lsvg -l dominovg
dominovg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
logdominovg         jfslog     1     1     1    closed/syncd  N/A
lvserver1           jfs        62    62    1    closed/syncd  /server1
```

The LV State should be closed. If it is open, find the applications that are using the logical volume. The `fuser` command can be used to see and kill the process using the file system.

- To see the process using a file system, issue: `fuser -u /dev/lvserver1`
- To kill the process using a file system, issue: `fuser -k /dev/lvserver1`

With the `fuser` command, you can specify the name of the file system or the logical volume.

3. Configure the HAGEO networks and adapters.

   The first step is to configure the adapter for HAGEO configuration with smitty chinet on the three machines and to update the /etc/hosts and /.rhosts files with these new addresses.

   Table 29 gives the address of each machine on the HAGEO network that must be defined:

   Table 29. Address of each machine on the HAGEO network

Machine	Label IP	Address IP	Netmask
black	black_geo1	192.168.187.192	255.255.255.0
brown	brown_geo1	192.168.187.194	255.255.255.0
blue	blue_geo1	192.168.187.188	255.255.255.0

   After planning the IP addresses, add the geo_net adapter to each node in the cluster (including blue) and configure the interfaces. The interfaces can be configured through `smit mkinet`. From the panel presented, choose the option according to the type of adapter you have.

4. Save the data

   If your application resides on a file system, as it does in our case, you must save the data because, when the GMDs are defined, you must create a new file system on the GMD and restore the data.

5. Rename the logical volumes in the HACMP cluster.

To rename the logical volumes, you must activate the volume group on one or both systems, and this operation must be done only on this node, in our case, black. The new definition of the volume group will be imported on the second node, in our case, brown in step 6.

As stated earlier, all the logical volumes that are used by the application that will be geographically available should be renamed unless you are willing to alter the application. We will use the naming convention presented in Section 7.2, “Naming convention” on page 217.

No matter what procedure you choose, the application must access the GeoMirror devices that will be configured, not the logical volumes. The GMDs will act as an interface between the application and the logical volumes so that mirroring across the geography can be accomplished; so, if you choose to alter the application, make sure you do all the necessary changes so that it does not access the logical volumes directly.

6. Create the state map logical volumes in the original cluster.

   One more device, apart from the logical volume that actually holds the data, used to hold the state map is needed when creating a GeoMirror. Each node will have its own state map logical volume for each GMD that it can take over. If the GMDs are to be taken over locally, the state maps must be located in a shared volume group.

   In cluster domino, for instance, all the state maps are located in the dominovg volume group that is handled by HACMP. Two GMDs are needed for this cluster: server11gmd and log2gmd. In the shared volume group, we created four state map logical volumes: server11sm_bla, server11sm_bro, log2sm_bla, and log2sm_bro. Each state map needs 640 KB of disk space.

7. Import the new definition on the second node.

   Since we have modified the shared volume group, dominovg, on machine black, we need to import the new definition on machine brown. We must export the definition of the volume group first and then reimport it.

Note

After the import of the volume group, verify that the `Activate volume group at system restart` option of this volume group is defined as no.

8. Create all the logical volumes in the remote node.

   If the logical volumes reside in the external volume group, as they do in our case, you must first create the volume group if it does not exist. In our

HAGEO configuration, we created the dominovg volume group on the system, blue.

At this point, you need to create the logical volumes (for data and state maps) in the node that are going to be added to the cluster. The logical volume names and sizes must be the same in all the machines in which the GMD is defined. The state map logical volume names must be different in every node, and each requires 640 KB (one physical partition) of disk space.

In cluster domino, we added the data logical volumes to node blue: server11lv, log2lv, server11sm_blu, and log2sm_blu

9. In the new nodes, install the applications that they can take over. If these applications are controlled by HACMP with an application server, do not forget to copy the start and stop scripts to the new nodes.

   In our case, on the blue machine, we must create the /usr/lpp/lotus file system in the rootvg volume group and create the group and user notes with the same ID.

10. Install HAGEO software.

    Before installing HAGEO in all the nodes, make sure that every node has HACMP installed. HAGEO needs HACMP to operate. It does not function by itself.

    Then, you may install HAGEO in all the nodes. We recommend that you install the latest PTFs available for the HAGEO software and reboot the system after this operation.

11. Alter the HACMP topology configuration.

    Before configuring HAGEO, you need to alter the cluster topology configuration to include the new node that you are adding and the geographical networks. Despite the nodes being in different sites, the cluster will be the same. If you are adding more than one node in the other site and there are local networks to connect them, you should add these also. In our cluster domino example, the topology should be altered to include node blue (with its local network adapters, blue_boot and blue_serv) and the geo_net network adapters (brown_geo1, black_geo1, and blue_geo1).

    After installing HAGEO, two new network interface modules (NIMs) are added: Geo_Primary and Geo_Secondary. You may alter the default settings for these NIMs in ODM to reflect the behavior, speed, and latency of your geographical network. However, you can still use the standard network types of HACMP.

After configuring the cluster topology, do not forget to run the `/usr/sbin/cluster/diag/clverify` utility to verify the altered cluster topology. If the network local adapters for the blue machine are defined on the topology of HACMP, `clverify` will give an error. You must synchronize the topology with the Ignore Cluster Verification Errors? option set to *yes*.

For more information about configuring HACMP, refer to the *HACMP for AIX Installation Guide*.

12. Configure GeoMessage.

In this step, you need to define the paths that can be used to communicate between the sites to HAGEO. There are two ways to accomplish this. One way is to import HACMP definitions and then delete the network definitions that are only local to the sites. This method was used throughout Chapter 4, “Installation and configuration” on page 81. Refer to this chapter for more details. The other option is to manually define the geographical networks.

We will use cluster domino to give an example of the second option. First, enter `smit hageo`, and then select **Manage GeoMessage**. From the following menu, choose **Configure GeoMessage**. The Configure GeoMessage screen shown in Figure 103 appears.

```
                              Configure GeoMessage

Move cursor to desired item and press Enter.

  Show GeoMessage Configuration
  Import HACMP Definitions
  Distribute GeoMessage Definitions to Remote Machines
  Configure GeoMessage Machines
  Configure GeoMessage Networks
  Configure GeoMessage Interfaces



F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 103. Configure GeoMessage screen

First, select **Configure GeoMessage Machines**, and then select **Create new machine definition**. The Create new machine definition screen appears as shown in Figure 104 on page 224.

```
                Create new machine definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Machine Name                                       [black]



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 104. Create new machine definition screen

Fill in the node name. You should repeat the configuration for nodes brown and blue. Next, go back to the Configure Geomessage menu and select **Configure GeoMessage Networks**, and then select **Create new network definition**. The Create new network definition screen appears as shown in Figure 105.

```
                Create new network definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Type                                               [Geo_Primary]
* Network Name                                       [geonet1]



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 105. Create new network definition screen

In this screen, you should fill in the type of network as well as its name. The next step is to define the interfaces; so, go back to the Configure Geomessage menu and choose **Configure GeoMessage Interfaces** and then **Create new interface definition**. This screen will appear as shown in Figure 106 on page 225.

```
                Create new interface definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Machine Link                                        [black]          +
* Network Link                                        [geonet1]        +
* Interface Address                                   [192.168.187.192]
* Interface Name                                      [black_geo1]



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 106. Create new interface definition screen

Fill-in the machine and network previously defined to GeoMessage, the IP address, and the name associated with the interface. You should repeat this configuration for all the service interfaces that connect to the geo_net network.

Note

When defining a network and its interfaces to GeoMessage, they should have been previously defined to HACMP. All the topology changes must first be done in HACMP and then in HAGEO. Also, notice that the changes made in HACMP are not automatically reflected to HAGEO. You must manually alter both configurations.

13. Distribute the configuration.

    In the main menu for HAGEO, select **Manage GeoMessage**. Then, choose **Configure GeoMessage** and **Distribute GeoMessage Definitions to Remote Machines**.

Note

The names of all the machines in the cluster must be resolved into an IP address (through /etc/hosts or DNS) associated with one of the geographical network interfaces, and the /.rhosts file must be configured for the synchronization operation to work properly.

14. Configure GeoManager.

    In this step, you configure the GEOnode and GEOsite ODM classes. These define the sites and nodes that are located at each site. They also contain information about the backup communication method being used and the user that receives notes after failures.

    For cluster domino, sites dallas and austin were configured. The former contains node blue and the latter contains nodes black and brown. Enter the definitions on only one node. These will be synchronized after configuring GeoMessage in the next step.

    Refer to Chapter 4, “Installation and configuration” on page 81, for a detailed description of site and machine configuration.

15. Synchronize the configuration.

    In the main menu for HAGEO, select **Manage HAGEO Sites/Machines**. Then, choose **Sync Site/Machine Definitions to all Machines**.

16. Start GeoMessage.

    You now need to start GeoMessage on all machines with the following command:

    ```
    # /usr/sbin/krpc/cfgkrpc -ci
    ```

    You may also start GeoMessage by entering `smit hageo` and selecting **Manage GeoMessage=>Start GeoMessage**.

17. Configure the GeoMirror Devices.

    You should now create the GMDs. This configuration should be done on each node. These definitions cannot be synchronized. Refer to Chapter 4, “Installation and configuration” on page 81, for a detailed description of GMD configuration.

    For cluster domino, we added the GMDs server11gmd and log2gmd on each of the nodes in clusters black, brown, and blue. For the brown and black machines, you must activate the dominovg volume group before defining the GMDs.

    Before proceeding to the next step, you must start the GMDs on the black and blue machines.

18. Create the file system

    You must add an entry in the /etc/filesystems file for /server1 filesystem. Here is the entry that was created in our case:

    ```
    /server1:
            dev             = /dev/server11lv
            vfs             = jfs
    ```

```
        log             = /dev/jfslogtmp
        mount           = false
        check           = false
        options         = rw
        account         = false
```

Afterwards, we must change the file system log logical volume to point to the GeoRM file system log with the following command:

```
chfs -a log=/dev/log2gmd /server1
```

Make sure that the entry in the /etc/filesystems file is pointing to the GMD. Here is the entry for the /server1 file system after the log logical volume change:

```
/server1:
        dev             = /dev/server11gmd
        vfs             = jfs
        log             = /dev/log2gmd
        mount           = false
        check           = false
        options         = rw
        account         = false
```

Then, create the mountpoint /server1, and create a file system on the device with the following command:

```
mkfs -l /server1 /dev/server11gmd
```

Finally, we must mount this file system to restore the data saved in Step 4.

19. Configure the resource groups.

    We will apply the considerations discussed here to our cluster domino example. If your environment is different, refer to Chapter 4, “Installation and configuration” on page 81, to learn how to configure the necessary resource groups for your case.

    Before the migration, the domino cluster had the res_domino resource group, which is described in Figure 107 on page 228.

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
Resource Group Name                                    res_domino
Node Relationship                                      cascading
Participating Node Names                               black brown

Service IP label                                      [black_serv]     +
Filesystems                                           [/server1]       +
Filesystems Consistency Check                          fsck            +
Filesystems Recovery Method                            sequential      +
Filesystems to Export                                 []               +
Filesystems to NFS mount                              []               +
Volume Groups                                         []               +
Concurrent Volume groups                              []               +
Raw Disk PVIDs                                        []               +
 AIX Connections Services                              []               +
AIX Fast Connect Services                             []               +
Application Servers                                   [app_domino]     +
Highly Available Communication Links                  []               +
Miscellaneous Data                                    []

Inactive Takeover Activated                            false           +
9333 Disk Fencing Activated                            false           +
SSA Disk Fencing Activated                             false           +
Filesystems mounted before IP configured               false           +
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 107. Configure Resources for a Resource Group - Screen 1

From Figure 107, we can see that black and brown share the black_serv IP address, the dominovg volume group, and the domino application server. Node black has priority over the resources. With the addition of node blue as a hot-backup for the application domino, the resource group configuration must be altered. When creating a resource group in HAGEO for an application server, site names must be specified, and node names must be given for volume groups. Therefore, these resources will have to be defined in different resource groups when migrating to HAGEO.

Let us first consider the volume group case. Since, in case of failure in one of the nodes, the volume groups will not be taken over across the geography, one resource group should be defined per site. Another

important issue is that even though there is not a node to take over the volume group dominovg from blue, it should also be defined in a resource group. This is due to the fact that HAGEO will only automatically start the GMDs whose logical volumes are located in a volume group defined in a resource group. Only for the rootvg volume group is this step not necessary.

In our example, we first altered the res_domino resource group to remove the application server and the file systems entries as shown in Figure 108.

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                [Entry Fields]
Resource Group Name                                  res_domino
Node Relationship                                    cascading
Participating Node Names                             black brown

Service IP label                                    [black_serv]             +
Filesystems                                         []                       +
Filesystems Consistency Check                        fsck                    +
Filesystems Recovery Method                          sequential              +
Filesystems to Export                               []                       +
Filesystems to NFS mount                            []                       +
Volume Groups                                       [dominovg]               +
Concurrent Volume groups                            []                       +
Raw Disk PVIDs                                      []                       +
AIX Connections Services                            []                       +
AIX Fast Connect Services                           []                       +
Application Servers                                 []                       +
Highly Available Communication Links                []                       +
Miscellaneous Data                                  []

Inactive Takeover Activated                         false                   +
9333 Disk Fencing Activated                         false                   +
SSA Disk Fencing Activated                          false                   +
Filesystems mounted before IP configured                false                    +
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 108. Configure Resources for a Resource Group - Screen 2

Then, we added another resource group, res_blue, to control the dominovg volume group and the IP service address, blue_serv, of the blue machine as shown in Figure 109 on page 230.

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
Resource Group Name                                    res_blue
Node Relationship                                      cascading
Participating Node Names                               blue

Service IP label                                      [blue_serv]             +
Filesystems                                           []                      +
Filesystems Consistency Check                          fsck                   +
Filesystems Recovery Method                            sequential             +
Filesystems to Export                                 []                      +
Filesystems to NFS mount                              []                      +
Volume Groups                                         [dominovg]              +
Concurrent Volume groups                              []                      +
Raw Disk PVIDs                                        []                      +
AIX Connections Services                              []                      +
AIX Fast Connect Services                             []                      +
Application Servers                                   []                      +
Highly Available Communication Links                  []                      +
Miscellaneous Data                                       []

Inactive Takeover Activated                            false                  +
9333 Disk Fencing Activated                            false                  +
SSA Disk Fencing Activated                             false                  +
Filesystems mounted before IP configured               false                  +
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 109. Configure Resources for a Resource Group - Screen 3

In this way, all the GMDs will be automatically started when HACMP is initialized.

For the application server, we created the res_hageo resource group as shown in Figure 110 on page 231.

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
Resource Group Name                                    res_hageo
Node Relationship                                      cascading
Participating Node Names                               austin dallas

Service IP label                                      []                        +
Filesystems                                           [/server1]                +
Filesystems Consistency Check                          fsck                     +
Filesystems Recovery Method                            sequential               +
Filesystems to Export                                 []                        +
Filesystems to NFS mount                              []                        +
Volume Groups                                         []                        +
Concurrent Volume groups                              []                        +
Raw Disk PVIDs                                        []                        +
AIX Connections Services                              []                        +
AIX Fast Connect Services                             []                        +
Application Servers                                    [app_domino]                +
Highly Available Communication Links                  []                        +
Miscellaneous Data                                     [server11gmd log2gmd]

Inactive Takeover Activated                            false                    +
9333 Disk Fencing Activated                            false                    +
SSA Disk Fencing Activated                             false                    +
Filesystems mounted before IP configured               false                    +
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 110. Configure Resources for a Resource Group - Screen 4

As stated before, for resource groups that include application servers, the site names should be specified. For the application server domino, site austin has the highest priority. This means that the priority order to take over this resource is black, followed by brown and blue. Blue will only take over the resource if both black and brown fail. Also, notice that GMDs used by the application server should be listed in the Miscellaneous Data field. This is mandatory for the handling of the domino application server in case of failure.

After adding these resource groups, the `clverify` utility will return an error because it does not know how to handle sites. You may ignore this error. Do not forget to synchronize the topology by setting the `Ignore Cluster Verification Errors?` option to *yes*.

20. Configure DBFS.

    Configure the dbfs.envfile in the non-dominant site as explained in Chapter 4, “Installation and configuration” on page 81.

21. Integrate HACMP and HAGEO with the configure_events script.

    The configure_events script will plug the GMD events into HACMP. Table 30 shows a summary of the HACMP events that will have pre-events and post-events added to them.

Table 30. HACMP and HAGEO integration

HAGEO Pre-event	HACMP event	HAGEO Post-event
Geo_stop_gmds	node_down node_down_complete	Geo_node_down Geo_krpc_unload
Geo_remote_node_up	node_up	Geo_start_server
Geo_shared_remote_dev	node_up_complete network_down_complete	Geo_network_down krpc_net_chng
Geo_stop_gmds	release_vg_fs network_up_complete	krpc_net_chng

    If you customized pre-events and/or post-events to any of these HACMP events, when you run the configure_events script, your custom configuration will be lost; so, first review your configuration and document all the pre-events and post-events that you have added.

    Next, execute the following command with the three nodes as arguments:

    ```
    # /usr/sbin/gmd/scripts/configure_events black brown blue
    ```

    You will need to reboot the nodes at this time. After the nodes have been reinitialized, you should append your previously-configured event customization scripts to the respective HAGEO pre/post event scripts.

22. Verify the configuration.

    Now, you need to verify the HAGEO configuration. Enter `smit hageo` and select **Verify HAGEO Configuration**.

    You can also verify the HAGEO configuration by executing the following command:

    ```
    # /usr/sbin/gmd/geo_verify
    ```

23. Initial synchronization of the mirror.

    If you already had data in the existing cluster, you will now need to synchronize the data between the sites. When you start the GMDs, the data will not be automatically synchronized. You need to modify the state

maps in the joining nodes. See Section 7.4, “Initial synchronization of GeoMirror devices” on page 234.

Go to the next step only after synchronizing the data. If you start HACMP and the data is still inconsistent, data divergence is likely to occur.

24. Start HACMP.

For some of the more complex failures HAGEO handles, clinfo needs to be in trap mode. To achieve this, you need to run the following command on both nodes:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

Now, start HACMP by entering `smit clstart`. The screen shown in Figure 111appears.

```
                Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Start now, on system restart or both                now                        +

  BROADCAST message at startup?                       true                       +
  Startup Cluster Lock Services?                      true                       +
  Startup Cluster Information Daemon?                 true                       +



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 111. Start Cluster Services screen

Make sure you change the Startup Cluster Lock Services and Startup Cluster Information Daemon options to yes.

You need to start HACMP in all nodes, and we advise you to first start it in the node having the highest priority on the dominant site. In our HAGEO configuration, the order of starting is black, brown, blue.

7.3.2 Scripts for the migration to HAGEO

This section describes the recommend migration procedure.

7.4 Initial synchronization of GeoMirror devices

When you first create the logical volumes on the joining nodes, they do not yet have any data. You need to synchronize the data from the original nodes to the joining nodes. This process is not automatic when you create the GMDs in all nodes. Only the future updates will be automatically mirrored after the GMDs are started.

As you already know, every GMD has at least one state map associated with it, that tracks which cells have not had their mirror copy written. When you create the GMDs the state maps are clean, as if all the data were synchronized between the sites. The `gmddirty` utility can be used to mark all the cells in the state map as stale, to force the transfer of data between the sites. The following procedure shows how this can be accomplished:

Start the GMDs on the local node.

First, start the GMDs on the node that holds the actual data. For example, in cluster domino, we would execute the following commands on node black:

```
# varyonvg dominovg (if necessary)
# /etc/methods/cfggmd -A
# /etc/methods/gmddown -A blue
# /etc/methods/startgmd -A
```

Mark all the state maps as stale.

Then, use the `gmddirty` command to mark all the cells in the state maps as stale. In cluster domino, we would execute the following commands on node black:

```
# /usr/sbin/gmd/gmddirty -l server11gmd
# /usr/sbin/gmd/gmddirty -l log2gmd
```

Start the GMDs on the remote node.

Now, you should start the GMDs on the remote node to start the synchronization. The script, `Geo_Prestart_GMDs`, located in /usr/sbin/gmd/scripts, can be used for this purpose. It automatically varies on the proper volume groups and starts the GMDs. In cluster domino, we executed this script in node blue.

There are some considerations you should take into account before starting the GMDs. If the application is currently available in the local site and the amount of data to be synchronized is very large, you may want to do the transfer of data during periods when the system is less busy. Otherwise, the

response time of the application may become unacceptable, especially if you are using sync or mwc GMDs.

If the amount of data to be synchronized is very large, you may want to take the remote machine or disks to the local site, connect to the local network, and synchronize the data there. This would take advantage of the faster local area network in the local site. Then, when the machine is installed on the remote site, most of the data will have already been mirrored, and the impact of the synchronization process will not be so high.

7.5 Scripts to reestablish the initial configuration

In this section, you are introduced to some scripts that automatically rename the LVs and create the GMDs according to the naming convention described earlier. The scripts substitute some of the steps of the migration procedure shown in Section 7.3.1, “Manual procedure” on page 218.

7.5.1 Scripts for the migration to HAGEO

This section will introduce scripts that automatically perform a migration.You should still follow the steps presented in the previously-mentioned section, but you can replace some of the steps with the scripts shown in this section. Do not skip steps or change the order of the steps in the migration procedure.The scripts support up to two nodes per site and assume that each new GMD name will be the same as the original logical volume name associated with it. However, you can always alter the scripts to match your required needs. All the script contents are listed in Appendix A, “Scripts for the migration to HAGEO” on page 305. In this section, you will be shown how to use them.

7.5.1.1 Scripts to rename LVs and create state map LVs

The scripts shown in this section replace steps 4, 5, and 6 of the manual migration procedure. The steps to properly use the scripts are as follows:

1. Save the volume group information to a file.
2. First, you need to run the `lsvg -l` command and save its output to a file. For instance, in cluster domino:

```
# lsvg -l dominovg > dominovg.out
# cat localvg.out

#cat dominovg.out

dominovg:

LV NAME            TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
```

```
logdomvg           jfslog     1     1     1    open/syncd    N/A
lvsrv1             jfs        62    62    1    open/syncd    N/A
```

The output file name does not need to be in a specific format or follow any naming conventions. Choose the name that is most meaningful to you.

If some of the logical volumes in the volume group are not going to be mirrored across the geography, you do not need to alter their names. Edit the output file, and remove the lines related to these logical volumes.

3. Execute the `Generate_LV_list.ksh` script.

   This script generates an intermediate file with new names for the logical volumes and the names for state map logical volumes, all according to the suggested naming convention. For cluster domino, we executed the following command:

```
# Generate_LV_list.ksh -l black,brown -r blue -i dominovg.out -o \
lvlist.out
Where:
-lnode names in the local site.
-rnode names in the remote site.
-iinput file name which was created in the previous step.
-ooutput file name. This file will be used in the remaining steps.
```

   Figure 112 shows the lvlist.out file generated for cluster domino.

```
# List of LV names generated with Generate_LV_list.ksh
# on node:  black
# date:  Thu Dec 9 11:47:22 CST 1999
#
dominovg
#
#Original         Original  PP   GMD  New                    New
#LV_name          LV_type   size min# LV_name                LV_type
# you should not change above this line.
logdomvg          jfslog    1    0    logdomvg0lv            gmdlog
=LOCAL            =black    1    0    logdomvg0sm_11         sm
=LOCAL            =brown    1    0    logdomvg0sm_12         sm
=REMOTE           =blue     1    0    logdomvg0sm_21         sm
lvsrv1            jfs       62   1    lvsrv11lv              gmd
=LOCAL            =black    1    1    lvsrv11sm_11           sm
=LOCAL            =brown    1    1    lvsrv11sm_12           sm
=REMOTE           =blue     1    1    lvsrv11sm_21           sm
```

Figure 112. lvlist file listing generated with Generate_LV_list.ksh

The first uncommented line shows the volume group name. Three similar blocks (every four rows), one for each of the logical volumes in the localvg

volume group, are shown next. For example, the logical volume lvsrv1, whose original type was jfs, would be changed to lrvsrv11lv, and the state map logical volumes would be created as lssrv11sm_11, lssrv11sm_12, and lssrv11sm_21 (columns one and five). The third and fourth columns show the number of partitions in the logical volume and the GMD minor number. The last column shows the new logical volume type that will be assigned to the logical volume.

After generating the file, you must verify its contents manually. First, ensure that the new logical volume name is unique within the cluster. Then, you should verify that the new logical volume names are no longer than 15 characters. You cannot create logical volumes with names longer than 15 characters. This could happen because the suggested naming convention adds some characters to the original logical volume name. Warnings will be given in the generated file in this case. If this is the case, you should edit the file and alter the appropriate line with a new name for the logical volume that is meaningful to you. Even in this case, try to adhere as much as possible to the naming convention. It will make administration easier in the future.

4. Create state map logical volumes in the local node.

   To create the state maps logical volumes, execute the following script on the local node:

   ```
   # Create_local_state map_LV.ksh -i lvlist.out
   ```

   The `-i` option specifies the input file created in step 2. All the local state map logical volumes specified in the input file will be created.

5. Rename the logical volume names on the local node.

   To rename the original logical volumes names on the local node, execute the following script:

   ```
   # Change_local_LV_name.ksh -i lvlist.out
   ```

   The original logical volumes will be renamed according to the contents of the fifth column of the lvlist.out file. At this point, you have finished all the logical volume changes on the local site.

   If you have local peer nodes, vary off the volume group on the node you were working on, export the volume group in the local peer node, and import the volume group again. In this way, the local peer node will reflect the new configuration.

6. Alter volume group information if necessary.

   If the volume group name in the remote site differs from that in the local site, you need to edit the file generated in step 2 to reflect the proper

name. In cluster domino, however, the volume group was named dominovg on all nodes.

```
# List of LV names generated with Generate_LV_list.ksh
# on node:  black
# date:  Thu Dec 9 11:47:22 CST 1999
#
dominovg
#
#Original         Original  PP   GMD  New                    New
#LV_name          LV_type   size min# LV_name                LV_type
(...)
```

7. Generate a script to create the state map LVs on the remote nodes.

   To generate a script to create the state map logical volumes on the remote nodes, execute the following script:

   ```
   # Generate_remote_state map_LV_script.ksh -i lvlist.out -o r_state
   map.ksh
   ```

   Once again, the necessary information is obtained from the file generated in step 2. For instance, the contents of the r_state map.ksh in cluster domino are:

   ```
   #cat r_state
   /usr/sbin/mklv -t sm -y logdomvg0sm_21 dominovg 1
   /usr/sbin/mklv -t sm -y lvsrv11sm_21 dominovg 1
   ```

8. Generate a script to create the data logical volumes on the remote nodes.

   To generate a script to create the data logical volumes on the remote nodes, enter the following:

   ```
   # Generate_remote_LV_script.ksh -i lvlist.out -o r_datalv.ksh
   ```

   The necessary information is supplied by the file generated in step 2. The file, r_datalv.ksh, for cluster domino would be:

   ```
   /usr/sbin/mklv -t gmdlog -y logdomvg0lv dominovg 1
   /usr/sbin/mklv -t gmd -y lvsrv11lv dominovg 62
   ```

9. Create logical volumes on the remote nodes.

   Transfer the files generated in steps 6 and 7 to the remote nodes. If you use the ftp protocol, remember that the file permissions will be changed. You need to give both files root execute permission.

   Then, vary on the volume group that will hold the logical volumes, and execute the scripts. Verify that the results were what you expected with the `lsvg -l` command.

7.5.1.2 Scripts to create the GeoMirror devices

The scripts presented in this section can be used to substitute for Step 14 of the manual procedure.

First, execute the following:

```
# Generate_GMD_script.ksh -i lvlist.out -o gmd.ksh
```

The output file, gmd.ksh, is a script that can be used in all the nodes to create the GMDs. The lvlist.out is the output file generated in Step 2.

The `gmd.ksh` script will, by default, create GMDs with sync device_mode. If you want another type of device_mode (MWC or async), you should edit the script, and change this parameter manually. Notice that, if you use async device_mode, you also have to define the device_role as primary or secondary.

For cluster domino, the following shows the `gmd.ksh` script:

```
#!/bin/ksh
#
# This script is generated by Generate_GMD_script.ksh
# on node:  black
# date:  Thu Dec 9 16:39:25 CST 1999
#

LOCALNODENAME=`/usr/sbin/cluster/utilities/get_local_nodename`

case $LOCALNODENAME in
black)
/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a device_mode=sync -a
device_role=none\
-l logdomvg -a minor_num=0\
 -a local_device=/dev/rlogdomvg0lv\
 -a state_map_dev=/dev/rlogdomvg0sm_11\
 -a locpeer_smdev=brown@/dev/rlogdomvg0sm_12\
 -a remote_device=blue@/dev/rlogdomvg0lv
/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a device_mode=sync -a
device_role=none\
 -l lvsrv1 -a minor_num=1\
 -a local_device=/dev/rlvsrv11lv\
 -a state_map_dev=/dev/rlvsrv11sm_11\
-a locpeer_smdev=brown@/dev/rlvsrv11sm_12\
 -a remote_device=blue@/dev/rlvsrv11lv
;;
brown)
```

```
/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a device_mode=sync -a
device_role=none\
 -l logdomvg -a minor_num=0\
 -a local_device=/dev/rlogdomvg0lv\
 -a state_map_dev=/dev/rlogdomvg0sm_12\
 -a locpeer_smdev=black@/dev/rlogdomvg0sm_11\
 -a remote_device=blue@/dev/rlogdomvg0lv
/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a device_mode=sync -a
device_role=none\
 -l lvsrv1 -a minor_num=1\
 -a local_device=/dev/rlvsrv11lv\
 -a state_map_dev=/dev/rlvsrv11sm_12\
 -a locpeer_smdev=black@/dev/rlvsrv11sm_11\
 -a remote_device=blue@/dev/rlvsrv11lv
;;
blue)
/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a device_mode=sync -a
device_role=none\
 -l logdomvg -a minor_num=0\
-a local_device=/dev/rlogdomvg0lv\
 -a state_map_dev=/dev/rlogdomvg0sm_21\
 -a remote_device=black@/dev/rlogdomvg0lv\
 -a remote_device=brown@/dev/rlogdomvg0lv
/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a device_mode=sync -a
device_role=none\
 -l lvsrv1 -a minor_num=1\
 -a local_device=/dev/rlvsrv11lv\
 -a state_map_dev=/dev/rlvsrv11sm_21\
 -a remote_device=black@/dev/rlvsrv11lv\
 -a remote_device=brown@/dev/rlvsrv11lv
;;
esac

#
# end of file.
```

Notice that this script contains the commands to generate the GMDs in every node in cluster domino. However, it only executes the ones associated with the node that is currently executing the script.

To use this script to create the GMDs, first give root execute permission to the `gmd.ksh` script, and then execute it in the local node. Next, transfer the file to the remaining nodes in the cluster (local and remote). Give root execute permission to the script on all nodes. Run the script on the other nodes. You can verify whether the GMDs were properly created with the `# lsdev -Cc geo_mirror` command.

7.6 Initial synchronization of GeoMirror devices

When you first create the logical volumes on the joining nodes, they do not yet have any data. You need to synchronize the data from the original nodes to the joining nodes. This process is not automatic when you create the GMDs in all nodes. Only future updates will be automatically mirrored after the GMDs are started.

As you already know, every GMD has at least one state map associated with it that tracks which cells have not had their mirror copy written. When you create the GMDs, the state maps are clean, as if all the data were synchronized between the sites. The `gmddirty` utility can be used to mark all the cells in the state map as stale to force the transfer of data between the sites. The following procedure shows how this can be accomplished:

1. Start the GMDs on the local node.

   First, start the GMDs on the node that holds the actual data. In cluster domino, for example, we would execute the following commands on node black:

   ```
   # varyonvg dominovg (if necessary)
   # /etc/methods/cfggmd -A
   # /etc/methods/gmddown -A blue
   # /etc/methods/startgmd -A
   ```

2. Mark all the state maps as stale.

   Use the `gmddirty` command to mark all the cells in the state maps as stale. In cluster domino, we would execute the following commands on node black:

   ```
   # /usr/sbin/gmd/gmddirty -l lvsrv1
   ```

3. Start the GMDs on the remote node.

   Now, you should start the GMDs on the remote node to start the synchronization. The `Geo_Prestart_GMDs` script, located in /usr/sbin/gmd/scripts, can be used for this purpose. It automatically varies on the proper volume groups and starts the GMDs. In cluster domino, we executed this script in node blue.

   There are some considerations you should take into account before starting the GMDs. If the application is currently available in the local site and the amount of data to be synchronized is very large, you may want to do the transfer of data during periods when the system is less busy. Otherwise, the response time of the application may become unacceptable, especially if you are using sync or mwc GMDs.

If the amount of data to be synchronized is very large, you may want to take the remote machine to the local site, connect it to the local network, and synchronize the data there. This would take advantage of the faster local area network in the local site. Then, when the machine is installed on the remote site, most of the data will have already been mirrored, and the impact of the synchronization process will not be so great.

7.7 Scripts to reestablish the initial configuration

For a number of reasons, at a certain point, you may want to reestablish the initial configuration. In other words, you may want to stop the mirroring across the sites and have the application access the logical volumes again instead of the GMDs.

This section will show some scripts to automate this task. These scripts are based on a naming convention that is different from what was described earlier. If you do not follow this naming convention, you will have to alter the scripts to suit your own needs.

The scripts assume that all state map logical volumes have sm as a logical volume type. You should not remove any GMDs before running the scripts. These scripts do not alter the HACMP and HAGEO configuration.

All script contents are listed in Appendix A. Only the usage will be shown here.

The following steps are necessary to successfully use the scripts:

1. Stop the applications.

   Stop all the applications that are currently accessing the GMDs whose configurations are going to be altered. In cluster domino, we stopped the application domino.

2. Change the state of the GMDs to defined.

   You now have to change the state of the GMDs that are going to be removed (in order to rename the logical volumes).

   To change the GMD state from available to stopped, use the following:

   ```
   # /etc/methods/stopgap -l <gmd_name>
   ```

   To change the GMD state from stopped to defined, use the following:

   ```
   # /etc/methods/ucfggmd -l <gmd_name>
   ```

   Remember that, if you want to change the state of all GMDs, just substitute `-l <gmd_name>` with `-A` in the command lines above.

3. Save volume group information to a file.

   First vary on the volume groups that contain the logical volumes associated with the GMDs on one of the nodes in each site. Then, save the output of the `lsvg -l` command to a file. For instance, in cluster domino, we executed the following command on node black:

   ```
   # lsvg -l localvg > localvg.out
   ```

   And on node blue:

   ```
   # lsvg -l geovg > geovg.out
   ```

4. Execute the `Generate_LV_action.ksh` script.

   This script should be executed on all the nodes where you varied on the volume groups. It does not alter any configuration. It simply generates an intermediate file to let you check or alter the modifications that will be done. On node black, we executed the following:

   ```
   # Generate_LV_action.ksh -i localvg.out -o lvaction.out
   ```

   On node blue we executed the following:

   ```
   # Generate_LV_action.ksh -i geovg.out -o lvaction.out
   ```

   The `-i` option specifies the input file, which was generated in the previous step. The `-o` option specifies an output file. On node black, a listing of the lvaction.out would show the following:

   ```
   # List of LV names generated with Generate_LV_action.ksh
   # on node:  black
   # date:  Thu Jul 24 10:40:37 CDT 1997
   #
   localvg
   #
   #Original          Original   New              New        Action to
   #LV_name           LV_type    LV_name          LV_type    be taken
   # you should not change above this line.
   simba0lv          gmd        simba            jfs        rename
   pumbaa1lv         gmd        pumbaa           jfs        rename
   timon2lv          gmd        timon            jfs        rename
   simba0sm_11       sm         N/A              N/A        remove
   simba0sm_12       sm         N/A              N/A        remove
   pumbaa1sm_11      sm         N/A              N/A        remove
   pumbaa1sm_12      sm         N/A              N/A        remove
   timon2sm_11       sm         N/A              N/A        remove
   timon2sm_12       sm         N/A              N/A        remove
   test              jfs        N/A              N/A        none
   ```

   In the first and second columns, this file lists all the logical volume names and types of the given volume group. The two subsequent columns list the

new names and types for the logical volumes or N/A if this information does not apply. The new names for the logical volumes are the names of the GMDs associated with them. These GMDs will have to be removed in order to rename the logical volumes. This will be done in another step.

The last column shows the action that will be taken. This can be *rename*, to rename the logical volume to the new name and type, *remove*, to remove the logical volume, or *none*, to take no further action.

In the example above, the data logical volumes will be renamed to the associated GMD names, and these GMDs will be automatically removed. The state map logical volumes will be removed. The logical volume test was added to the localvg volume group, and it does not have any GMD associated with it; so, no action will be taken. You may alter this file to change the actions or the new names and types for the LVs. If you do not want any action to be taken to a certain logical volume, just change the action column in the corresponding line to *none*, or remove the line.

5. Execute the Take_action_LV.ksh script.

   This script should also be executed on the nodes where you varied on the volume groups. It takes the file generated in the previous step as input and executes the actions listed there. We executed the following script command on nodes black and blue:

   ```
   # Take_action_LV.ksh -i lvaction.out
   ```

   Notice that the lvaction.out file is different between the nodes. Each node generated its own file.

   This script automatically removes the GMDs that will have logical volumes renamed to their names, renames the LVs, and removes the state map logical volumes. It also generates another script, Remove_GMDs.ksh, in each node. This script will be covered in the next step.

   Before continuing, vary off the volume groups in the respective nodes. We varied off localvg in node black and geovg in node blue.

6. Execute the Remove_GMDs.ksh script.

   This step should be done in the nodes where you did not vary on the volume groups.

   One node in each site executed the Take_action_LV.ksh script. As stated earlier, this script generates the Remove_GMDs.ksh script, which you should now transfer to all the other nodes. The generated Remove_GMDs.ksh should be the same in all the nodes. This would not be true only if you had removed or altered the GMD configuration on one of the nodes.

You should give root execute permission to this script and execute it on the nodes to which it has been transferred. In cluster domino, we transferred the Remove_GMDs.ksh script to brown, gave it root execute permission, and executed it.

7. Export/Import the volume groups on the remaining nodes.

   This step should only be done on the nodes that did not vary on the volume groups.

   Export the volume groups that contained the GMD and state map logical volumes on the nodes and import them again to update the changes made in the volume groups in the previous steps.

Chapter 8. Failure and recovery

In this chapter, we will discuss the strategy for handling general failures in an HAGEO cluster or a GeoRM configuration.

8.1 GeoRM general failure

We will talk about some general failures of GeoRM, such as disk failure, machine failure, and geographic network failure. In GeoRM, we need more manual intervention than HAGEO when we do the recovery. The disk failures and machine failures sometimes will be promoted to site failure.

8.1.1 Machine failure and site failure

Disk failures, CPU failures, memory card failures, adapter failures, or other failures may invoke a machine failure. If all machines in one site are unavailable, it is a site failure.

Local machine failures and remote machine failures are handled differently. Various factors affect machine failure recovery. The procedures vary depending on whether the machine owns the primary or secondary GMD devices and whether there is another machine at the site that can be configured to assume the role of the failed machine.

A machine failure can be intentional (for system maintenance purposes) or unexpected. The following recovery procedures apply to either of these scenarios.

8.1.1.1 Local machine failure with takeover at the same site

At first, when you think about the local machine failure with takeover at the same site, the following prerequisites should be ready:

- GeoRM is installed and configured on the takeover machine. The takeover machine must be listed as a site member in the GeoManager configuration component.
- GeoMessage started on the takeover machine.
- The GMD is configured on the takeover machine and in the DEFINED status.
- The disks containing the primary GMD device on the failed machine, that is, the GMD local logical volume and state map logical volume, must be accessible by the takeover machine.
- The application is installed on the takeover machine.

Under this environment, if the local machine (primary machine) goes down for some reason, another machine at the same site can be configured to take over the role of the failed machine. We need to do the following procedure manually:

1. Stop the GMDs on the remote machine. This will protect against any inadvertent errors on the remote site causing possible data divergence.

   ```
   # stopgmd -l gmdname
   # ucfggmd -l gmdname
   ```

2. Attach the failed machine's disks to the takeover machine. The data can reside in a shared volume group between the machines, or the disks can be manually attached to the takeover machine. Vary on the volume group(s) containing the GMD logical volumes.

3. Configure and start the GMDs on the takeover machine.

   ```
   # cfggmd -l gmdname
   # gmddown -l gmdname remote_machine
   # startgmd -l gmdname
   ```

4. Configure and start the GMDs on the remote machine

   ```
   # cfggmd -l gmdname
   # startgmd -l gmdname
   ```

5. Start the application on the takeover machine

8.1.1.2 Local machine failure with takeover at the remote site

A machine at the remote site can also be configured to take over the role of the failed machine. This procedure involves switching the remote machine's GMD devices' role from secondary to primary because data can only be written to the primary device. This procedure can make the application temporarily available until the local machine comes up again. During this time, no mirroring occurs.

The same application should be installed on the remote takeover machine.

When the local machine fails and no local takeover is available, perform the following steps to start the application on the remote machine:

1. Stop and unconfigure the relevant GMDs on the remote machine. Other GMD devices on the remote machine can be left active:

   ```
   # stopgmd -l gmdname
   # ucfggmd -l gmdname
   ```

2. Switch the GMD devices' roles from secondary to primary.

   ```
   # chdev -l gmdname -a device_role=primary
   ```

3. Configure and start the GMDs on the remote machine:

```
# cfggmd -l gmdname
# gmddown -l gmdname remote_machine
# startgmd -l gmdname
```

Now, the application can be started on the remote machine. Staleness marking will begin on the remote machine GMDs.

Once the local machine becomes available again, it can resume in the secondary or primary role (depending on how you set the GMD device roles).

If you want to resume the geographic mirror as soon as possible and do not want to intervene with the application now, you can do the following:

1. Change the device role of the local GMD from primary to secondary:

```
#chdev -l gmdname -a device_role=secondary
```

2. Configure and start GMD. Then, the mirror is resumed:

```
#cfggmd -l gmdname
#startgmd -l gmdname
```

But, if you want to switch the application to local machine now, perform the following steps:

1. Stop and unconfigure the relevant GMDs on the remote machine. Other GMD devices can be left active:

```
# stopgmd -l gmdname
# ucfggmd -l gmdname
```

2. Switch the GMDs device role from primary to secondary.

```
# chdev -l gmdname -a device_role=secondary
```

3. Configure and start the GMDs on the remote machine:

```
# cfggmd -l gmdname
# gmddown -l gmdname remote_machine
# startgmd -l gmdname
```

4. Configure and start the GMDs on the local machine:

```
# cfggmd -l gmdname
# startgmd -l gmdname
```

Then, the data will be synchronized from the remote machine to the local machine. Remember, in this situation, the correct GMD start sequence is from remote to local because the remote machine owns the stale state map.

8.1.1.3 Local machine failure with no takeover

A local machine does not have to have another machine that can assume its role. In this case, the application is unavailable until the local machine comes back online. Once the local machine is available again, resume mirroring by starting the GMD(s) and the application.

- While the local machine is unavailable, stop the GMDs on the remote machine. This will protect against any inadvertent errors on the remote side.

```
# stopgmd -l gmdname
# ucfggmd -l gmdname
```

- When the local machine comes back on line, vary on the volume group(s) that contain the GMD logical volumes on the local machine.
- Start GeoMessage on the local machine:

```
# cfgkrpc -ci
```

- Configure and start the GeoMirror Device(s) on the local machine:

```
# cfggmd -l gmdname
# gmddown -l gmdname remote_machine
# startgmd -l gmdname
```

- Configure and start the GMD(s) on the remote machine:

```
# cfggmd -l gmdname
# startgmd -l gmdname
```

8.1.1.4 Remote machine failure with takeover at the same site

If a machine failure happens at the remote site, another machine at that site can be configured to take over the role of the failed machine. This enables mirroring to resume on the takeover machine.

The following configuration requirements apply:

- GeoRM is installed and configured on the takeover machine. The takeover machine must be listed as a site member in the GeoManager configuration component.
- The GMD is configured on the takeover machine and is in the DEFINED state.
- The takeover machine is configured as an additional remote peer for the relevant GMD(s).
- The disk(s) containing the secondary GMD device on the failed machine, that is, the GMD local logical volume and state map logical volume, must be accessible by the takeover machine.

When the remote machine fails, perform the following steps to resume application mirroring:

1. Attach the failed machine's disks to the takeover machine. The data can reside in a shared volume group between the machines, or the disks can be manually attached to the takeover machine. Vary on the volume group(s) containing the GMD logical volumes.
2. Configure and start the GMDs on the takeover machine, but first, make sure GeoMessage is running.

   ```
   # cfggmd -l gmdname
   # startgmd -l gmdname
   ```

   This may take a while depending on how long the application was running while not being mirrored. There may be a lot of data to synchronize to the remote machine.

   The application can now be started on the remote site and will run with mirroring.

8.1.1.5 Remote machine failure with no takeover

You do not have to configure multiple remote peers for a given GMD. Depending on the application, the surviving local copy of the data may continue to change while the mirror copy is unavailable. When the remote machine reintegrates, the state map is processed to synchronize the data on the devices.

Do the following on the failed remote machine to reintegrate it:

1. Vary on the volume group(s) containing the GMD logical volumes.
2. Start GeoMessage:

   ```
   # cfgkrpc -ci
   ```

3. Configure and start the GMDs:

   ```
   # cfggmd -l gmdname
   # startgmd -l gmdname
   ```

   This may take a while depending on how long the application was running while not being mirrored. There may be a lot of data to be synchronized to the remote machine.

8.1.2 Disk failure

As described before, we highly recommended that AIX LVM mirroring and RAID technology be implemented. Under this environment, when one of the disks is unavailable, the application can continue to operate. If the disk is hot swap, we do not need to shut down the machine when we replace the failed

disk; otherwise, we have to shut down the system, replace the disk, and restart. If there is no LVM mirroring and RAID support, the disk failure needs to be prompted to machine failure or site failure if you want another machine or site to take over.

8.1.3 Global network failure

A global network failure is the failure of all the geographic networks. In this situation, the Promote Failure Timeout option takes effect. You must be sure you have a non-zero value to handle global networks failure. A Global network failure in a GeoRM configuration is represented in Figure 113.

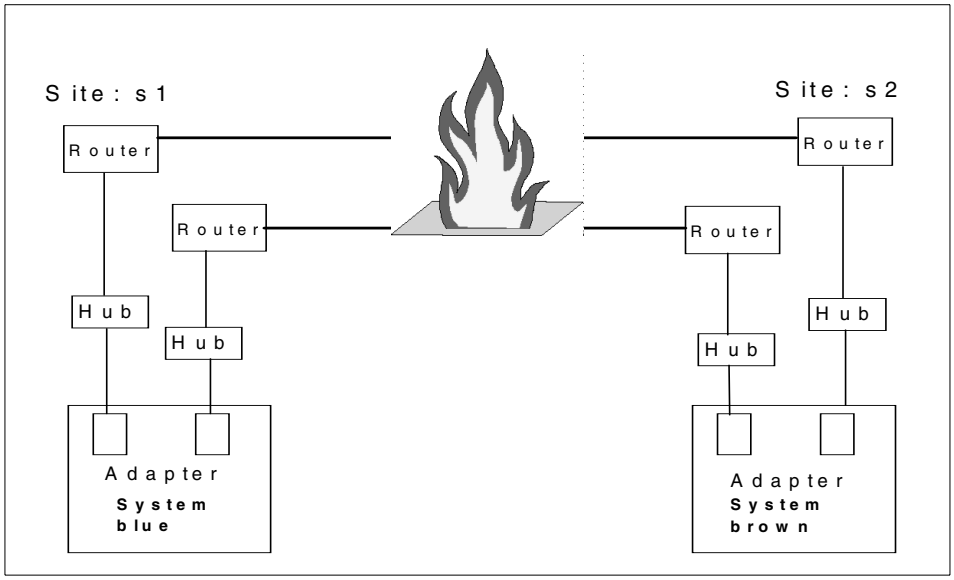

Figure 113. Global network failure in a GeoRM configuration

When a global network failure occurs, the primary GMD devices will stop trying to make remote writes after a period of time based on the Promote Failure Timeout value. Their state maps are marked dirty, and writes on the local machines continue. These writes may all be occurring at one site or at both depending on whether you have an Active-Backup or Mutual Backup configuration. Once communications are reestablished, do the following on the remote machines to synchronize the data:

1. Make sure GeoMessage is running:

   ```
   # genkex |grep krpc
   ```

   if it is not, use the following command to start it:

```
# cfgkrpc -ci
```

2. Configure and start the GMD(s).

```
# cfggmd -l gmdname
# startgmd -l gmdname
```

Then, begin to synchronize data.

8.2 HAGEO general failure

In this section, we discuss several general failure situations that HAGEO handles and provide a strategy and some techniques for failure recovery.

8.2.1 Local failures

The following sections cover various local failures.

8.2.1.1 Redundant disk failure

If logical volumes are mirrored using LVM mirroring or if a failed disk is a part of RAID subsystem, the failure of a disk component is managed by the LVM mirroring or RAID function.

In the LVM mirroring case, you could recover the disk failures as follows:

1. Unmirror all logical volumes that have mirrored data on the failed disk component.
2. Remove the failed physical volume from the associated volume group.
3. Physically replace the failed disk component.
4. Add a replacement physical volume to the volume group.
5. Synchronize all logical volumes that have one copy in that physical volume, or synchronize the whole volume group.

In the case of the RAID subsystem, you can recover the failure by doing a hot-swap of the failed disk component, or you can replace the failed disk by following the procedure described in the service manual.

8.2.1.2 Redundant adapter failure

A storage device adapter failure (SSA adapter for example) is handled by the AIX device driver level transparently if there is another adapter that services the same function and should be handled by HACMP properly. To fully recover from these situations, you need to shut down the node and physically replace the failed adapter. This is equivalent to a planned site failure in a one-node-per-site configuration or a planned node failure in a

more-than-one-node-per-site configuration. To recover from a site failure situation, refer to the following.

Node failure in a more-than-one-node-per-site configuration
If node failure occurs in a more-than-one-node-per-site configuration, this node failure is managed by HACMP. You can recover from this situation as follows:

1. Confirm that necessary HACMP resources have been taken over by the local peer node and shut down the node.
2. Physically replace the failed node component(s), and then power on the node.
3. Start HACMP on the node.
4. If the resource group is configured as cascading, confirm that the resources determined by the resource group type are reacquired by the rejoining node.

8.2.1.3 Non-redundant component failures

In most cases, these types of failures cause (or should be customized to cause) a site failure in a one-node-per-site configuration or a node failure in a more-than-one-node-per-site configuration. However, there is one special case that is worth explaining because of its complexity. It is the case of a disk component failure where there is no mirror.

Physical volume failure with no mirroring
This situation should be prevented by always using LVM mirroring or RAID technology. However, in a real situation, if it is impossible and the failure is actually encountered, you can use the following method to recover.

For example, suppose a physical volume that contains one of the GeoMirror devices fails on node blue in non-dominant site s2 (see Figure 114 on page 255). Unfortunately, there is no LVM mirroring on the data logical volume of the GeoMirror device. The failure causes a loss of the GeoMirror device on the non-dominant site. Without an AIX error notification method customization, this failure is not detected by HACMP. Therefore, it does not generate a node failure or site failure.

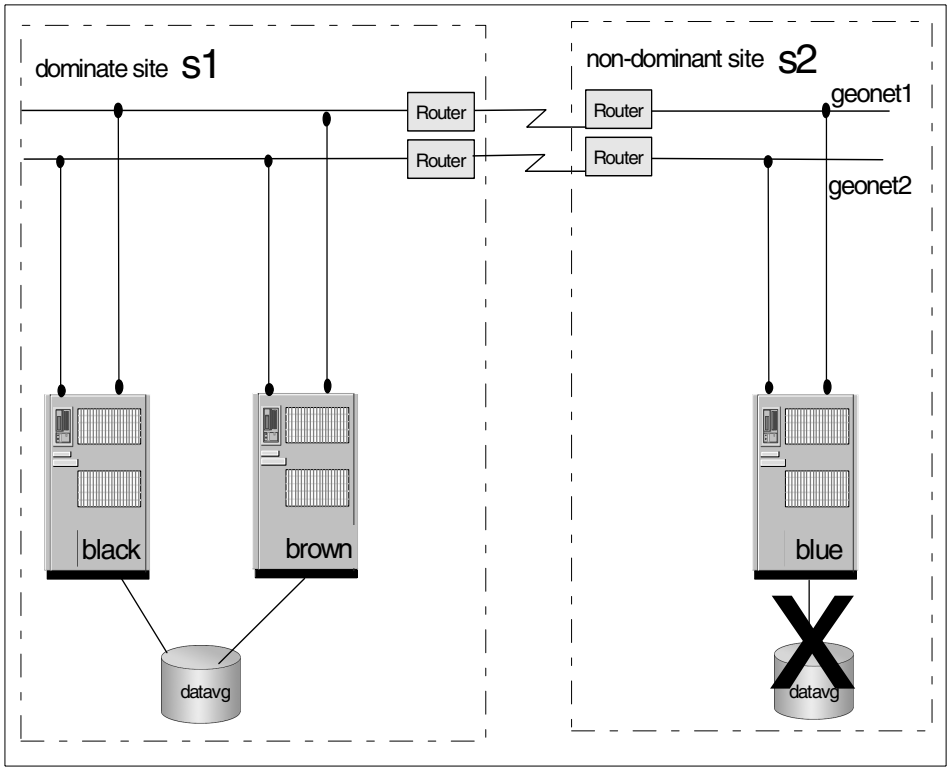

Figure 114. Physical volume failure with no mirroring

To recover from this situation, we can perform the following steps:

At site blue (or remote site)

1. Make sure you have documented all the GMDs configurations including their logical volume relationships. This can be done by capturing the output of a geoverify.

   If the GMDs whose data is located on the failed disk are still in available status, stop them.

   ```
   # stopgmd -l gmdname
   # ucfggmd -l gmdname
   ```

2. If possible, remove all logical volume definitions from the failed disk.

   ```
   # rmlv -f lvname
   ```

3. Remove the failed physical volume from its volume group.

   ```
   # reducevg vgname pvname
   ```

4. Physically replace the failed disk.
5. Add the new disk as a physical volume to the volume group.

   ```
   # extendvg vgname pvname
   ```

6. Re-create the logical volumes for the GeoMirror devices and state maps.

At site black (or local site)

Before starting the GeoMirror devices at blue, you need to dirty the state maps for them on the **surviving node** (where the disk failure did not occur) to force the synchronization. In this example, this step is done on node black. To dirty the state map, use the following command:

```
# gmddirty -l gmdname
```

At site blue

Start the GeoMirror devices. This may take a long time to complete, depending on the amount of data to synchronize across the sites (see “Pre-starting GMDs on a rejoining node” on page 262).

```
# cfggmd -l gmdname
# startgmd -l gmdname
```

8.2.2 Site isolation

The algorithm that distinguishes between site isolation and site failure is very complicated. There are two cases for a site isolation situation as follows:

- All primary GeoNetworks are unavailable on one node only.
- All primary GeoNetworks are unavailable on all nodes.

We will refer to the first case as *Node Isolation* and to the second case as *Real Site Isolation* to distinguish between them.

8.2.2.1 Node isolation

If a node has a non-Primary GeoNetwork (this can be a secondary GeoNetwork connected to a remote peer node or a serial network connected to a local peer node) that can send and receive heartbeats, and that node loses all network connections to the primary GeoNetwork, it will be in a n*ode isolation* situation. In this case, the node kills itself to abandon the geographical resources because it has no way of servicing them across the sites. As a result of this suicide, a local peer node will take over the geographical resources if there is one configured with HAGEO resource

groups. If there is no such local peer node, a remote peer node will take over on the other site as shown in Figure 115.

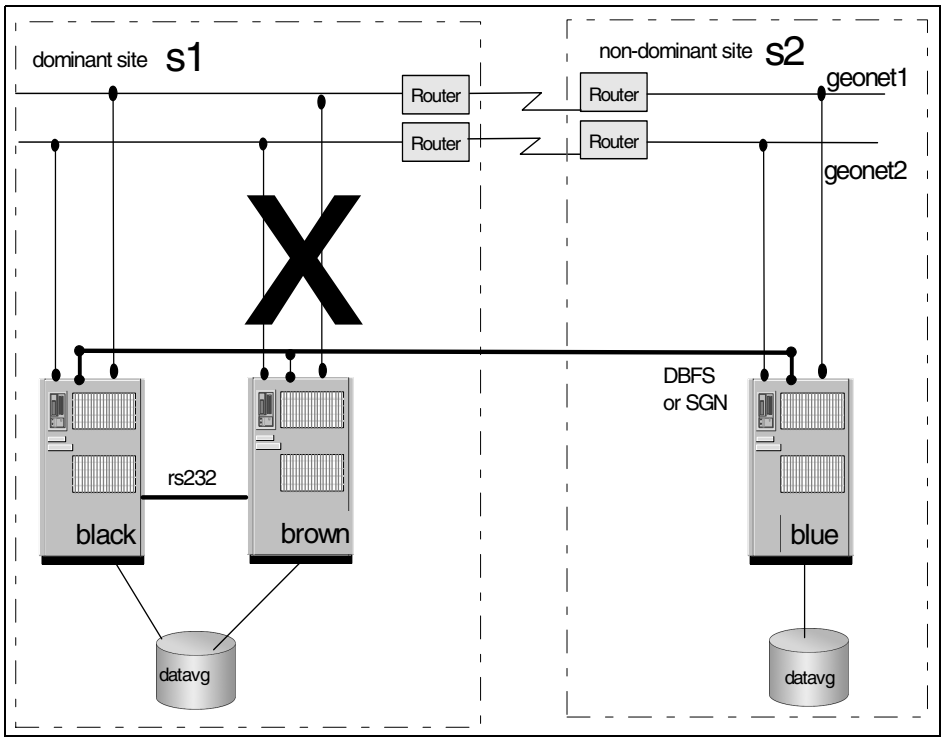

Figure 115. Node isolation

In this situation, all the geographic networks of node brown are unavailable, but brown is still able to send and receive heartbeat messages via non-geo network; so, brown will kill himself to free the geographic resources, and node black will take over as configured.

We can recover from this situation as follows:

1. Restore the Primary GeoNetwork communication path to the failed node.
2. Boot the failed node and test communications on the Primary GeoNetwork.
3. Start HACMP on this node.
4. Confirm the successful return of the geographical resources to the rejoining node.

Obviously, there may be considerable downtime for the geographical resources during reintegration depending on how much data needs to be

synchronized when the GMDs start up. If there is a large amount of data, remember to use the Geo_prestart_gmds facility.

8.2.2.2 Real site isolation

If the last Primary GeoNetwork fails but at least one node on the non-dominant site is still able to send heartbeats over an SGN or successfully call and log in to a node on the dominant site using DBFS, GeoManager determines that there is *real site isolation*. It halts all nodes on the non-dominant site gracefully to avoid data divergence.

Data divergence occurs when data continues to be updated at both sites while geomirroring is not possible (see Figure 116 on page 259). This can happen when all networks fail between sites, but the nodes at each site are still up. Without a secondary checking mechanism, such as SGN or DBFS, each site's nodes will think that the other site's nodes are down, and will start to take over resources and to bring up applications. In the geographic mirroring environment, you could have a node at one site running an application that updates one side of a GMD mirror, and a node at the other site running the same application that updates the other side of the same GMD. The two sides of the mirror will start to differ, and there will be no way to merge them. Data divergence is a very dangerous situation and must be avoided at all costs in a HAGEO cluster. This is why the nodes on the non-dominant site are immediately shut down once site isolation is detected.

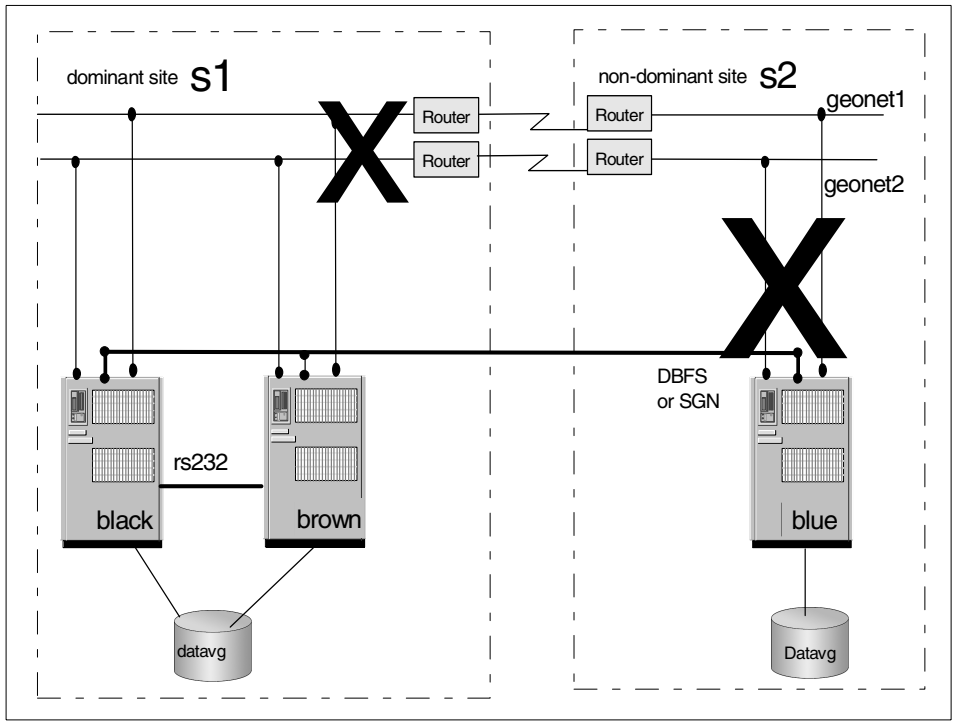

Figure 116. Real site isolation

After using the `gmd_show_state` command to check and you are sure there was no data divergence across the sites, you can recover from this situation as follows:

1. Restore the primary GeoNetwork communication path on all nodes.
2. Power-on nodes on the non-dominant site.
3. Start up HACMP on these nodes.
4. Confirm that the synchronizing of GMDs has completed successfully.

GeoMirror devices on a rejoining node try to synchronize their GeoMirror devices from a remote surviving node. This process is done automatically unless there is a data divergence.

If data divergence is found on a GeoMirror device, startup of the GeoMirror device will fail. GeoMirror can detect a difference of data between mirrors for this GeoMirror device, but it cannot determine which data is correct (we called this correct data GeoMirror *Master*). Therefore, this GeoMirror startup failure means there is a requirement for manual intervention by the system

administrator. To manually intervene is to *unify the state map*. You need to unify the state map associated with this GeoMirror device before you can successfully start it.

To see how to unify a state map, refer to Section 8.3.5, “Unifying state maps” on page 271.

For recovery from a site isolation situation, we strongly recommend using gmd_show_state or smit to check for possible data divergence in all GeoMirror device state maps before allowing the node to try to rejoin the HAGEO cluster. This will prevent the GeoMirror device from failing when it rejoins the node.

8.2.3 Site failure

If the last Primary GeoNetwork fails and the node is not able to send heartbeats over an SGN or successfully call and log in using DBFS, GeoManager determines that there is a site failure. At this point, the other site takes over the HAGEO resources. GeoManager sends the appropriate warning messages to the system administrator at the surviving node (see Figure 117 on page 261).

After a site failure, all write requests to a GeoMirror device occur only locally on the surviving node(s), and these requests change the state map state for the cell involved to stale.

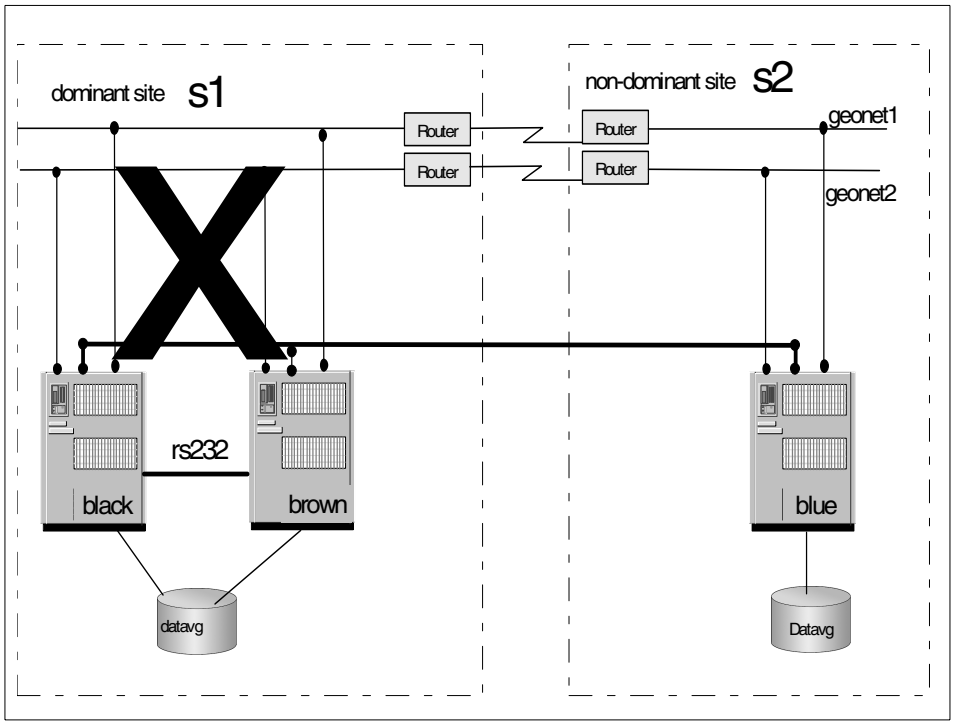

Figure 117. Site failure

8.2.3.1 Short-term site failure

Communication failures, scheduled or unscheduled machine reboot, power failures, and other short-term failures that affect all the nodes at a site are also handled as site failures by HAGEO. You can recover from these situations as follows.

1. Remove the cause of the site failure.
2. If it is powered-off, power-on the node(s).
3. Check whether there is data divergence. If it exists, you will need to unify the state maps. See the Section 8.3.5, “Unifying state maps” on page 271.
4. Pre-start the GeoMirror device on the rejoining node before starting its HACMP (which causes it to start to join the cluster). In many cases, this will take a long time depending on the amount of data to synchronize across the sites (see Section 8.2.3.2, “Pre-starting GMDs on a rejoining node” on page 262).
5. Start HACMP on the rejoining node.

8.2.3.2 Pre-starting GMDs on a rejoining node

In a cascading resource group, the owner node holds the HAGEO resources that are comprised of geographic applications or file systems created on GeoMirror devices. In the event of a failure of the owner node, the next-highest priority node varies on the volume groups, starts the GeoMirror devices and takes over the HAGEO resources (mounts the file systems and/or starts the geographical applications). When the owner node rejoins the cluster, the nodes that have taken over the resources will release them. The nodes that have taken over the resources stop the applications, stop the GMDs, vary off the volume groups, and unmount the file systems.

If the resources failover across the geography, it may take a considerable amount of time for a reintegration of the owner node due to the need to synchronize the GeoMirror devices. The takeover node, blue, cannot start the applications until the state map and mirrors are back in synchronization between black and brown. The start of GeoMirror devices is handled by the Geo_start_server HAGEO event script that is defined as a post-event of the node_up HACMP event. If it takes more than six minutes to start the GeoMirror devices, the config_too_long event is triggered. More importantly, the users who were using the geographical application would be interrupted until the application comes up again on black. The time it takes for the mirrors to synchronize depends on the amount of data that was written or updated on node blue while node black was down. This could turn out to be a long time. If you start your GMDs as a by-product of starting HACMP on your rejoining node, the applications stop on the takeover node as it releases the resources in preparation for them to go to the rejoining node. However, the joining node cannot start them up until it has fully synchronized its GMDs. While the synchronization is happening, the applications are down. This can have a significant impact to the users.

Therefore, it is better for you to start the GMDs on the rejoining node before starting HACMP. This would allow the synchronization to go on while the application remains up and running on the takeover node. Remember, it does not release the resources until the owning node starts to join the cluster. After the synchronization has completed (which could, but does not necessarily have to, take a lot of time), you can then start HACMP on the rejoining node. There will be a short interruption of the application while it stops on the takeover node and immediately starts on the rejoining node.

A script called Geo_Prestart_gmds is provided with HAGEO. It can be run by the user from the command line as a way of starting GeoMirror devices (and, therefore, do the synchronization) before starting a node trying to join the cluster. The script will vary on the appropriate volume groups, configure

GeoMirror devices, and start the GeoMirror devices. This will allow the synchronization of all data before the application services on the takeover node are stopped.

If this is a node failure and not a site failure, only the volume groups that can be accessed by the node ready to rejoin will be varied on. If a volume group is varied on by the local peer, synchronization will not be an issue since no staleness will have occurred. Asynchronous GMD devices that do not failover locally should be in a separate volume group and, thus, available for access by the recovering node.

Before starting HACMP for AIX on brown (the joining node), run the following command making sure the output is appended to the /tmp/hacmp.out log:

```
# /usr/sbin/gmd/scripts/Geo_prestart_gmds | tee -a /tmp/hacmp.out
```

When the script completes (the time required is dependent on the amount of data to be synchronized and the speed of the network), HACMP for AIX can be started on black.

If you would like the pre-starting of GeoMirror devices to be automated, you must have a /.rhosts file on each machine in the HAGEO cluster, and you must edit the /usr/sbin/gmd/scripts/Geo_remote_node_up script manually. This script, as it is delivered with the HAGEO product, has an `exit 0` command as the first line. Therefore, by default, it does not do anything. If this first line is removed or commented out, the script will automate the pre-starting of the GeoMirror devices at the joining node to minimize application downtime during reintegration. This script makes calls that run `rsh` commands. This requires that the `/.rhosts` file be in place on the joining node, and it must include the node that has taken over the resource.

8.2.3.3 Long-term site failure (catastrophic failure)

In the case of a long-term site failure, HAGEO works as described in the previous section. This site failure is usually caused by some more serious event. The minimal cause might be an electrical failure involving the entire site, and, in the worst case, the entire site might be permanently disabled or destroyed by fire, flood, or an earthquake.

After the disaster site is reconstructed (or constructed in another location), you can use the following steps to recover:

1. Install all the necessary hardware.
2. Install all the necessary software.
3. Set up the network and make sure it works properly.

4. Reconfigure your entire HACMP and HAGEO configuration as it was set up before with any changes applied since the outage.
5. All GeoMirror devices should be marked completely dirty on the surviving node (see Section 8.3.3, “Dirtying a state map” on page 270).
6. Synchronize all GeoMirror devices manually (see Section 8.2.3.2, “Pre-starting GMDs on a rejoining node” on page 262).

Note

If you can install your replacement node at the surviving site temporarily, this will save initial synchronization time since using local networks (Ethernet or token-ring) supplies a much faster communication path than using the primary geographical networks across sites.

After this initial synchronization using local networks, you can stop the GeoMirror devices on this node, then move them to the reconstructed site. As a result of this work, there will only be a small amount to synchronize across the sites (only the updates from the time you shut down the replacement node at the surviving site until you started it up at the reconstructed site).

7. Start HACMP on the node at the reconstructed site, and let it rejoin the HAGEO cluster.

To accomplish this recovery method, you need to make sure you maintain the following documentation of your cluster.

- Documentation used when the cluster was initially designed and configured should be maintained to reflect the current environment.
  - Diagram and label the cluster.
  - HAGEO configuration worksheets from Appendix A of the *High Availability Geographic Cluster V2.1 for AIX: Planning and Administration Guide*, SC23-1886.
  - Also document the AIX configuration.
- Daily operational procedures documentation.
  - Cluster startup and shutdown.
  - Cluster monitoring.
  - Application considerations.
  - Error messages - what they mean and what to do.
- A tested and well-documented recovery plan.

- How to make applications and resources available if the cluster will not start.
- How and when to reintegrate a failed node back into the cluster.
- How to recover from hardware failures (such as disks).

- A cluster snapshot will assist in gathering information on the cluster configuration and the AIX environment.

8.2.4 Site isolation/failure detection logic

Due to the complexity of the site isolation/failure detection logic, this logic is illustrated in Figure 115 on page 257 through Figure 117 on page 261. Without a complete understanding of this logic, it will be unlikely that you will follow the correct recovery actions.

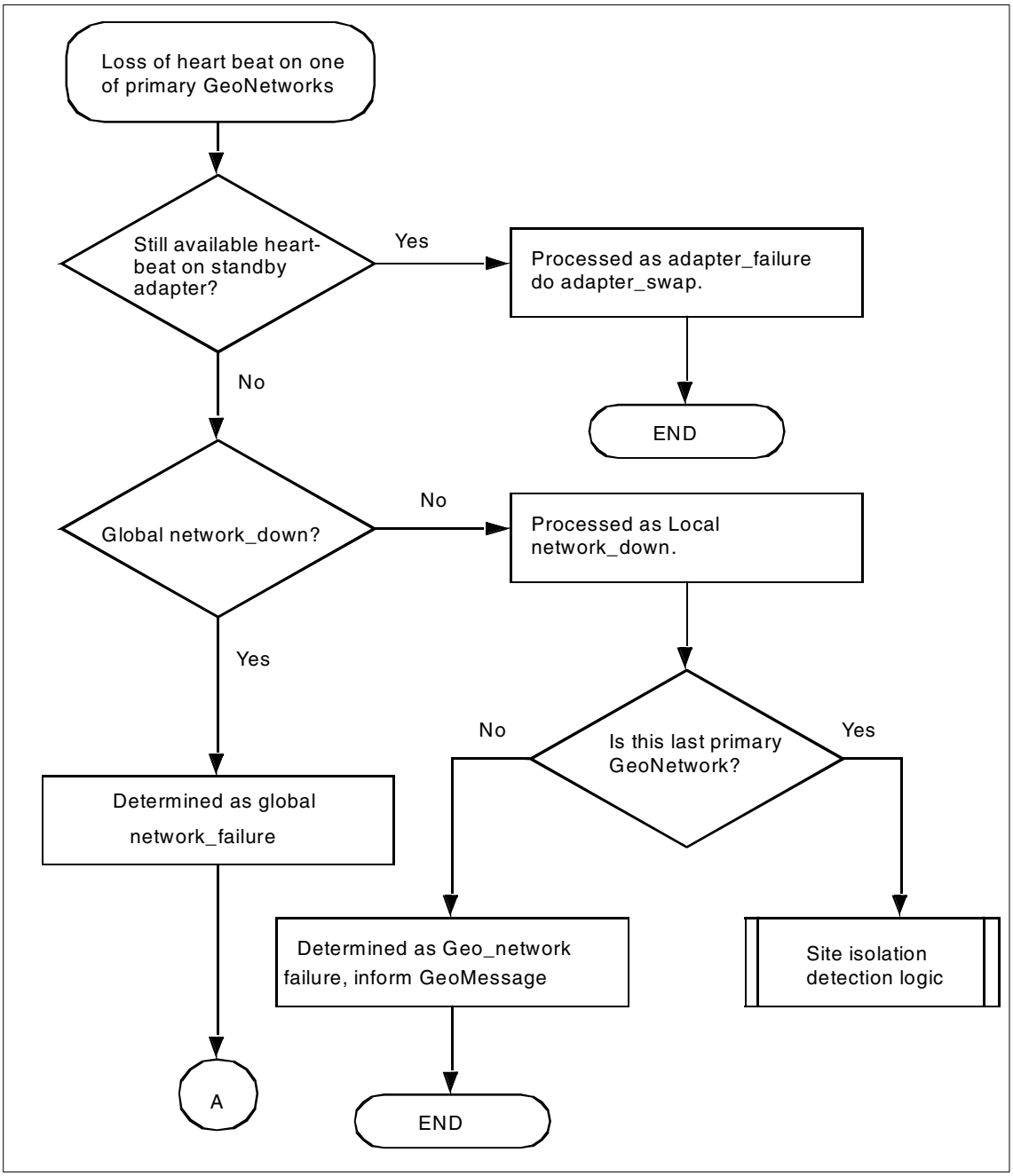

Figure 118. Site isolation/site failure detection logic (Part 1)

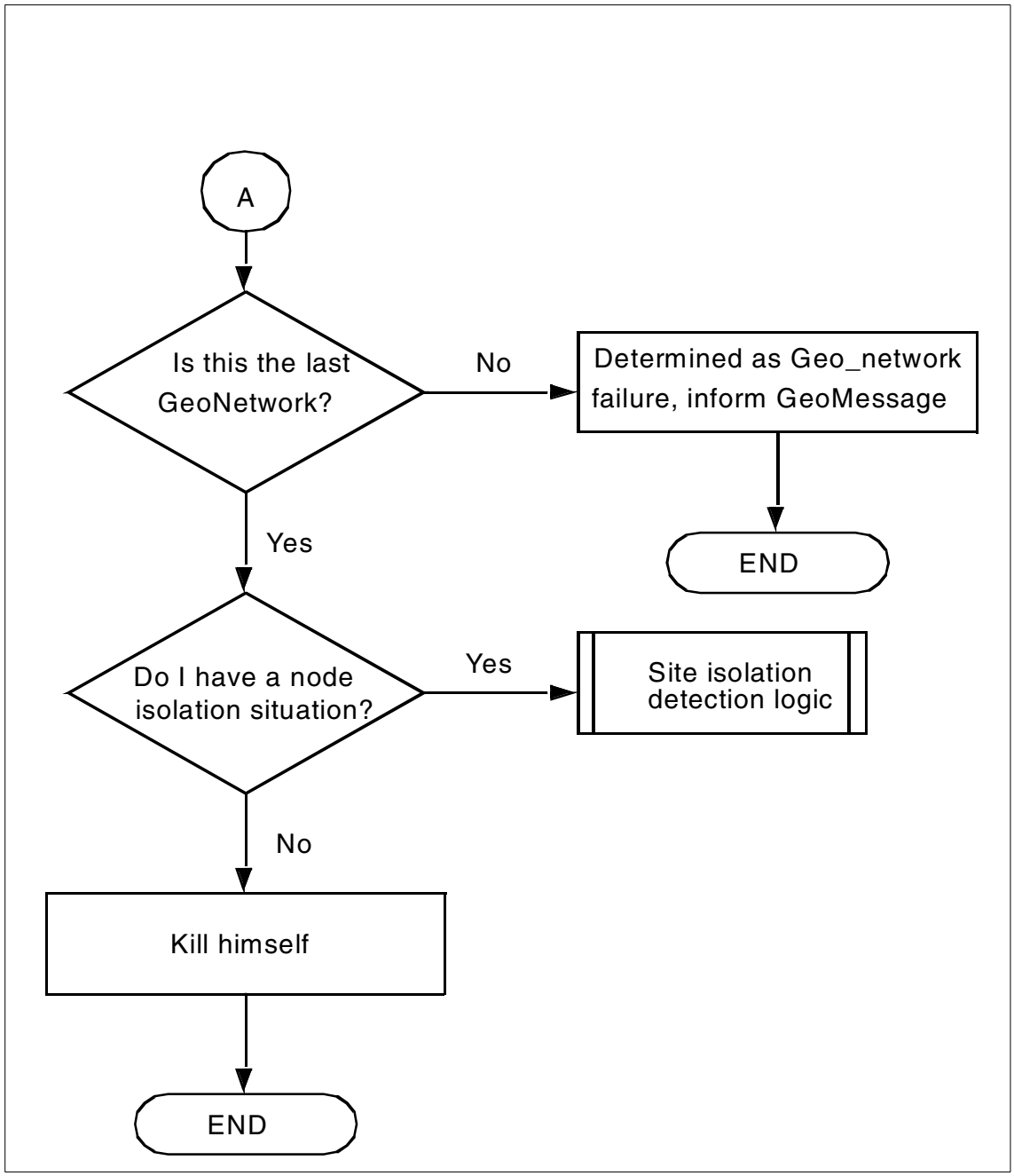

Figure 119. Site isolation/site failure detection logic (Part 2)

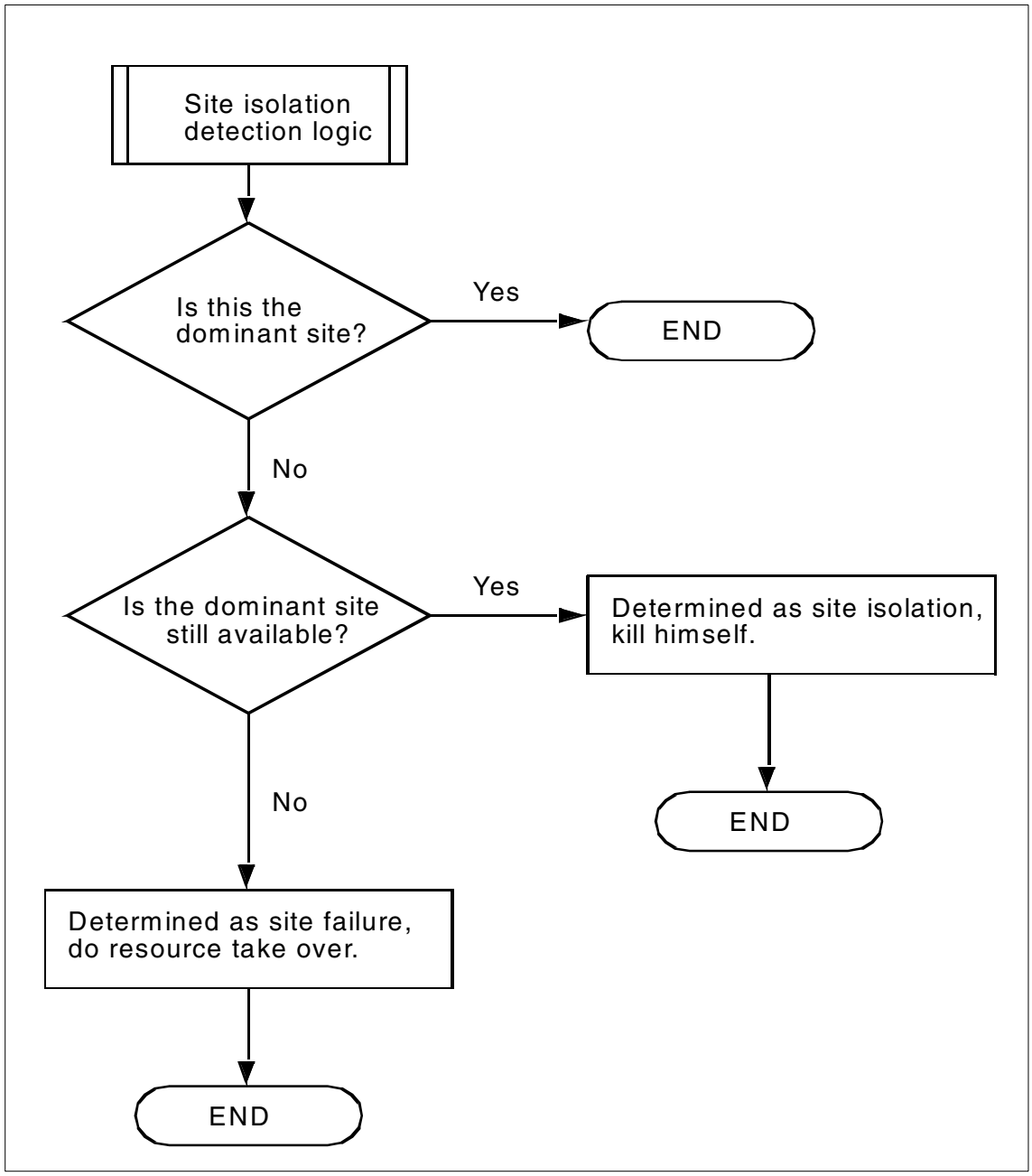

Figure 120. Site isolation/site failure detection logic (Part 3)

8.3 State map managing utilities

In this section, we will discuss some useful state map managing commands; some of them should be used by very carefully.

8.3.1 Viewing the state map status

To view state map status, we can use the `gmd_show_state` command, which reads state maps from different sources depending on the various environment:

- GeoMirror device driver is not loaded: State map is read from disk.
- Local mirrored device is on line but unavailable to applications: State map is read from kernel.
- sync process in operation: State map of all regions is being synchronized.
- Active mirroring in progress: Snapshot of state map is read from kernel.

To view state map information, we can issue the following command:

```
# /usr/sbin/gmd/gmd_show_state -l gmdname
```

An output example of the `gmd_show_state` command is shown in Figure 121.

```
Point of View: Node delta
--------------

Point of View GMD List:
-----------------------

Name: test
Status: AVAILABLE

State Map:

 Cell     Value
-------   -----
      0   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
*
1048576
```

Figure 121. Output example of the gmd_show_state command

Where * means the value of the line above is repeated.

We can also use smit.

8.3.2 Dumping a state map

We can dump a machine readable state map of a gmd to a file and convert the file to human-readable form. For example:

1. Dump a machine readable mapfile of gmd test, and save the file to /tmp/mapfile:

```
# gmd_show_state -D -l test -M /tmp/mapfile
```

2. Then, convert the /tmp/mapfile to a human-readable file:

```
# gmd_show_state -M /tmp/mapfile -m >/tmp/testmap
```

The output is shown in Figure 122.

```
Mapfile: /tmp/mapfile
Site: s2
Node: delta
Device: test
State Map size: 524288 byte(s)
State Map Cell size: 4 bit(s)
State Map data regions: 32768 byte(s)
State Map type: HOST
Remote Peers: beta


State Map:

 Cell     Value
-------   -----
      0   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
```

Figure 122. Output of the state map mapfile dumped

8.3.3 Dirtying a state map

Basically, the *dirty state map* process changes the status of all cells of a state map to stale (value 0xf) on one node. Once the state map is marked as dirty, the GeoMirror device associated with this state map will synchronize all of its data blocks from the node where the dirty state map operation was run to the other site node when it starts up its GMD. Ensure that the remote GeoMirror device is unavailable. To dirty a state map, we can use the following command or smit:

```
# /usr/sbin/gmd/gmddirty -l gmdname
```

8.3.4 Cleaning a state map

We might use this function in the test stage. We recommend that you never use it after the system has been in production; otherwise, it will cause data loss.

To clean the state map, you can use the `gmdclean` command or smit. This command is executed on the node that has the *incorrect* state map status:

```
# /usr/sbin/gmd/gmdclean -l gmdname
```

This command cannot be performed while a remote GMD is still available. Again, the execution of this command requires extreme caution; you should really understand what you are doing.

8.3.5 Unifying state maps

If the state maps of one GMD are not clean on both sides, data divergence has occurred. To recover from this situation, we need to unify the state map. Before unifying the state maps, we must decide which state map should be used.

We can preview the unification of all the specified state maps before we actually perform the unification:

```
# gmd_show_state -U -l gmdname -V
```

To unify the state map, issue one of the following commands:

```
# gmd_show_state -U -d destination -l gmdname
```

or

```
# gmd_update_state -U -d destination -l gmdname
```

We will discuss this when we use it in the next section.

8.4 Data divergence

Data divergence is a situation in which both mirror copies for a given GeoMirror device are being updated independently. This will result in two versions for a given cell of data. There may be no way to tell which is the most valid - a very bad situation indeed.

Data divergence may happen when a geographical application writes data on both sites while all primary GeoNetworks are unavailable. This might happen in the case of site isolation, which we discussed earlier. In site isolation, the nodes at each site think the nodes at the other site are down because they

are no longer able to receive keepalive packets on their primary GeoNetworks. Therefore, the site that was running the geographical application continues running it marking its state map as stale for any cells it updates. Meanwhile, the other site, as part of its takeover processing, starts the same application and begins running it against its copy of the GeoMirror Device marking its state map as stale for those cells it updates. The two versions of data begin to differ, and you have data divergence.

We will begin to illustrate data divergence starting with Figure 123. Where there is a white rectangle in the state map LV, it means a consistent (0x0) state map cell. A white rectangle in the data LV means there is no difference between data blocks across sites for this section of data.

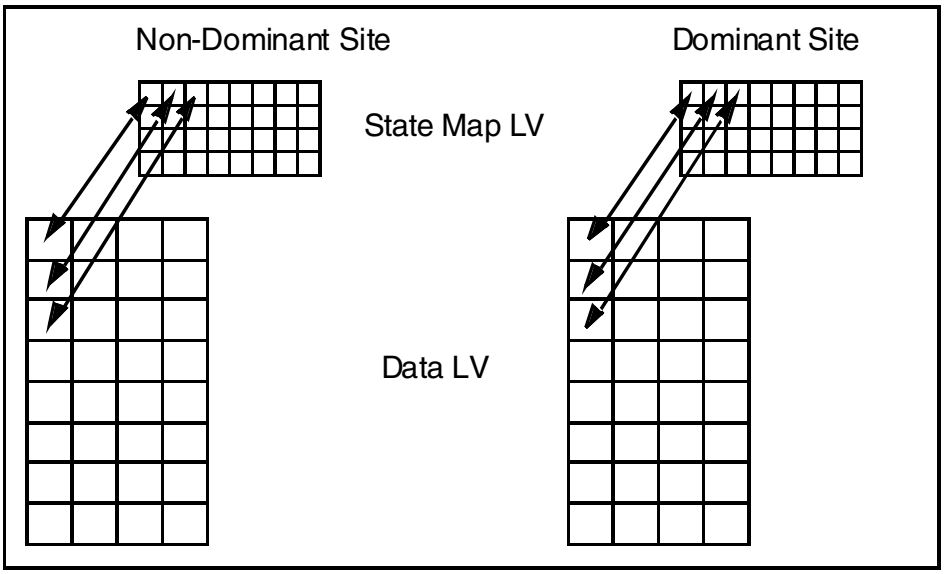

Figure 123. State map status before data divergence situation

The GeoManager halts the non-dominant site node as soon as possible after it detects a site isolation condition. However, until the node shutdown is completed, there is a possibility that data divergence has occurred. If it has occurred, the state map status might display something like that shown in Figure 124 on page 273. In this figure, a hatched rectangle in the state map LV represents a stale (0xf) state, and the A and B characters in the data LV represent data updates written since the GeoMirror devices lost contact with one another.

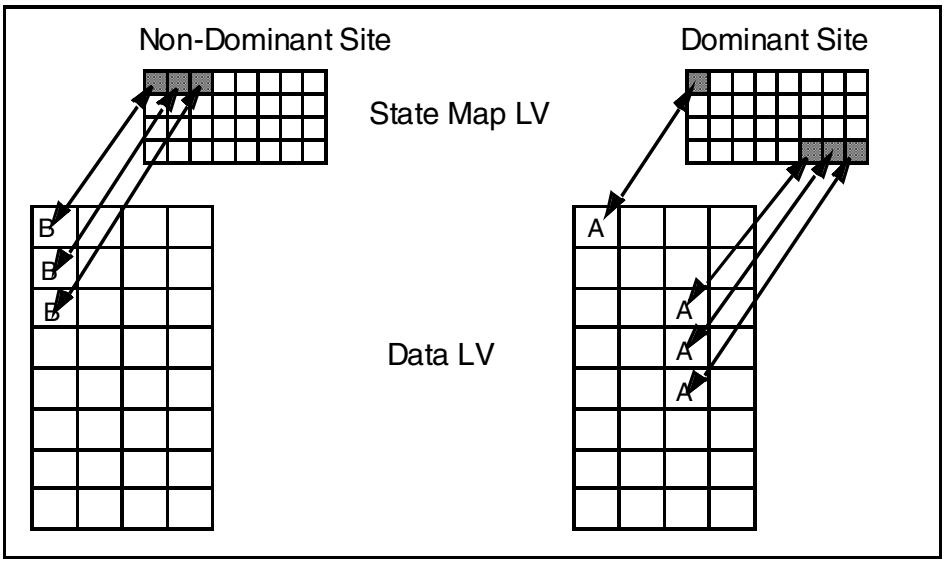

Figure 124. State map and data status during data divergence

To recover from this condition and re-create one version of the data, just rebooting one node and starting HACMP is not enough. Anytime there are staleness markings in the state map of a GeoMirror device as it is starting up, GeoMirror interprets the situation as data divergence. GeoMirror aborts the startup of the GeoMirrror device.

The reason for this action is that GeoMirror can detect the existence of data difference between the sites, but it is definitely not able to determine which site's data is the correct copy (*Master*). Only the administrator should make that decision. Therefore, the manual intervention of the administrator is forced at this point. This manual intervention process is called *unifying the state map.*

The process of unifying a state map is as follows:

1. To check for the existence of data divergence, view the state map for the GeoMirror device on the machines at both sites. If you have cells marked in an inconsistent state (other than 0x0 value) on both site nodes, you have data divergence.

   In most cases (if DBFS or an SGN is being used) where data divergence has occurred, the node at the non-dominant site has started the

application as part of a takeover before site isolation has been diagnosed. If this is the case, the node at the non-dominant site should have been halted by GeoManager at this point. If this is the case, you need to power-on this node and manually vary on volume group(s) that contain the data logical volumes and state map logical volumes.

2. Preview the result of unifying the state map.

   The state map unification process is a bit-wise copy of state map cell information from both site nodes. For example, let us say you have the data divergence situation illustrated in Figure 124 on page 273. After unification, the state map information would be changed as shown in Figure 125. This preview operation does not actually change the state map cell information, it simply creates a preview of what it would look like after the unification.

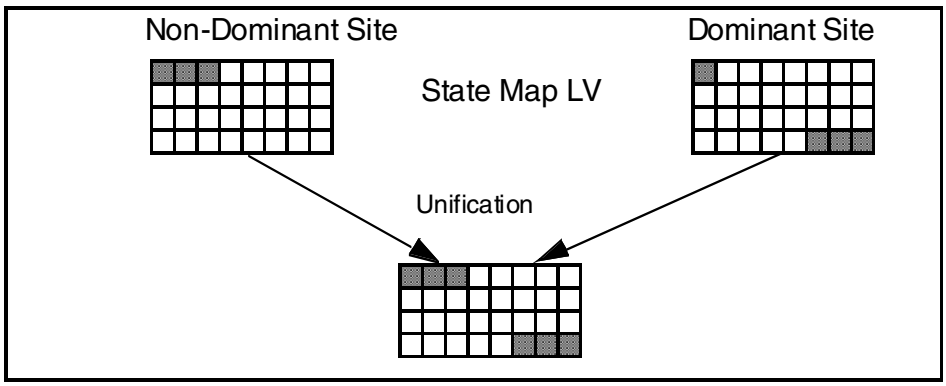

Figure 125. Unification process

To preview the result of unifying the state map, you can use smit or the command line:

```
# /usr/sbin/gmd/gmd_show_state -U -l test; echo $?
```

the output looks like this:

```
Point of View: Node black
--------------

Point of View GMD List:
-----------------------

Name: test
Status: AVAILABLE
```

```
State Map:

 Cell     Value
-------   -----
      0   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0xf 0x0 0x0 0x0 0x0 0x0 0x0 0x0
     16   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
     32   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0xf 0x0 0x0 0x0 0x0 0x0 0x0 0x0
     48   0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
*
1048576
2
```

Where *test* is the GeoMirror device name.

The `gmd_show_state` command returns with exit code 2 if one of the state maps is marked dirty. In the unification process, there is always at least one state map marked dirty; so, you always get exit code 2. Thus, the smit command panel shows *Fail* for the Command heading

3. Determine which node should be the master.

   This selection should be made carefully. If you make a mistake, it will cause data loss. For example, if you have the data divergence situation shown in Figure 124 on page 273, and you decide the master will be the dominant site node, all data in the regions connected with the hatched state map cells on the non-dominant site node (represented as B) will be overwritten. If you decide the master should be the non-dominant site node, all data in the regions connected with the hatched state map cells on the dominant site node (represented as A) will be overwritten. It is recommended that you take a backup of each node's data logical volume before proceeding with the unification.

4. Stop the GeoMirror device on the node that is determined to be the non-master state map in the previous step. Stopping the GeoMirror device is required for the `gmd_update_state` command to execute (see the next step).

5. Unify the state map.

   This step can be done on any node. When it is done, the non-master state map node should not have GeoMirror devices available. Hopefully, this operation will be done after a node reboot of a non-dominant site node that has halted and before starting HACMP. The gmd_update_state utility can be used when the GeoMirror device is in the *Defined* or *Stopped* state.

   After this operation, the state map cell information on the master node is changed to a *unified* state map, and the state map cell information on the

non-master node is *cleared*. You can unify the state map smit or command line:

```
# /usr/sbin/gmd/gmd_update_state -U -d black -l test; echo $?
Clearing test's state map on blue.
0
```

Where *black* is the master node name that determined in the previous step, and *test* is the GeoMirror device name.

8.5 Client recovery considerations

In site failure situations, the clients and routers that depend on geographic resources (applications or file systems) need to change their target from a failed site node to a surviving site node. What is the smoothest way to have clients reconnect to a surviving site with the least disruption possible? The following sections present some considerations for designing your client recovery plan.

8.5.1 Intersite client networks

Since the primary GeoNetwork must only be used for the GeoMirror function (no client traffic), if you want to restore client connectivity with geographical resources, you will need separate cross-site networks (we call them *Inter-site Client Networks*) for the clients to use. Without this type of network, clients cannot connect to geographic resources after a site failure (see Figure 126 on page 277).

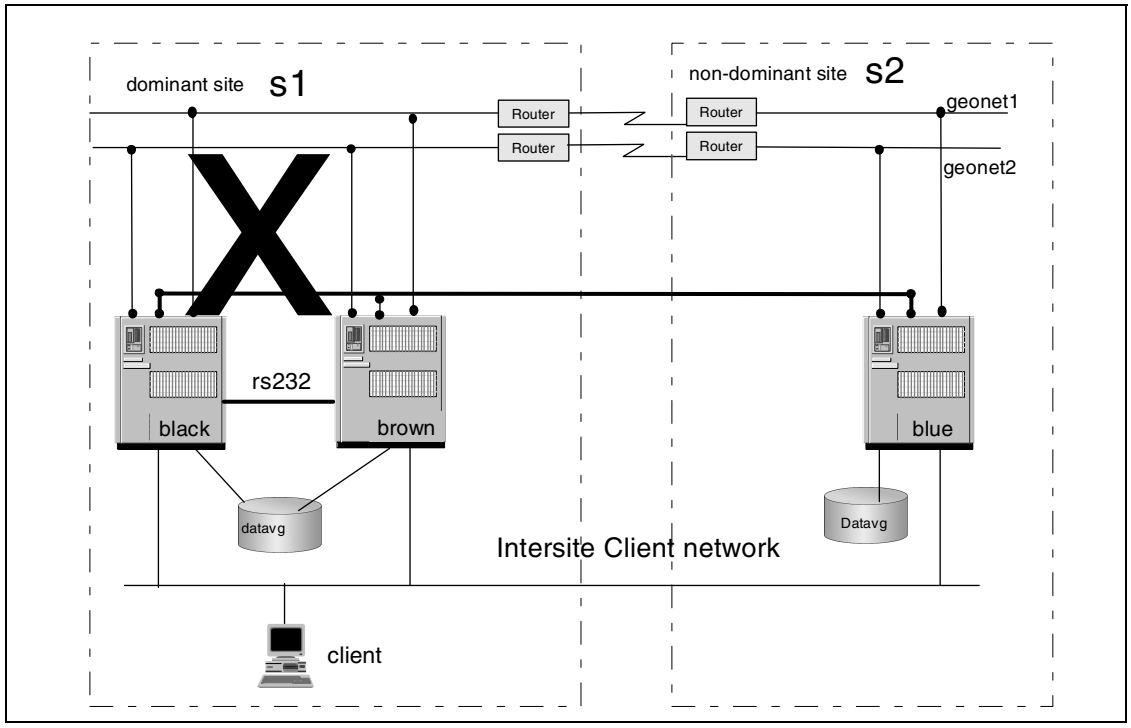

Figure 126. Inter-site client network in site failure situation

You do not need to define the inter-site client network as an HACMP network. However, if you define it as an HACMP network and there is no router in the network (sites are connected by a bridge), you can configure IPAT (IP address take over). Remember that IPAT is not supported on any GeoNetwork. If IPAT is possible, the recovery of client access after the site failure is much easier to implement than in the case without IPAT.

If you do not define an inter-site client network as an HACMP network, or if you have router(s) in the network (in most cases, there will be a router per site), you cannot configure IPAT. In this case, the recovery of client access after a site failure situation will be more complex. We will explore both of these options in the following sections.

8.5.1.1 IPAT-enabled on inter-site client networks

After a site failure, the geographical resources should be taken over by a non-dominant site node. If the client uses the inter-site client network to connect to geographical resources and IPAT is configured on this network, the clients simply wait for the takeover of geographical resources to complete

successfully. This resource takeover will take more time than in an ordinary HACMP cluster (it depends on the configuration, but is usually less than 30 minutes). You could extend the timeout duration and/or retry count of the user application on these clients (if possible). Otherwise, the user application will need to be reconnected manually.

8.5.1.2 Without IPAT on inter-site client networks

After a site failure, you need to switch the server IP address the client uses for application access. There are several ways to accomplish this including the ideas we will present in the following sections. We have not tested these ideas, but they should be viable options.

Manually switching

With this method, the client application user simply reconnects to a new server address (previously known to them) and continues using the application. This simple method is fine to use and maintain if the following conditions are fulfilled. A real site failure situation will seldom occur (or should seldom occur); so, this strategy might be sufficient. The following conditions make this idea more applicable:

- The user application using the geographical resource does not require continuous connection.
- The manual switching method is planned in advance and is known to the application user and/or client system administrator.
- There is an alert mechanism or system to let application users know of the site failure situation.

Using DNS application server

If your client is using DNS to resolve IP addresses and is using the HAGEO cluster node as its DNS server, you can prepare for a site failure situation with the following method. This method supplies transparent name resolution for the geographical application server name and IP address in a site failure situation. Even if you prepare for a site failure situation using these methods, you still need to detect the site failure situation and reconnect the client to the user application.

1. All clients use the HAGEO cluster node as their DNS server.

   All clients using the user application that needs geographical resources have to use an HAGEO cluster node as their DNS server.

   On the client side, the first DNS server should be the highest priority resource node on the dominant site, and the second DNS server should refer to the highest priority node on the non-dominant site.

If this client's operating system is UNIX, the /etc/resolv.conf file should be configured as shown in Figure 127 on page 282.

2. Modify /etc/named.data.

   All geographical application server names that are accessed by the user application and clients must be referred to by a DNS alias name (CNAME resource record) in the /etc/named.data file. To minimize the application server outage duration, the Time To Live (TTL) field must be set to a short time period (five to ten minutes is suitable).

3. Configure DNS application server.

   Configure a primary DNS server on a dominant site node and a secondary DNS server on a non-dominant site node. These DNS servers are defined as a geographical application server resource in HACMP, and the start/stop shell script would be like the following example:

   Before site failure

   1. On the dominant site primary DNS server node, do the following file copy manipulation:

      ```
      # cp /etc/named.boot.primary.dominant /etc/named.boot
      # cp /etc/named.data.primary.dominant /etc/named.data
      ```

      The primary DNS server on the dominant site uses the following configuration files:

      ```
      ; /etc/named.boot.primary.dominant
      ;
      ; use before site failure situation on dominant site.
      ;
      ; type          domain            source file or host
      ;
      domain          foo.com
      primary         foo.com           /etc/named.data
      primary         in-addr.arpa      /etc/named.rev
      ;
      ; end of /etc/named.boot.primary.dominant
      ;
      ; /etc/named.data.primary.dominant
      ;
      ; use before site failure situation on dominant site.
      ;
      ; OWNER                TTL             CLASS  TYPE    RDATA
      ;
      brown                 99999999        IN     A       X.X.X.X
      black                  99999999        IN     A       X.X.X.Y
      blue                 99999999        IN     A       Y.Y.Y.Y
      ```

```
geoserver               600             IN      CNAME   brown.foo.c
om.
;
; end of /etc/named.data.primary.dominant
;
```

2. On the non-dominant site secondary DNS server node, do the following file copy manipulation:

   ```
   # cp /etc/named.boot.secondary.non-dominant
   /etc/named.boot
   ```

   The secondary DNS server on the non-dominant site uses the following configuration file:

   ```
   ; /etc/named.boot.secondary.non-dominant
   ;
   ; use before site failure situation on dominant site.
   ;
   ; type          domain            source file or host
   ;
   domain          foo.com
   secondary       foo.com           X.X.X.X         /etc/nam
   ed.data.bak
   secondary       in-addr.arpa
        X.X.X.X         /etc/named.rev.bak
   ;
   ; end of /etc/named.boot.secondary.non-dominant
   ;
   ```

3. Start the primary DNS server on the dominant site node, and then start the secondary DNS server on the non-dominant site node.

After site failure:

1. Stop the primary DNS server on the dominant site node (if it is still alive) and the secondary DNS server on the non-dominant site node.
2. On the non-dominant site node, which will become the new primary DNS server node, do the following file copy manipulation:

   ```
   # cp /etc/named.boot.primary.non-dominant /etc/named.boot
   ```

   ```
   # cp /etc/named.data.primary.non-dominant /etc/named.data
   ```

   The new primary DNS server on the non-dominant site node uses the following configuration files:

   ```
   ; /etc/named.boot.primary.non-dominant
   ```

```
;
; use before site failure situation on dominant site.
;
; type          domain            source file or host
;
domain          foo.com
primary         foo.com           /etc/named.data
primary         in-addr.arpa      /etc/named.rev
;
; end of /etc/named.boot.primary.non-dominant
;
; /etc/named.data.primary.non-dominant
;
; use before site failure situation on dominant site.
;
; OWNER               TTL            CLASS  TYPE    RDATA
;
brown                99999999        IN     A       X.X.X
.X
black                 99999999        IN     A       X.X.
X.Y
blue                99999999        IN     A       Y.Y.Y.
Y
geoserver             600             IN     CNAME   blue
.foo.com.
;
; end of /etc/named.data.primary.non-dominant
;
```

3. Start the primary DNS server on the non-dominant site node.

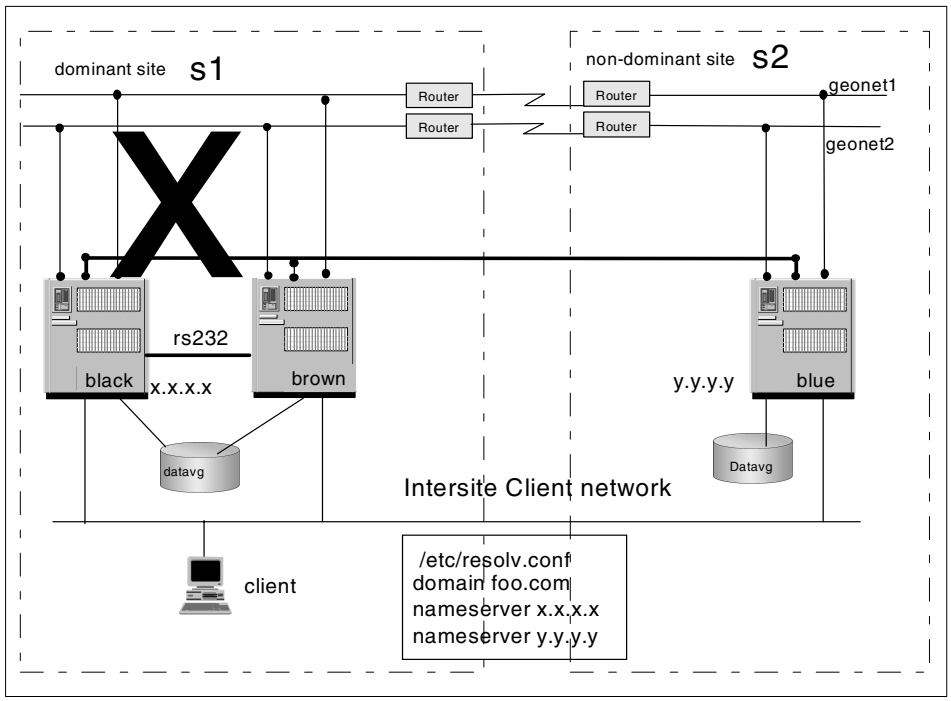

Figure 127. /etc/resolv.conf reference on inter-site client network

Applications that use clinfo API functions

If your application is running on AIX and configured with clinfo, your application can have better site failure detection methods. This method requires rewriting the application to use the clinfo API.

1. Configure clinfo on AIX client.
2. Rewrite the application using clinfo API functions.

Chapter 9. FAQs, hints, and tips

This chapter will present some frequently-asked questions (FAQs), hints, and tips about GeoRM and HAGEO. Since most components are almost the same for GeoRM and HAGEO, if nothing else is specified, the information is valid for both.

9.1 HACMP, GeoRM, and HAGEO

This section will discuss the relationship between HACMP, GeoRM, and HAGEO.

GeoRM and HAGEO
GeoRM can be regarded as simply HAGEO without the automatic failover and recovery capabilities. It provides mirroring of data between geographic remote sites. It provides less function than HAGEO.

Can GeoRM coexist with HACMP in the same system?
Yes, GeoRM can coexist with HACMP on the same system.

Can GeoRM coexist with HAGEO in the same system?
No, there is no integration support between these two products at this time.

Does HACMP 4.3 support HAGEO 2.1?
Yes, HAGEO 2.1 now supports HACMP 4.3 and AIX 4.3.2. You need to install HAGEO PTFs. Contact an IBM representative or technical support center to get these PTFs.

Do HACMP/ES or HACMP/ES CRM support HAGEO 2.1?
No, neither HACMP Enhanced Scalability nor Enhanced Scalability Concurrent Resource Manager support HAGEO 2.1. Only classic HACMP supports HAGEO 2.1.

How many nodes can we have in one cluster?
Up to now, we can have eight nodes in an HAGEO cluster, 32 nodes in one HACMP cluster, and up to eight servers in a GeoRM configuration (at least one target server).

Difference between HAGEO and GeoRM site
There are some differences: HAGEO can have two physical sites, and GeoRM can have two logical sites. Both sites include different physical locations and up to eight physical locations in one GeoRM configuration.

GeoRM and HAGEO: Which one is suitable for me?
It depends on your expectations. Both products provide High Availability solutions, but GeoRM only mirrors data to the remote site - there is no automatic failover once a disaster occurs. HAGEO provides not only data mirroring but also application automatic failover; so, the High Availibility solution GeoRM provided is lower than HAGEO. It needs more time and higher skills to recover from a disaster manually, but it is a cheaper solution than HAGEO, and, most importantly, GeoRM can also keep the data consistent.

9.2 GeoManager

This section will present some FAQs, hints, and tips related to GeoManager.

Hostname and machine name
In this release of HAGEO or GeoRM, you must use the hostname of a node as the machine name when you configure GeoManager and GeoMessage.

Permission denied when sync GeoManager definition
Make sure the IP labels of all GeoRM or HAGEO network interfaces are defined in the /.rhosts file on all machines at each site. The synchronization function uses rsh and, thus, requires these /.rhosts entries. Failing to do this will result in a permission denied error from the rshd daemon.

Configuration of GeoRM leaves temporary files in /tmp
In this version of GeoRM, configuration of GeoRM will leave some temporary files in /tmp, such as machine.add.11844. You can remove these yourself after you finish the definition.

Configuring the GeoRM site and machine separately
You will configure GeoRM ‘s site and machine via smit. These two menus contain the same information, but you have to define them separately.

9.3 GeoMessage

This section will present some FAQs, hints, and tips related to GeoMessage.

Does GeoRM or HAGEO require you to lay fiber links between sites?
GeoRM or HAGEO does not require fiber to be laid. However, if fiber can be laid, GeoRM or HAGEO can use a communications path supporting IP running over this fiber link. Also, GeoMessage can transmit data over this path to gain better performance.

Purpose of Geo_Primary and Geo_Secondary networks
In terms of HAGEO/GeoRM, there are HAGEO data networks and non-HAGEO data networks. The HAGEO data network carries HAGEO-mirrored data and heartbeat packets. The non-HAGEO data network only carries heartbeat packets. Both span the HAGEO sites.

There can be multiple HAGEO data networks. It is required that you also have a non-HAGEO data network. Since the non-HAGEO data network is not used for any HAGEO data mirroring, it does not matter what you define as a secondary network. The non-HAGEO data network is strictly a backup network to prevent site isolation.

You do not have to use the predefined Geo_Primary and Geo_Secondary network modules. You can use the NIM that is appropriate for your network type. The failure detection rate may have to be adjusted to a slower rate to account for the geographic dispersion.

Secondary Geo Network (SGN) represents a non-HAGEO data network spanning the sites to check for site isolation. The alternative to SGN is to use Dial Back Fail Safe (DBFS). This uses a telephone line and a pair of modems to connect the sites when all the HAGEO data networks are unavailable. Like SGN, it is used to distinguish between site isolation and site failure. You cannot have SGN and DBFS configured simultaneously.

Altering the default Geo_Primary and Geo_Secondary ODM stanzas
Depending on the average round trip time of your geographical network, you may need to change the heartbeat timeout values (hbrates expressed in milliseconds) to a more suitable value. You can use the following procedure to alter the hbrate attribute:

1. Save the original ODM class files:

```
# cd $ODMDIR
# cp HACMPnim HACMPnim.orig
```

2. Dump the ODM class file to a text file:

```
# odmget HACMPnim > HACMPnim.out
```

3. Edit the text file manually, and change the value of the hbrate attribute to the value in the Geo_Primary or Geo_Secondary stanzas.
4. Delete the current HACMPnim ODM class definitions:

```
# odmdelete -o HACMPnim
```

5. Add the modified text file contents to the HACMPnim ODM class file:

```
# odmadd HACMPnim.out
```

6. Check the new values for the modified attributes with:

```
# odmget -q name=Geo_Primary HACMPnim
# odmget -q name=Geo_Secondary HACMPnim
```

Importing HACMP for AIX network definitions to HAGEO
When you import the HACMP network definitions to HAGEO, the Geo_Secondary networks and the standby adapter definitions will not be imported. This is not a problem since the GeoMessage configuration should only include networks used for mirroring the data, and we know the Geo_Secondary network is used only for heartbeats. If it were imported, you would have to delete it since you only need to define the networks that are used by HAGEO for mirroring the data.

If you have standby adapters defined for a network (it does not matter whether it is a GEO or non-GEO network), they will not be imported to HAGEO. This is because these adapters will not be used by HAGEO until a failure occurs. At that time, HACMP will handle the service and standby adapters swapping transparently to HAGEO. HAGEO thinks it is still using the same adapter.

Network supported by GeoRM and HAGEO
GeoRM and HAGEO only support IP-based networks, and this network can carry UDP/IP.

IP address takeover across the GEO Networks
You cannot perform IP address takeover across the geographic network. IP address takeover is not supported over geographical networks.

When Geo_Primary Network goes down in HAGEO
GeoManager will use the secondary geographical network or DBFS to differentiate between site isolation and site failure when all primary geographical networks fail.

If heartbeat communication can be transmitted over the secondary network or if it is verified by DBFS that the remote site is still alive, the problem is site isolation. In that case, GeoManager checks which site is configured as the dominant site. This site continues functioning, and the other site is brought down gracefully.

If heartbeat communication cannot be transmitted over the secondary network or if it is verified by DBFS that the remote site is dead, GeoManager determines that a site failure has occurred, and the surviving site takes over.

In both cases, resources are distributed among the surviving machines at the viable site according to the configuration.

When GEO network goes down in GeoRM
When you configure GeoMessage to create a machine definition, there is a parameter called Promote Failure Timeout. If you specify a non-zero value, GeoRM will use this value when the communication failure occurs. If the network communication does not recover after this timeout value, local peer considers remote peer is down, marks the state map to dirty, and does not try again to send data to the remote site. When the communication recovers, data will be synchronized to the remote site.

If you leave this parameter as zero, it implies infinite timeout.

Distance limitations between geographically-dispersed sites
Neither GeoRM nor HAGEO have any real distance limitations other than those brought about by the latency of the communication link.

Is GeoMessage reliable?
Yes. GeoMessage uses RPC to deliver messages while RPC guarantees message delivery.

Load balancing on multiple GeoPrimary networks
GeoRM and HAGEO support multiple GeoPrimary networks for the purpose of increasing availability. The GeoMessage device driver periodically measures the round trip time on each of the networks defined to it and uses this information to determine which network to use to send KRPC packets across. But, only one network is used at a time regardless of how many GeoPrimary networks are configured to the cluster. GeoMessage does not load balance across these networks.

The output from the krpcstat utility does not show the correct distribution of messages over multiple networks.

Configuring geographical networks in HAGEO
When defining a network and its interfaces to GeoMessage in HAGEO, they should have been previously defined to HACMP. All the topology changes must be completed in HACMP and then altered in HAGEO. Also, notice that the changes made in HACMP are not automatically reflected to HAGEO. You must manually alter both configurations.

How to verify the KRPC kernel extension is already loaded?
You can check whether the KRPC kernel extension is loaded into the kernel with the following command:

```
# genkex | grep krpc | grep sbin
    1ace340        29c70        /usr/sbin/krpc/krpc
```

If the return from the command is null, it means that GeoMessage is not started. The `genkex` command is included in perfagent.tools fileset, which is part of the Performance Toolbox for AIX.

Another way is to use `/usr/sbin/krpc/cfgkrpc` to load it. You will get the message, *krpc already loaded*, if it has been loaded.

KRPC error message when already unload in HAGEO
In HAGEO, KRPC is unloaded as a post event to node_local_complete. This script may be called a number of times depending on the number of resource groups the node participates in. This results in error messages after the first successful unload of KRPC that ends with the message, *cfgkrpc: KRPC unload failed*.

You can ignore this message as long as the other log files show that the scripts have completed successfully.

Changing GeoMessage's configuration
You cannot add, rename, or remove a machine, network, or interface in the GeoMessage configuration while GeoMessage is active on any machines. To Change an attribute, you must stop the application, move all the GeoMirror devices to DEFINED, then stop GeoMessage on the affected machine before making any changes.

9.4 GeoMirror

This section will present some FAQs, hints, and tips related to GeoMirror.

The possible states of a state map cell
Data may exist in any one of the following states at a given time:

- Consistent (0x0)
- Data is the same at both sites.
- Stale (0xf)
- Data does not match at both sites, local write done, remote peer down.
- Inconsistent or Vulnerable (0x1 - 0x7)
- Data may not match between local and remote because writes are currently being processed. It is in an intermediate state.

- In sync mode, you should see that the cells are set to either 0x0 or 0xf. In MWC or async mode, you can see a variety of states among the cells reflecting intermediate stages of completion of local and remote writes.

State map consideration for local takeover

When you have two or more local nodes and are planning the local failover function, you are required to put the state map in the shared disks. When the local peer takeover occurs, it can access the state map of the failing node located in the shared disks to update its own state map and then clean the failing node's state map.

Sizing of GeoMirror Devices

The underlying logical volume for a GeoMirror Device must be the same size on both sites. It is possible to resize one side alone; there is no warning information unless you run the `geo_verify` utility. If the size is not the same, you may lose data.

When will a GMD fail to start?

There are certain conditions under which a GMD will fail to start. In general, these restrictions are in place to protect against data divergence. The following restrictions apply:

- A GMD will not start if the remote peer is available and there are staleness markings in the local state map. You must unify the state maps in this instance.
- A GMD will not start if more that one remote peer is active.
- A GMD will not start if a remote peer is in the process of starting.
- A GMD will not start if the `gmddown` command is not executed on the starting machine when the remote peer is up but the remote GMD is not available.
- A GMD will not start if the `gmddown` command is not executed and the remote machine is not reachable due to a geographic network error.
- A primary GMD will not start if an available remote peer is also primary.

Why does it take so long to start a GMD if the remote site is down?

When GMDs are started, an attempt is made to validate their configuration with the remote nodes before making the device available. This is done through the GeoMessage subsystem. The rpc daemon attempts to communicate with the remote host using RPCs. Starting GMDs takes a long time when the remote site is down because there is a wait for the RPC to time out. The default timeout is 25 seconds. The attempt to get several pieces of information using multiple RPCs (three) from each remote host could take a while if there are many GMDs and remote hosts.

If you are sure that the remote GeoMirror device is not available, you can use the `gmddown` command to inform the local node of this condition. The rpc daemon will not try to communicate with the remote host. The procedure is as follows:

```
# /etc/methods/gmddown -l gmdname remote-machine
```

or

```
# /etc/methods/gmddown -A remote-machine
```

The `gmddown` command informs the local node that one (option `-l`) or all (option `-A`) GMDs in the remote node are down. Then, you can issue the following command to start the GMD(s):

```
# /etc/methods/startgmd -l gmdname
```

or

```
# /etc/methods/startgmd -A
```

Later, when you start the GMDs on the remote node, you do not issue the `gmddown` command referring to this local node. The process of starting the GMDs on the remote node will be fast anyway because the GMDs on the local node are already started.

If the `gmddown` command is issued and the remote device is still available, the remote device is stopped to protect against data divergency. If the `gmddown` command was issued by mistake (the remote device was not down) and you use `smit` or the `lsdev -Cc geo_mirror` command to display the device status, the device status will be listed as available. The `lsdev -C` command displays device information from the Device Configuration Database of the ODM. The Device Configuration database is not updated when the device is taken off line internally.

If the command is issued by mistake, you must first stop and then restart the GMD device on the remote node to return it to service.

If one of the nodes that can take over the GMD is powered-off or is inaccessible through TCP/IP, the GMDs will take a long time to start, even if you issue the `gmddown` command as explained previously.

Synchronous, MWC, and asynchronous mode differences

A synchronous GeoMirror device first updates the logical volume on the remote system and then on the local system before returning control to the application that initiated the write request. When the write returns, data is

known to be secured on both sites. Due to network times and latency, this is the slowest performing mode as seen by the application.

Synchronous with Mirror Write Consistency (MWC) GeoMirror devices add extra logging using the state map to the synchronous mode, and the write requests are dispatched to both sites simultaneously. This makes MWC mode the best choice for mirrors that require reasonable performance and a guarantee of no data loss in the event of a disaster.

An asynchronous GeoMirror device simply performs the local and remote writes at the same time. As soon as the local write returns, the next write is handled. An asynchronous GeoMirror also uses state maps. Within the time discrepancy (milliseconds) between the local site and the remote site, there exists the possibility of losing data in the event of a disaster.

The valid range for minor device numbers to configure a GMD
The range is 0-1023. Each different GMD must have its own unique minor number. Remember that the minor numbers must be identical at both sites for a given GMD.

Can the asynchronous high water mark be changed on the fly?
It cannot. Changes to GMDs can only be done when the device is in a defined state.

High and low water mark for pending write I/Os per file
You can alter or set the values for the parameters above using `smit` and selecting **System Environments=>Change/Show Characteristics of the Operating System**.

If set, will these values conflict with GMD high/low water marks?
The parameters controls two different things. Remember the layering looks like this:

```
Application ---> Filesystems ---> GMD--->GeoMessage--->Network ---> Logical
Volume Manager ---> Disks
```

The application writes to the file system, which then, whenever it wants, sends the data down to the GMD. The amount that it can fall behind is controlled by the AIX Operating System High Water and Low Water parameters described in the question.

Applications also have the ability to make this behavior synchronous by opening the file with an option called `O_SYNC` or synchronizing after completion of a transaction.

Assume the file system High Water Mark allowed the application to buffer 20 KB of writes, and the GMD High Water Mark allows the remote site to fall 20 KB behind. The application does 20 KB of writes to an asynchronous GMD, which then returns to the file system. The application can now do another 20 KB of writes before the GMD knows about it; so, in the worst case, the remote site is now 40 KB behind the application. Therefore, you really need to add them together to determine the total amount of data you may lose in a system crash.

If, in this same example, this was an synchronous GMD, the file system could still buffer 20 KB of writes, and you could lose this 20 KB on a crash.

General rules for configuring GMD

- The GMD name must be the same in all local and remote peers.
- The device minor number of a GMD must be the same in both sites.
- You should specify a raw data LV when configuring the GMD.
- You should specify raw state map LVs when configuring the GMD.
- The LV name must be the same in all local and remote peers.
- The LV size must be the same in all peers (not PPsize, but actual size).
- The LVs must exist in shared VG if it is to be taken over locally within a site.
- One state map LV per GMD per node.
- The state map LV name must be unique within a site.

Reaching the HWM on an asynchronous GMD

This attribute is only used by async mode devices assigned the primary role.

The GMD will behave in MWC mode until all outstanding writes catch-up. The user will not see a change in the device mode anywhere. This change is completely internal to the driver.

Can we tune the HWM level to not switch between modes?

There is no way you can tune when you hit the GMD High Water Mark. You cannot tune when it is switching between asynchronous and MWC modes of operation. Users should have sufficient network bandwidth to handle their normal level of write traffic.

9.5 Geographic configurations

This section will present some FAQs, hints, and tips related to Geographic configurations.

Criteria used to choose your GeoRM or HAGEO configuration
A number of factors should be taken into account when planning a GeoRM or HAGEO cluster. These include applications, networks, and other issues.

What applications are critical and to what degree? All critical applications should be configured as HAGEO resources. Consider the trade-off between data integrity and performance. Synchronous and MWC GMDs will have higher data integrity than asynchronous GMDs at the cost of slower performance.

What is your required network bandwidth? You can estimate the network bandwidth according to the application's write requirements (see Section 3.4, "Planning networks" on page 45. A secondary HAGEO network or DBFS is strongly recommended to detect site isolation or failure and prevent data divergence.

Is one site more prone to disaster than the other? Does one site have more critical data than the other, or is it accessed by more clients? These are a few of the criteria to consider when determining which site should be dominant.

What is a standard GeoRM or HAGEO configuration?
There is no standard configuration. Application, network, and performance issues will influence your GMD and resource type configuration. It is expected that a four-node cluster with two nodes per site will be common. This will enable local failover within both sites (except when async GMDs are used).

Are two dispersed sites seen as one HACMP cluster or two?
The two sites are viewed as one HACMP cluster to both HACMP for AIX and HAGEO.

How many nodes and sites can I have in a cluster?
A maximum of eight nodes spread across two sites can be configured in one HAGEO cluster. The eight nodes can be distributed across the two sites in any combination. One GeoRM site can include different physical locations.

What happens to a client at site A when site A goes down?
The users are not automatically logged on to site B. They must reconnect to the surviving site in the HAGEO. For GeoRM, clients can be logged to site B only after the application has been brought up manually.

Steps for extending a logical volume

Before changing a GMD's underlying logical volume, you have to bring down the application, stop, and remove the GMD to defined status. Then, you can increase the size of the logical volume. The GMD has no way of knowing that the LV is extended; so, it will start returning errors if you read or write past the original end of the LV. It is, therefore, recommended that you create your GMD logical volumes with sufficient free space to avoid this impact on a regular basis, and remember that the size of LV for one GMD should be the same on both sites.

How to increase the size of a file system mounted over a GMD

The following procedure enables you to increase the size of a file system on which a GeoRM or HAGEO application resides. The application on which the file system resides is off-line during this procedure. However, other applications that do not use this file system can still be available. Here is the procedure:

1. On the local machine, stop the GeoRM or HAGEO application that resides on the file system you want to change.
2. On the local machines, unmount the file system on which the GeoRM or HAGEO application resides.
3. On both the local and remote site nodes, stop the GMD on which the file system resides, and put it in a defined state:

   ```
   # /etc/methods/stopgmd -l <filesystem_gmdname>
   # /etc/methods/ucfggmd -l <filesystem_gmdname>
   ```

The file system log GMD can remain available during this procedure enabling other file systems that use it to remain in service.

Complete steps 4 through 8 on the local machine first, then on the remote machine.

4. Change the file system entry in `/etc/filesystems` to point to the non-GMD file system logical volume (that is, change /dev/<hageo_fs_GMD> to /dev/<non_hageofslv>):

   ```
   # vi /etc/filesystems
   ```

5. Change the file system log logical volume info to point to the non-GMD file system log logical volume.

   a. If a non-GMD file system log logical volume does not exist, create one through `smit mklv`.

   b. Use `logform` to initialize the new logical volume as a jfs log:

      ```
      # logform /dev/loglvname
      ```

c. Change the file system log entry in /etc/filesystems to point to the non-GMD file system log logical volume using `chfs`. This will ensure that the lvcb log entry points to the correct log logical volume.

```
# chfs -a log="/dev/<non_gmdfsloglv>" <fsname>
```

6. Increase the size of the file system through `smit`:

```
# smit chfs
```

7. Change the file system log logical volume info to point to the GMD file system log logical volume:

```
# chfs -a log="/dev/<hageo_fs_log_GMD>" <fsname>
```

8. Change the file system and file system log entries in `/etc/filesystems` to point to the GMD file system logical volume and GMD file system log logical volume (that is, change /dev/<non_hageofslv> to /dev/<hageo_fs_GMD>):

```
# vi /etc/filesystems
```

9. After completing steps 4 through 8 on both sites, dirty the file system GMD on the local node.

```
# gmddirty -l <filesystem_gmdname>
```

10. Configure the file system GMD on the local node:

```
# cfggmd -l <filesystem_gmdname>
```

11. On the local node, run `gmddown` on the file system GMD for the remote node(s):

```
# gmddown -l <filesystem_gmdname> <remote_node>
```

12. Start the file system GMD on the local node:

```
# startgmd -l <filesystem_gmdname>
```

13. On the remote node, configure and start the file system GMD. This will cause the GMD data to synchronize from the local node:

```
# cfggmd -l <filesystem_gmdname>
# startgmd -l <filesystem_gmdname>
```

14. If the file system GMD is relatively small, you may wish to verify data integrity on both sites.

a. Save a copy of the data on the GMDs on the local node.

```
# dd if=/dev/filesystem_gmdname of=/tmp/fsgmd.local
```

b. Save a copy of the data on the GMDs on the remote node.

```
# dd if=/dev/filesystem_gmdname of=/tmp/fsgmd.remote
```

c. Verify that the data on the GMDs is identical.

```
# cmp /tmp/fsgmd.local /tmp/fsgmd.remote
```

9.6 Data consistency

This section will present some FAQs, hints, and tips related to data consistency.

How many GMDs are synchronized at one time?
The only way to simultaneously start multiple GMDs is to use the `startgmd -A` command. The `startgmd` command starts five GMDs at a time, and all five GMDs are synchronized at the same time.

Due to the integration issues with HACMP for AIX when controlled by HACMP, the GMDs are processed on a resource group level. When a volume group is varied on, it is scanned for any GMDs, which are then started. Otherwise, you could have GMDs configured that are not ready to be started because the underlying volume group was not varied on. The execution order of the scripts prevents parallelizing the startup of GMDs.

The -A option starts GMDs in parallel but needs to be used outside of the HAGEO and HACMP for AIX scripts. We recommend that you not use it. If you would like to use this command, it is better to specify the concurrency_level as 1.

Prestarting the GMDs in HAGEO
In an HAGEO Cascading Resource Group, the Owner Node has the applications started, the volume groups varied on, and the file systems mounted. In the event of a failure of the Owner Node, the next highest priority node varies on the volume groups, mounts the file systems, and starts the applications. When the Owner Node rejoins the cluster, the node that has taken over the resources of the Owner Node should give them back. The node that has taken over the resources stop the applications, unmounts the file systems, and varies off the volume groups.

If the resources fail over across the geography, it may take a considerable amount of time for a reintegration of the owner node due to the mirroring data synchronization of the GMDs. Also, during this period, the application is unavailable; it cannot be accepted normally.

Therefore, HAGEO provides a utility to solve this problem called `Geo_prestart_gmds`. Before starting HACMP in the joining node, run the following command first. The data synchronization begins while the application continues to operate making sure the output is appended to the /tmp/hacmp.out log:

```
# /usr/sbin/gmd/scripts/Geo_prestart_gmds | tee -a /tmp/hacmp.out
```

When the script completes (the time it takes is dependent on the amount of data), HACMP for AIX can be started.

If you would like the prestarting of GMDs to be automated, you must have a `/.rhosts` file on each machine in the geographic cluster, and you must edit the `/usr/sbin/gmd/scripts/Geo_remote_node_up` script. By default, this script will not run unless the `exit 0` command in its first line is removed or commented out. Once it is removed, the script will automate the prestarting of the GMDs at the joining node to minimize the application downtime during reintegration. This script makes calls that run `rsh` commands. This requires that the `/.rhosts` file on the joining node include the local node.

Data consistency in HAGEO file systems

If an HAGEO file system was created over a sync or a MWC GMD, you cannot mount it on more than one node at the same time. This is not supported and should never be done. The file system should not be mounted on the secondary site as long as the primary site is active. Attempting to read and write the file system from a node on the secondary site could crash that node and cause data corruption. By the way, both the file system and the log must have the same device mode.

Recover from data divergence

When you suspect data divergence, you first need to use the `gmd_show_state` command or `smit` to check the state map's status on both sites. If it really happens, you must decide which version of the state map is the master and unify the state map.

Before unifying the state map, it is better to pick the master state map(s) in which the other state maps from the nodes will be merged. These state maps will become the master state map view that will be used for synchronization of the GMDs.

The node of the master state map will become the Destination Machine Name on the Unify State Maps dialogue screen from `smit`.

State maps need to be examined to determine the state of all GMDs on all participating nodes at each site. Save a backup of the state map for each GMD on each node prior to unification by executing the following command:

```
# gmd_update_state -M mapfile
```

Use the file mapfile for the specified operation. This file is a machine-readable format data file.

Preview the unified state maps before unifying in HAGEO by entering `smit hageo` and selecting **HAGEO utilities=>GeoMirror Utilities=>Change State Map=>Preview Unified State Map** or by entering `smit georm` and selecting **GeoRM Utilities=>GeoMirror Utilities=>Change State Map=>Preview Unified State Map in GeoRM**.

After previewing the unified state map, use `smit` to unify the state maps. Enter `smit hageo`, and select **HAGEO Utilities=>GeoMirror Utilities=>Change State Map=>Unify State Maps**.

Also, in GeoRM, the path is **smit georm=>GeoRM Utilities=>GeoMirror Utilities=>Change State Map=>Unify State Maps**.

Remember to use the node name of the master state map in the Destination Machine Name field.

Upon successful unification of the state maps, the state maps from the other nodes will be cleared and displayed as consistent. The master state maps contain the view of what data will be synchronized.

9.7 Geographic utilities

This section will present some FAQs, hints, and tips related to geographic utilities.

Modems and configuring DBFS
Correctly-configured DBFS will have each node with its own DBFS (only) modem to confirm that all nodes have failed before taking further action. The modem is dialed from remote node to local node. Actually, this facility is very important to HAGEO to identify site isolation and site failure, and it needs to be checked often.

Can you specify more than one DBFS number to call?
When configuring the HAGEO site, set Backup Communications to DBFS. Enter the correct modem number for each node when configuring the HAGEO nodes. The scripts will get these numbers and dial them for the remote nodes. You can configure one telephone number per node in the cluster.

Under what circumstances will DBFS be executed?
The DBFS feature is not used until the non-dominant site notices that all the dominant site nodes have gone. The dominant site should never perform DBFS calls, and, normally, DBFS will call three times for each dominant node before identifying the dominant node as actually being down.

Security issues when using DBFS
There are a few security issues with DBFS.

Someone could dial in and hold the line busy so that a dominant site could be marked down and the non-dominant site would take over, potentially causing data divergence. To safeguard against this, HAGEO attempts to call each node at the dominant site three times to find one with the cluster running (this should be sufficient to get the correct information). The modem should not be used for any purpose other than DBFS. It should be dedicated to DBFS use.

The dbfs.env file must be modified for DBFS to work correctly. This file contains the login ID and password to the called system. This file should only be accessible to the root user.

9.8 Performance

This section will present FAQs, hints, and tips related to the performance of a GeoRM or HAGEO cluster.

Factors affecting the cluster's performance
There are many factors that will influence the cluster performance. The leading factors affecting a cluster's performance are the amount of data written by the application, the read and write frequency, the bandwidth of the networks, network and disk latency, the amount of CPU available, and the amount of available memory. We can buy more CPU and memory to avoid bottlenecks, and we can use the right network to meet the request. The most important thing to remember is that there is actually very little we can do to tune GeoRM or HAGEO once it is installed; so, what we can do for tuning good performance should mostly be done in the design stage.

Advantages of the 64 K sync concurrency rate
The advantage is that you can simultaneously issue two synchronous requests (two regions equals 64 K). Increasing the rate also puts an additional load on the network and the other site for the duration of the synchronizing process.

Overhead of MWC and synchronous mode
This depends on a number of things. With MWC, you have one extra write to the disk (the initial state map write) to wait for. However, the remote and local writes are dispatched at the same time (as opposed to sequentially in synchronous mode); so, the response time could possibly be faster with MWC than without it.

Why is MWC mode recommended in most cases?
MWC allows the local and remote writes to be overlapped. Hence MWC mode in the vast majority of cases will perform better than synchronous mode mirrors. Only use asynchronous mirror mode if you really can lose data.

Where should we put the state map for better performance?
Do not co-locate the state map for the GMD in a physical volume with the related data logical volume. In an ideal world, each state map should be placed on its own physical volume. This is certainly true for MWC and asynchronous mirror mode, because the seek and write times of the state map has been found to be the bottleneck on GMD throughput.

How fast can a failover take effect?
HACMP for AIX will detect the failure in less than 5 minutes (if using the default NIM values). The complete failover time depends on how many GMDs and applications need to be started, and so on. Also factor-in the amount of time the application itself needs to restart.

Performance differences of asynchronous and synchronous
Using asynchronous mode will improve response time by eliminating the round trip time on the network as well as the time to do the write on the remote site. All other things being equal, if data integrity is not the primary concern, use an asynchronous mirror mode. If data integrity is a must but performance is an issue, an MWC mirror mode should be used in preference to an asynchronous and synchronous mode. Remember that MWC is still one kind of synchronous mode. Beware of the increased local I/O workload.

9.9 Applications

This section will present FAQs, hints, and tips related to applications.

Do I need to modify applications running on GeoRM or HAGEO?
In fact, geographic mirroring is transparent to the applications since the GeoMirror devices provide a pseudo device that has the same interface as the raw disk driver provided by the LVM. Therefore, no changes to the application are required.

Read-intensive and write-intensive applications not performing well
Application environments are very important to the way that GeoRM or HAGEO perform. GeoRM or HAGEO may not perform well in a very read-intensive and write-intensive application environment. Because GeoRM or HAGEO will transfer writes across the network and wait for acknowledgement, so a write-intensive application is unlikely to perform well in this environment. That does not mean that a read-intensive environment is

better because writes will block reads. Consequently, in a read-intensive environment, a synchronous write to a geomirrored logical volume will cause subsequent read requests to be blocked pending its return. A balanced workload application is a good environment for GeoRM or HAGEO.

What kind of databases are supported?
Any kind of database is supported in GeoRM or HAGEO.

9.10 Miscellaneous

This section will present miscellaneous FAQs, hints, and tips.

Are there issues with fuser on a mirrored file system in HAGEO?
It should work the same way it usually does. A file system can only be mounted on one node at a time.

What happens if someone kills the rpc.geod daemon?
The rpc.geod daemon will be restarted automatically.

How much memory is required for HAGEO?
For virtual memory, about 0.5 MB of kernel virtual memory per GMD are required for the state map plus memory for buffers (in asynchronous mode, you can buffer up to the High Water Mark for the GMD).

For pinned memory, GeoMessage allocates from the kernel heap only. GMD allocates from the pinned heap for configuration information and synchronizing. Gbufs are allocated from the kernel heap and then pinned when needed. The amount will vary depending on site configuration and usage.

Does HAGEO complement or replace HACMP for AIX?
You must use HACMP for AIX with HAGEO in order to have both data mirroring and automated failure detection and recovery. Therefore, HAGEO complements HACMP.

Can individual components of HAGEO run by themselves?
They cannot.

Disk space required for implementing remote data mirroring
You need enough disk space at the remote site to hold all of the data you want to mirror. Moreover, you need to create state map LVs for each GMD per node, each of which will take up one physical partition.

Do GeoRM or HAGEO support ATM?
Yes, if you use IP over the ATM network.

Is the /.rhosts file required on systems running GeoRM or HAGEO?
The /.rhosts file is required when performing the following:

1. When we do the following through `smit` or the command line:
   - Verify the GeoRM or HAGEO Configuration.
   - Sync the Site/Machine Definitions to all Machines.
   - Distribute the GeoMessage Definitions to Remote Machines.
2. When prestarting of GMDs is to be automated.

Also, the names of all the machines in the cluster must be resolved into an IP address (through /etc/hosts or DNS) associated to one of the geographical network interface for these operations to work properly.

The /.rhosts file can be removed after completing either of these actions for the security concern.

Language settings for HAGEO
The language must be set to en_US. The event scripts do not work correctly without the language environment set this way. Make sure the filesets hageo.msg.en_US.message and hageo.msg.en_US.mirror are installed; otherwise, some functions will not behave properly.

Clinfo in trap mode
For some of the more complex failures that HAGEO handles, clinfo needs to be set in trap mode. To achieve this, you need to run the following command in both machines:

```
# chssys -a "-a" -s clinfo
```

This command only needs to be executed once.

config_too_long message in HACMP when starting GMD
The HACMP Cluster Manager will process the config_too_long event if the process of starting several GMDs exceeds six minutes. You may see the message displayed, but you can ignore it. The Cluster Manager will continue the start process.

Inactive takeover consideration in HAGEO
The scenario is as follows:

Node X and node Y are at site 1, and node Z is at site 2. There is an MWC GMD set up with local failover and remote failover and cascading resources in the order node X, node Y, node Z. A shared volume group exists between nodes X and Y. Inactive takeover was not activated in the configuration of the resource groups.

Site 1 becomes unavailable. Node Z takes over the resources.

Now, the machines at site 1 are available again. If you start HACMP in node X, node Z will release the resources, and node X will vary on the shared volume group and start the applications.

But, suppose that, instead of reintegrating node X first, you decided to start HACMP in node Y first. Node Z would release the resources because site 1 has priority over them, but the shared volume group is handled by a resource group that is cascading between nodes X and Y. Since inactive takeover has not been activated for this resource group, when node Y joins the cluster, it does not vary on the volume groups. Consequently, the GMDs are not started, and the applications and/or file systems that were released by node Z are not taken over.

If it is necessary to reintegrate node Y first, you can vary on the volume groups, start the GMDs, mount the file systems, and start the applications manually. You may also change the local resource group configuration to activate inactive takeover before starting HACMP in node Y. Remember that, before reintegrating node X in the cluster, you will have to synchronize the new resource configuration to this node. In this case, you can use `Geo_prestart_gmds` scripts.

Appendix A. Scripts for the migration to HAGEO

This appendix will show the contents of the scripts used to help the migration of an existing HACMP environment to HAGEO. There are also some scripts to reestablish the original HACMP environment. The use of these scripts was explained in detail in Chapter 7, “Migrating an existing application to HAGEO” on page 215. These scripts are also contained on the diskette included with this redbook.

A.1 Generate_LV_list.ksh

The following script generates an intermediate file with new names for the logical volumes according to the naming convention suggested in Section 7.2, “Naming convention” on page 217. This file also contains the names for the local and remote statemap logical volumes.

```
#!/bin/ksh -x
#
#
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Generate_LV_list.ksh -- generates a file with a list of  LV names.
#########################################################################

#########################################################################
# Step.0
# defines misc environment values.
#########################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH
```

```
#########################################################################
# Step.1
# defines miscellaneous functions.
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exits with return code 1.

function long_usage
{
print -u2 '    -l: local_peer1,local_peer2'
print -u2 '        white spaces are not allowed in this argument'
print -u2 '    -r: remote_peer1,remote_peer2'
print -u2 '        white spaces are not allowed in this argument'
print -u2 '    -i: input file'
print -u2 '        this file is the output of lsvg -l <vg_name>.'
print -u2 '    -o: output file'
print -u2 '        this file will be overwritten.'
exit 1;
}

# function:     generate_new_data_lvname
# arguments:    original LV name, GMD minor number
# return value: new name for the data LV

function generate_new_data_lvname
{
    if [ $# -ne 2 ]; then
        return;
    fi

    ORIGINAL_DATA_LV_NAME=$1;
    GMD_MINOR_NUMBER=$2;

    NEW_DATA_LV_NAME="${ORIGINAL_DATA_LV_NAME}""${GMD_MINOR_NUMBER}""lv";

    print ${NEW_DATA_LV_NAME};
}

# function:     generate_part_of_new_sm_lvname
# arguments:    original LV name, GMD minor number
# return value: part of the new name for the statemap LV
# Note:         final new name for the statemap LV will have
#               the site and node numbers added to the return
#               value of this function
```

```
function generate_part_of_new_sm_lvname
{
    if [ $# -ne 2 ]; then
        return;
    fi

    ORIGINAL_DATA_LV_NAME=$1;
    GMD_MINOR_NUMBER=$2;


PART_OF_NEW_SM_LV_NAME="${ORIGINAL_DATA_LV_NAME}""${GMD_MINOR_NUMBER}""sm_
";
    print ${PART_OF_NEW_SM_LV_NAME};
}


################################################################################
# Step.2
# handles command line options and checks for the existence of input file.
################################################################################

SHORT_USAGE="usage: $0 -l local_peer_specify -r remote_peer_specify -i
input_file -o output_file"

if [ $# -eq 0 ]; then
    print -u2 ${SHORT_USAGE}
    exit 1;
fi

while getopts :l:r:i:o: arguments
do
    case $arguments in
        :)
        print -u2 "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
        l)
        LOCAL_PEER_ARG=${OPTARG};
        ;;
        r)
        REMOTE_PEER_ARG=${OPTARG};
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        o)
```

```
        OUTPUTFILE=${OPTARG};
        ;;
        \?)
        print -u2 "${OPTARG} is not a valid switch.";
        print -u2 "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    print -u2 "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

#
# parses LOCAL_PEER_ARG.
#
LOCAL_PEER_COUNT=`echo ${LOCAL_PEER_ARG} |\
                awk -F',' '{print $1 " " $2}' |\
                wc -w | sed 's/ //g'`

LOCAL_PEER_1=`echo ${LOCAL_PEER_ARG} | awk -F',' '{print $1}'`
LOCAL_PEER_2=`echo ${LOCAL_PEER_ARG} | awk -F',' '{print $2}'`

if (( ${LOCAL_PEER_COUNT} < 1 )) || (( ${LOCAL_PEER_COUNT} > 2 ))
then
    print -u2 "Error: you must specify 1 or 2 local peers."
    exit 1;
fi

#
# parses REMOTE_PEER_ARG.
#
REMOTE_PEER_COUNT=`echo ${REMOTE_PEER_ARG} |\
                awk -F',' '{print $1 " " $2}' |\
                wc -w | sed 's/ //g'`

REMOTE_PEER_1=`echo ${REMOTE_PEER_ARG} | awk -F',' '{print $1}'`
REMOTE_PEER_2=`echo ${REMOTE_PEER_ARG} | awk -F',' '{print $2}'`

if (( ${REMOTE_PEER_COUNT} < 1 )) || (( ${REMOTE_PEER_COUNT} > 2 ))
then
    print -u2 "Error: you must specify 1 or 2 remote peers."
    exit 1;
fi
```

```
#########################################################################
# Step.3
# Checks whether exists enough free PPs on the given VG.
#########################################################################

# gets VG name.
VG_NAME=`head -1 ${INPUTFILE} | cut -d: -f 1`

# gets Free PPs of the VG.
FREE_PP_NUM=`lsvg ${VG_NAME} | grep 'FREE PPs' | awk '{print $6}'`

# counts LVs.
TOTAL_LV_NUM=`tail +3 ${INPUTFILE} | wc -l`

# calculates the number of statemap LVs.
(( TOTAL_SM_LV_NUM = ${TOTAL_LV_NUM} * ${LOCAL_PEER_COUNT} ))

# checks whether exists enough free PP on the VG.
if (( ${TOTAL_SM_LV_NUM} >=  ${FREE_PP_NUM} ))
then
    print -u2 "Error: You need ${TOTAL_SM_LV_NUM} PPs for the statemap LVs
on the VG: ${VG_NAME}."
    exit 1;
fi

#########################################################################
# Step.4
# Generates output file.
#########################################################################

# skips the first two lines from the input file and creates a temporary
file.
tail +3 ${INPUTFILE} > /tmp/$$."Generate_LV_list".$$

# print header information to output file.
> ${OUTPUTFILE}
echo "# List of LV names generated with Generate_LV_list.ksh" >>
${OUTPUTFILE}
echo "# on node: " `get_local_nodename` >> ${OUTPUTFILE}
echo "# date: " `date` >> ${OUTPUTFILE}
echo "#" >> ${OUTPUTFILE}
echo ${VG_NAME} >> ${OUTPUTFILE}
echo "#" >> ${OUTPUTFILE}
echo "#Original         Original  PP   GMD  New                    New" >>
${OUTPUTFILE}
echo "#LV_name          LV_type   size min# LV_name                LV_type"
>> ${OUTPUTFILE}
```

```
echo "# you should not change above this line." >> ${OUTPUTFILE}

# GMD device minor number
GMD_MINOR_NUM=-1

# sets print filed
typeset -L10 L10_LVTYPE
typeset -L18 L18_LVNAME
typeset -L5  L5_PP_NUM
typeset -L5  L5_GMD_MINOR_NUM
typeset -L10 L10_NEW_LVTYPE
typeset -L24 L24_NEW_LVNAME

while read LVNAME LVTYPE LP_NUM PP_NUM PV_NUM LV_STATE MOUNT_POINT
do
    # checks LVTYPE
    case $LVTYPE in
    "boot"|"paging"|"dump")
        continue
        ;;
    "jfslog")
        ((GMD_MINOR_NUM = ${GMD_MINOR_NUM} + 1))
        NEW_LVTYPE="gmdlog"
        ;;
    *)
        ((GMD_MINOR_NUM = ${GMD_MINOR_NUM} + 1))
        NEW_LVTYPE="gmd"
        ;;
    esac

    # generates new LV name.
    NEW_LVNAME=$(generate_new_data_lvname $LVNAME $GMD_MINOR_NUM)

    # generates part of the name for the statemap LV.
    NEW_SMLVNAME_BASE=$(generate_part_of_new_sm_lvname $LVNAME
$GMD_MINOR_NUM)

    # checks the string length of ${NEW_LVNAME}.
    if (( ${#NEW_LVNAME} >= 16))
    then
        print "# !!! this LV name is too long.!!!"
    fi

    L10_LVTYPE=${LVTYPE}
    L18_LVNAME=${LVNAME}
    L5_PP_NUM=${PP_NUM}
    L5_GMD_MINOR_NUM=${GMD_MINOR_NUM}
```

```
    L10_NEW_LVTYPE=${NEW_LVTYPE}
    L24_NEW_LVNAME=${NEW_LVNAME}

    # prints lines to ${OUTPUTFILE}.
    print
"${L18_LVNAME}${L10_LVTYPE}${L5_PP_NUM}${L5_GMD_MINOR_NUM}${L24_NEW_LVNAME
}${L10_NEW_LVTYPE}"

    #
    # local statemap LVs.
    #
    if (( ${LOCAL_PEER_COUNT} >= 1 ))
    then
        counter=1
        while ((${counter} <= ${LOCAL_PEER_COUNT}))
        do
            case ${counter} in
            2)
                LVNAME="=LOCAL"
                LVTYPE="=""${LOCAL_PEER_2}"
                PP_NUM=1
                NEW_LVTYPE="sm"
                NEW_LVNAME="${NEW_SMLVNAME_BASE}""1""${counter}"
                ;;
            1)
                LVNAME="=LOCAL"
                LVTYPE="=""${LOCAL_PEER_1}"
                PP_NUM=1
                NEW_LVTYPE="sm"
                NEW_LVNAME="${NEW_SMLVNAME_BASE}""1""${counter}"
                ;;
            *)
                print -u2 "unknown local peer node count."
            esac

            # checks the string length of ${NEW_LVNAME}.
            if (( ${#NEW_LVNAME} >= 16))
            then
                print "# !!! this LV name is too long.!!!"
            fi

            L10_LVTYPE=${LVTYPE}
            L18_LVNAME=${LVNAME}
            L5_GMD_MINOR_NUM=${GMD_MINOR_NUM}
            L5_PP_NUM=${PP_NUM}
            L10_NEW_LVTYPE=${NEW_LVTYPE}
            L24_NEW_LVNAME=${NEW_LVNAME}
```

```
            # prints lines to ${OUTPUTFILE}.
            print
"${L18_LVNAME}${L10_LVTYPE}${L5_PP_NUM}${L5_GMD_MINOR_NUM}${L24_NEW_LVNAME
}${L10_NEW_LVTYPE}"

            ((counter = ${counter} + 1))
        done
    fi

    #
    # statemap LVs for the remote peers.
    #
    if (( ${REMOTE_PEER_COUNT} >= 1 ))
    then
        counter=1
        while ((${counter} <= ${REMOTE_PEER_COUNT}))
        do
            case ${counter} in
            2)
                LVNAME="=REMOTE"
                LVTYPE="=""${REMOTE_PEER_2}"
                PP_NUM=1
                NEW_LVTYPE="sm"
                NEW_LVNAME="${NEW_SMLVNAME_BASE}""2""${counter}"
                ;;
            1)
                LVNAME="=REMOTE"
                LVTYPE="=""${REMOTE_PEER_1}"
                PP_NUM=1
                NEW_LVTYPE="sm"
                NEW_LVNAME="${NEW_SMLVNAME_BASE}""2""${counter}"
                ;;
            *)
                print -u2 "unknown remote peer node count."
            esac

            # checks the string length of ${NEW_LVNAME}.
            if (( ${#NEW_LVNAME} >= 16))
            then
                print "# !!! this LV name is too long.!!!"
            fi

            L10_LVTYPE=${LVTYPE}
            L18_LVNAME=${LVNAME}
            L5_GMD_MINOR_NUM=${GMD_MINOR_NUM}
            L5_PP_NUM=${PP_NUM}
```

```
            L10_NEW_LVTYPE=${NEW_LVTYPE}
            L24_NEW_LVNAME=${NEW_LVNAME}

            # prints lines to ${OUTPUTFILE}.
            print
"${L18_LVNAME}${L10_LVTYPE}${L5_PP_NUM}${L5_GMD_MINOR_NUM}${L24_NEW_LVNAME
}${L10_NEW_LVTYPE}"

            ((counter = ${counter} + 1))
        done
    fi

done < /tmp/$$."Generate_LV_list".$$ >> ${OUTPUTFILE}

# prints footer

print "#" >> ${OUTPUTFILE}
print "# end of file." >> ${OUTPUTFILE}
print "#" >> ${OUTPUTFILE}

# removes temporary file.

/usr/bin/rm /tmp/$$."Generate_LV_list".$$

exit 0;

#
# end of Generate_LV_list.ksh
#
```

A.2 Create_local_statemap_LV.ksh

The following script is used to create the local state map logical volumes according to the names described in the input file.

```
#!/bin/ksh -x
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
```

```
# Create_local_statemap.ksh -- creates statemap LVs according to input
file.
#########################################################################

#########################################################################
# Step.0
# define misc environment values.
#########################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH

#########################################################################
# Step.1
# defines miscellaneous functions.
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
echo '    -i: input file'
echo '        this input file must be generated with Generate_LV_list.ksh'
echo '        and should be revised manually.'
echo '    -D: debug mode only'
echo '        no action will be taken.'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and checks for the existence of the input
file.
#########################################################################

SHORT_USAGE="usage: $0 [-D] -i input_file"
DEBUG_FLAG="OFF"
```

```
if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :Di: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
        D)
        DEBUG_FLAG="ON";
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

#########################################################################
# Step.3
# sets misc variables.
#########################################################################

# gets VG name.
VG_NAME=`egrep -v '^(#|=)' ${INPUTFILE} | head -1`

# sets path for the mklv command.
if [ ${DEBUG_FLAG} = "ON" ]; then
    MKLV='echo /usr/sbin/mklv';
else
    MKLV='/usr/sbin/mklv';
fi

#########################################################################
```

```
# Step.4
# creates statemap logical volumes.
#########################################################################

# skips unnecessary lines from the input file and stores it in a temp file.
grep '^=LOCAL' ${INPUTFILE} > /tmp/$$."Create_local_statemap_LV".$$

while read LVNAME LVTYPE PP_NUM GMD_MINOR_NUM NEW_LVNAME NEW_LVTYPE
do
    ${MKLV} -t ${NEW_LVTYPE} -y ${NEW_LVNAME} ${VG_NAME} ${PP_NUM}
done < /tmp/$$."Create_local_statemap_LV".$$

# removes temporary file.
/usr/bin/rm /tmp/$$."Create_local_statemap_LV".$$

exit 0;

#
# end of Create_local_statemap_LV.ksh
#
```

A.3 Change_local_LV_name.ksh

The following script is used to rename the original logical volume names in the local node with the names described in the input file.

```
#!/bin/ksh -x
##
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Change_local_LV_name.ksh -- changes LV names according to input file.
#########################################################################

#########################################################################
# Step.0
# defines misc environment values.
#########################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
```

```
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH

#########################################################################
# Step.1
# defines miscellaneous functions.
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
echo '    -i: input file'
echo '        this input file must be generated with Generate_LV_list.ksh'
echo '        and should be revised manually.'
echo '    -D: debug mode only'
echo '        no action will be taken.'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and checks for the existence of input file.
#########################################################################

SHORT_USAGE="usage: $0 [-D] -i input_file"
DEBUG_FLAG="OFF"

if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :Di: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
```

```
        D)
        DEBUG_FLAG="ON";
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

#########################################################################
# Step.3
# sets misc variables.
#########################################################################

# gets VG name.
VG_NAME=`egrep -v '^(#|=)' ${INPUTFILE} | head -1`

# sets the path for the chlv command.
if [ ${DEBUG_FLAG} = "ON" ]; then
    CHLV='echo /usr/sbin/chlv';
else
    CHLV='/usr/sbin/chlv';
fi

#########################################################################
# Step.4
# creates statemap logical volumes.
#########################################################################

# skips unnecessary lines from the input file and stores it in a temp file.
tail +10 ${INPUTFILE} | egrep -v '^(=|#)' >
/tmp/$$."Change_local_LV_name".$$

while read LVNAME LVTYPE PP_NUM GMD_MINOR_NUM NEW_LVNAME NEW_LVTYPE
do
    case ${NEW_LVTYPE} in
    gmdlog|gmd)
```

```
        ${CHLV} -n ${NEW_LVNAME} ${LVNAME}
        ${CHLV} -t ${NEW_LVTYPE} ${NEW_LVNAME}
        ;;
    *)
        continue
        ;;
    esac
done < /tmp/$$."Change_local_LV_name".$$

# removes temporary file.
/usr/bin/rm /tmp/$$."Change_local_LV_name".$$

exit 0;

#
# end of Change_local_LV_name.ksh
#
```

A.4 Generate_remote_statemap_LV_script.ksh

The following script is used to generate a script to create the state map logical volumes on the remote nodes.

```
#!/bin/ksh -x
#
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Generate_remote_statemap_LV_script.ksh -- Generates a shell script
#                      to create the statemap LVs in the remote nodes
#########################################################################

#########################################################################
# Step.0
# defines misc environment values.
#########################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
```

```
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH

#########################################################################
# Step.1
# define miscellaneous functions.
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
echo '    -i: input file'
echo '        this input file must be generated with Generate_LV_list.ksh'
echo '        and should be revised manually.'
echo '    -o: output file'
echo '        this file will be overwritten.'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and checks for the existence of input file.
#########################################################################

SHORT_USAGE="usage: $0 [-D] -i input_file"
DEBUG_FLAG="OFF"

if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :Di:o: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
```

```
        D)
        DEBUG_FLAG="ON";
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        o)
        OUTPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

#########################################################################
# Step.3
# sets misc variables.
#########################################################################

# gets VG name.
VG_NAME=`egrep -v '^(#|=)' ${INPUTFILE} | head -1`

# sets the path for the mklv command.
if [ ${DEBUG_FLAG} = "ON" ]; then
    MKLV='echo /usr/sbin/mklv';
else
    MKLV='echo /usr/sbin/mklv';
fi

#########################################################################
# Step.4
# creates statemap logical volumes.
#########################################################################

# skips unnecessary lines from the input file and stores it in temp file.
grep '^=REMOTE' ${INPUTFILE} >
/tmp/$$."Generate_remote_statemap_LV_script".$$

> $OUTPUTFILE
```

```
while read LVNAME LVTYPE PP_NUM GMD_MINOR_NUM NEW_LVNAME NEW_LVTYPE
do
    ${MKLV} -t ${NEW_LVTYPE} -y ${NEW_LVNAME} ${VG_NAME} ${PP_NUM}
done < /tmp/$$."Generate_remote_statemap_LV_script".$$ >> $OUTPUTFILE

# removes temporary file.
/usr/bin/rm /tmp/$$."Generate_remote_statemap_LV_script".$$

exit 0;

#
# end of Generate_remote_statemap_LV_script.ksh
#
```

A.5 Generate_remote_LV_script.ksh

The following script is used to generate a script to create the data logical volumes on the remote node.

```
#!/bin/ksh -x
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Generate_remote_LV_script.ksh -- generates a shell script to create
#                                  the data LVs in the remote nodes.
#########################################################################

#########################################################################
# Step.0
# defines misc environment values.
#########################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH
```

```
#########################################################################
# Step.1
# defines miscellaneous functions.
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
echo '    -i: input file'
echo '        this input file must be generated with Generate_LV_list.ksh'
echo '        and should be revised manually.'
echo '    -o: output file'
echo '        this file will be overwritten.'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and checks for the existence of input file.
#########################################################################

SHORT_USAGE="usage: $0 [-D] -i input_file"
DEBUG_FLAG="OFF"

if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :Di:o: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
        D)
        DEBUG_FLAG="ON";
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
```

```
        o)
        OUTPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

#########################################################################
# Step.3
# sets misc variables.
#########################################################################

# gets VG name.
VG_NAME=`egrep -v '^(#|=)' ${INPUTFILE} | head -1`

# sets the path for the mklv command.
if [ ${DEBUG_FLAG} = "ON" ]; then
    MKLV='echo /usr/sbin/mklv';
else
    MKLV='echo /usr/sbin/mklv';
fi

#########################################################################
# Step.4
# creates statemap logical volumes.
#########################################################################

# skips unnecessary lines from the input file and stores it in a temp file.
tail +10 ${INPUTFILE} | egrep -v '^(=|#)' > /tmp/$$."Create_remote_LV".$$

> $OUTPUTFILE
while read LVNAME LVTYPE PP_NUM GMD_MINOR_NUM NEW_LVNAME NEW_LVTYPE
do
    case ${NEW_LVTYPE} in
    gmdlog|gmd)
        ${MKLV} -t ${NEW_LVTYPE} -y ${NEW_LVNAME} ${VG_NAME} ${PP_NUM}
        ;;
    *)
```

```
        continue
        ;;
    esac
done < /tmp/$$."Create_remote_LV".$$ >> $OUTPUTFILE

# removes temporary file.
/usr/bin/rm /tmp/$$."Create_remote_LV".$$

exit 0;

#
# end of Generate_remote_LV_script.ksh
#
```

A.6 Generate_GMD_script.ksh

The following script generates a script that can be used to create the GMDs on all the nodes. In order for the Generate_GMD_script.ksh script to work properly, the file Generate_GMD_script.awk must be located in the current directory.

```
#!/bin/ksh -x
#
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Generate_GMD_script.ksh -- Generates a shell script to create the
#                            GMDs in all nodes.
###############################################################################

###############################################################################
# Step.0
# defines misc environment values.
###############################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
```

```
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH

#########################################################################
# Step.1
# defines miscellaneous functions.
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
print -u2 '    -i: input file'
print -u2 '        this input file is generated with Generate_LV_list.ksh'
print -u2 '        and should be revised manually.'
print -u2 '    -o: output file'
print -u2 '        this file will be overwritten.'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and checks for the existence of the input
file.
#########################################################################

SHORT_USAGE="usage: $0 [-D] -i input_file -o output_file"
DEBUG_FLAG="OFF"

if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :Di:o: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
        D)
        DEBUG_FLAG="ON";
```

```
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        o)
        OUTPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

#########################################################################
# Step.3
# sets misc variables.
#########################################################################

# gets VG name.
VG_NAME=`egrep -v '^(#|=)' ${INPUTFILE} | head -1`

# sets the path for the mklv command.
if [ ${DEBUG_FLAG} = "ON" ]; then
    MKDEV='echo /usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a
device_mode=sync -a device_role=none';
else
    MKDEV='/usr/sbin/mkdev -c geo_mirror -s gmd -t lgmd -S -a
device_mode=sync -a device_role=none';
fi

#########################################################################
# Step.4
# creates temporary file.
#########################################################################

# skips unnecessary lines from input file and stores it in temp#1 file.
tail +10 ${INPUTFILE} | egrep -v '^#' > /tmp/$$."Generate_GMD_script1".$$

awk -f /usr/local/hageo/Generate_GMD_script.awk
/tmp/$$."Generate_GMD_script1".$$\
```

```
    > /tmp/$$."Generate_GMD_script2".$$

# removes temp#1 file.
/usr/bin/rm /tmp/$$."Generate_GMD_script1".$$

# skips unnecessary lines from temp#2 file and store it in temp#3 file.
egrep -v '^#' /tmp/$$."Generate_GMD_script2".$$\
    > /tmp/$$."Generate_GMD_script3".$$
# removes temp#2 file.
/usr/bin/rm /tmp/$$."Generate_GMD_script2".$$

#########################################################################
# Step.5
# prints header lines.
#########################################################################

> ${OUTPUTFILE}
echo '#!/bin/ksh' >> ${OUTPUTFILE}
echo '#' >> ${OUTPUTFILE}
echo '# This script is generated by Generate_GMD_script.ksh' >>
${OUTPUTFILE}
echo "# on node: " `get_local_nodename` >> ${OUTPUTFILE}
echo "# date: " `date` >> ${OUTPUTFILE}
echo '#' >> ${OUTPUTFILE}
echo '' >> ${OUTPUTFILE}
echo 'LOCALNODENAME=`/usr/sbin/cluster/utilities/get_local_nodename`' >>
${OUTPUTFILE}
echo '' >> ${OUTPUTFILE}
echo 'case $LOCALNODENAME in'  >> ${OUTPUTFILE}


#########################################################################
# Step.6
# gets local/remote peer names. Header lines.
#########################################################################

while read MINOR GMD_NAME GMD_LV LOCAL_PEER1 LOCAL_SM1 LOCAL_PEER2
LOCAL_SM2 REMOTE_PEER1 REMOTE_SM1 REMOTE_PEER2 REMOTE_SM2
do
    if [ ${LOCAL_PEER1} = "=" ]; then
        print -u2 "Error: there is no local peer.";
        exit 1;
    else
        LOCAL_PEER_COUNT=1;
        L1=${LOCAL_PEER1};
    fi
```

```
    if [ ${LOCAL_PEER2} = "=" ]; then
        L2=""
        L_SM2=""
    else
        LOCAL_PEER_COUNT=2;
        L2=${LOCAL_PEER2};
        L_SM2=${LOCAL_SM2}
    fi

    if [ ${REMOTE_PEER1} = "=" ]; then
        print -u2 "Error: there is no remote peer.";
        exit 1;
    else
        REMOTE_PEER_COUNT=1;
        R1=${REMOTE_PEER1};
    fi

    if [ ${REMOTE_PEER2} = "=" ]; then
        R2=""
        R_SM2=""
    else
        REMOTE_PEER_COUNT=2;
        R2=${REMOTE_PEER2};
        R_SM2=${REMOTE_SM2}
    fi
    break;
done < /tmp/$$."Generate_GMD_script3".$$


#########################################################################
# Step.7
# case of $L1
#########################################################################
echo "$L1"')' >> ${OUTPUTFILE}

while read MINOR GMD_NAME GMD_LV LOCAL_PEER1 LOCAL_SM1 LOCAL_PEER2
LOCAL_SM2 REMOTE_PEER1 REMOTE_SM1 REMOTE_PEER2 REMOTE_SM2
do
    if ((${LOCAL_PEER_COUNT} == 1)) && ((${REMOTE_PEER_COUNT} == 1))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${LOCAL_SM1}'\' >> ${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER1}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    elif ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 1))
```

```
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${LOCAL_SM1}'\' >> ${OUTPUTFILE}
        echo ' -a locpeer_smdev='${LOCAL_PEER2}'@/dev/r'${LOCAL_SM2}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER1}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    elif ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 2))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${LOCAL_SM1}'\' >> ${OUTPUTFILE}
        echo ' -a locpeer_smdev='${LOCAL_PEER2}'@/dev/r'${LOCAL_SM2}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER1}'@/dev/r'${GMD_LV}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER2}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    else
        print -u2 "Unsupported local/remote peers combination."
        exit 1;
    fi
done < /tmp/$$."Generate_GMD_script3".$$

echo ';;' >> ${OUTPUTFILE}

#########################################################################
# Step.8
# case of $LOCAL_PEER2
#########################################################################
if ((${LOCAL_PEER_COUNT} == 2))
then
###

echo "$L2"')' >> ${OUTPUTFILE}

while read MINOR GMD_NAME GMD_LV LOCAL_PEER1 LOCAL_SM1 LOCAL_PEER2
LOCAL_SM2 REMOTE_PEER1 REMOTE_SM1 REMOTE_PEER2 REMOTE_SM2
do
    if ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 1))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
```

```
        echo ' -a state_map_dev=/dev/r'${LOCAL_SM2}'\' >> ${OUTPUTFILE}
        echo ' -a locpeer_smdev='${LOCAL_PEER1}'@/dev/r'${LOCAL_SM1}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER1}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    elif ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 2))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${LOCAL_SM2}'\' >> ${OUTPUTFILE}
        echo ' -a locpeer_smdev='${LOCAL_PEER1}'@/dev/r'${LOCAL_SM1}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER1}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
        echo ' -a remote_device='${REMOTE_PEER2}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    else
        print -u2 "Unsupported local/remote peers combination."
        exit 1;
    fi
done < /tmp/$$."Generate_GMD_script3".$$

echo ';;' >> ${OUTPUTFILE}

###
fi
#########################################################################
# Step.9
# case of $REMOTE_PEER1
#########################################################################
echo "$R1"')' >> ${OUTPUTFILE}

while read MINOR GMD_NAME GMD_LV LOCAL_PEER1 LOCAL_SM1 LOCAL_PEER2
LOCAL_SM2 REMOTE_PEER1 REMOTE_SM1 REMOTE_PEER2 REMOTE_SM2
do
    if ((${LOCAL_PEER_COUNT} == 1)) && ((${REMOTE_PEER_COUNT} == 1))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${REMOTE_SM1}'\' >> ${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER1}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    elif ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 1))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
```

```
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${REMOTE_SM1}'\' >> ${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER1}'@/dev/r'${GMD_LV}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER2}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    elif ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 2))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${REMOTE_SM1}'\' >> ${OUTPUTFILE}
        echo ' -a locpeer_smdev='${REMOTE_PEER2}'@/dev/r'${REMOTE_SM2}'\'
>> ${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER1}'@/dev/r'${GMD_LV}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER2}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    else
        print -u2 "Unsupported local/remote peers combination."
        exit 1;
    fi
done < /tmp/$$."Generate_GMD_script3".$$

echo ';;' >> ${OUTPUTFILE}

#########################################################################
# Step.10
# case of $REMOTE_PEER2
#########################################################################
if ((${REMOTE_PEER_COUNT} == 2))
then
###

echo "$R2"')' >> ${OUTPUTFILE}

while read MINOR GMD_NAME GMD_LV LOCAL_PEER1 LOCAL_SM1 LOCAL_PEER2
LOCAL_SM2 REMOTE_PEER1 REMOTE_SM1 REMOTE_PEER2 REMOTE_SM2
do
    if ((${LOCAL_PEER_COUNT} == 2)) && ((${REMOTE_PEER_COUNT} == 2))
    then
        echo $MKDEV'\' >> ${OUTPUTFILE}
        echo ' -l '${GMD_NAME}' -a minor_num='${MINOR}'\' >> ${OUTPUTFILE}
        echo ' -a local_device=/dev/r'${GMD_LV}'\' >> ${OUTPUTFILE}
        echo ' -a state_map_dev=/dev/r'${REMOTE_SM2}'\' >> ${OUTPUTFILE}
```

```
        echo ' -a locpeer_smdev='${REMOTE_PEER1}'@/dev/r'${REMOTE_SM1}'\'
>> ${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER1}'@/dev/r'${GMD_LV}'\' >>
${OUTPUTFILE}
        echo ' -a remote_device='${LOCAL_PEER2}'@/dev/r'${GMD_LV} >>
${OUTPUTFILE}
    else
        print -u2 "Unsupported local/remote peers combination."
        exit 1;
    fi
done < /tmp/$$."Generate_GMD_script3".$$

echo ';;' >> ${OUTPUTFILE}
echo 'esac' >>  ${OUTPUTFILE}

###
fi
#########################################################################
# Step.11
# prints footer information to output file.
#########################################################################
echo 'esac' >> ${OUTPUTFILE}
echo '' >> ${OUTPUTFILE}
echo '#' >> ${OUTPUTFILE}
echo '# end of file.' >> ${OUTPUTFILE}
echo '#' >> ${OUTPUTFILE}

# removes temp#3 file.
# /usr/bin/rm /tmp/$$."Generate_GMD_script3".$$

exit 0;

#
# end of Generate_GMD_script.ksh
#
```

A.7 Generate_GMD_script.awk

The following file is used by the Generate_GMD_script.ksh script.

```
#!/usr/bin/awk
#
# Generate_GMD_Script.awk -- parses temporary file.
#

BEGIN {
```

```
#
# prints header lines.
#

printf("# List of GMD names generated with Generate_GMD_Script.ksh\n");
printf("#\n");
printf("%4s%16s%16s%10s%16s%10s%16s%10s%16s%10s%18s\n"\
    , "#GMD#", "GMD name", "GMD LV name", "local#1", "local#1 SM"\
    , "local#2", "local#2 SM", "remote#1", "remote#1 SM"\
    , "remote#2", "remote#2 SM");
printf("#\n");
}


{
if ($6 != "sm") {
# This line does not represent a statemap LV.
    gmd_minor = $4;
    gmd_name[gmd_minor] = $1;
    gmd_localdev[gmd_minor] = $5;

    # clear counter.
    local_sm_counter = 0;
    remote_sm_counter = 0;

    # fill '=' value in misc variables for the statemaps.
    local_peer_1[gmd_minor]="=";
    local_peer_2[gmd_minor]="=";
    local_peer_sm_1[gmd_minor]="=";
    local_peer_sm_2[gmd_minor]="=";
    remote_peer_1[gmd_minor]="=";
    remote_peer_2[gmd_minor]="=";
    remote_peer_sm_1[gmd_minor]="=";
    remote_peer_sm_2[gmd_minor]="=";
} else {
    # This line represents a statemap LV.
    if ($1 == "=LOCAL") {
        #
        # local statemap LV.
        #

        # increment counter.
        local_sm_counter++;

        if (local_sm_counter == 1) {
            sub("=", "", $2);
            local_peer_1[gmd_minor]=$2;
```

```
            local_peer_sm_1[gmd_minor]=$5;
        } else if (local_sm_counter == 2) {
            sub("=", "", $2);
            local_peer_2[gmd_minor]=$2;
            local_peer_sm_2[gmd_minor]=$5;
        }
    } else {
        #
        # remote statemap LV.
        #

        # increment counter.
        remote_sm_counter++;

        if (remote_sm_counter == 1) {
            sub("=", "", $2);
            remote_peer_1[gmd_minor]=$2;
            remote_peer_sm_1[gmd_minor]=$5;
        } else if (remote_sm_counter == 2) {
            sub("=", "", $2);
            remote_peer_2[gmd_minor]=$2;
            remote_peer_sm_2[gmd_minor]=$5;
        }
    }
}
}

END {
max_gmd_minor=gmd_minor;

for (i = 0; i <= max_gmd_minor; i++) {
    printf("%-4d%16s%16s%10s%16s%10s%16s%10s%16s%10s%18s\n"\
, i, gmd_name[i], gmd_localdev[i]\
        , local_peer_1[i], local_peer_sm_1[i]\
        , local_peer_2[i], local_peer_sm_2[i]\
        , remote_peer_1[i], remote_peer_sm_1[i]\
        , remote_peer_2[i], remote_peer_sm_2[i]);
}

#
# print footer lines.
#

printf("#\n");
printf("# End of file.\n");
printf("#\n");
}
```

```
#
# end of Generate_GMD_Script.awk
#
```

A.8 Generate_LV_action.ksh

The following script generates an intermediate file with new names and types for some logical volumes. It also lists which logical volumes will be removed and which will not be changed.

```
#!/bin/ksh -x
#
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Generate_LV_action.ksh -- generates a list of LVs with actions to be
taken.
##############################################################################

##############################################################################
# Step.0
# defines environment variables.
##############################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH

##############################################################################
# Step.1
# defines miscellaneous functions.
##############################################################################

# function:     long_usage
```

```
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
echo '    -i: input file'
echo '        this input file is the output of the lsvg -l <vg_name>
command'
echo '    -o: output file'
echo '        this file will be overwritten.'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and check for the existence of input file.
#########################################################################

SHORT_USAGE="usage: $0 -i input_file -o output_file"

if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :i:o: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        o)
        OUTPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done
```

```
if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
fi

# gets VG name from input file.
VG_NAME=`head -1 ${INPUTFILE} | cut -d: -f 1`

#########################################################################
# Step.3
# generates output file.
#########################################################################

# prints header in the output report
> ${OUTPUTFILE}
echo "# List of LV names generated with Generate_LV_action.ksh" >>
${OUTPUTFILE}
echo "# on node: " `get_local_nodename` >> ${OUTPUTFILE}
echo "# date: " `date` >> ${OUTPUTFILE}
echo "#" >> ${OUTPUTFILE}
echo ${VG_NAME} >> ${OUTPUTFILE}
echo "#" >> ${OUTPUTFILE}
echo "#Original          Original   New               New        Action to"
>> ${OUTPUTFILE}
echo "#LV_name           LV_type    LV_name           LV_type    be taken"
>> ${OUTPUTFILE}
echo "# you should not change above this line." >> ${OUTPUTFILE}

# sets print field
typeset -L10 L10_LVTYPE
typeset -L18 L18_LVNAME
typeset -L10 L10_NEW_LVTYPE
typeset -L18 L18_NEW_LVNAME
typeset -L10 L10_ACTION

# skips first two lines from input file and creates a temporary file.
tail +3 ${INPUTFILE} > /tmp/$$."Generate_LV_action".$$

while read LVNAME LVTYPE LP_NUM PP_NUM PV_NUM LV_STATE MOUNT_POINT
do
    # checks LVTYPE
    case $LVTYPE in
    "gmdlog")
        NEW_LVTYPE="jfslog"
        ;;
    "gmd")
        NEW_LVTYPE="jfs"
```

```
        ;;
    "sm")
        NEW_LVTYPE="N/A"
        ;;
    *)
        NEW_LVTYPE=$LV_TYPE
        ;;
    esac

    # generates new LV name.
    if [ $LVTYPE = "sm" ]; then
        NEW_LVNAME="N/A"
    ACTION="remove"
    else
GMDs_TO_SEARCH=`lsdev -Cc geo_mirror | awk '{print $1}'`
NEW_LVNAME=""
for i in $GMDs_TO_SEARCH
do
POSSIBLE_LV=`lsattr -El $i -a local_device | awk '{print $2}' | cut -c 7-`
if [ $POSSIBLE_LV = $LVNAME ]; then
NEW_LVNAME=$i
NEW_LVTYPE="jfs"
ACTION="rename"
fi
done

if [ ${#NEW_LVNAME} -eq 0 ]; then
NEW_LVNAME=N/A
NEW_LVTYPE=N/A
ACTION="none"
fi
fi

    L10_LVTYPE=$LVTYPE
    L18_LVNAME=$LVNAME
    L10_NEW_LVTYPE=$NEW_LVTYPE
    L18_NEW_LVNAME=$NEW_LVNAME
    L10_ACTION=$ACTION

    # prints lines to output file
    print "$L18_LVNAME $L10_LVTYPE $L18_NEW_LVNAME $L10_NEW_LVTYPE
$L10_ACTION"
done < /tmp/$$."Generate_LV_action".$$ >> $OUTPUTFILE

# prints footer
echo "#" >> $OUTPUTFILE
echo "#end of file" >> $OUTPUTFILE
```

```
echo "#" >> $OUTPUTFILE
# removes temporary file
rm /tmp/$$."Generate_LV_action".$$
exit 0;

#
# end of Generate_LV_action.ksh
#
```

A.9 Take_action_LV.ksh

The following script takes as input the file generated by the Generate_LV_action.ksh script and executes the actions described in the file.

```
#!/bin/ksh -x
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1999
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#
# Take_action_LV.ksh -- executes the actions described in the input file
#                       and generates the Remove_GMDs.ksh.
################################################################################

#######################################################################
# Step.0
# defines environment variables.
#######################################################################
LANG=en_US
ODMDIR=/etc/objrepos
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin
HACMP_PATH=/usr/sbin/cluster:/usr/sbin/cluster/utilities:/usr/sbin/cluster
/diag
HAGEO_PATH=/usr/sbin/krpc:/usr/sbin/gmd:/usr/lib/methods
PATH=$PATH:${HACMP_PATH}:${HAGEO_PATH}

export LANG ODMDIR PATH

#######################################################################
# Step.1
# defines miscellaneous functions.
```

```
#########################################################################

# function:     long_usage
# arguments:    NONE
# return value: NONE
# Note:         exit with return code 1.

function long_usage
{
echo '    -i: input file'
echo '        this file is the output of Generate_LV_action.ksh'
exit 1;
}

#########################################################################
# Step.2
# handles command line options and checks for the existence of input file.
#########################################################################

SHORT_USAGE="usage: $0 -i input_file"

if [ $# -eq 0 ]; then
    echo ${SHORT_USAGE}
    exit 1;
fi

while getopts :i: arguments
do
    case $arguments in
        :)
        echo "switch -${OPTARG} needs optional arguments".
        long_usage;
        ;;
        i)
        INPUTFILE=${OPTARG};
        ;;
        \?)
        print "${OPTARG} is not a valid switch.";
        print "${SHORT_USAGE}.";
        exit 1;
        ;;
    esac
done

if [ ! -f ${INPUTFILE} ]; then
    echo "Error: input file \"${INPUTFILE}\" does not exist."
    exit 1;
```

```
fi

#########################################################################
# Step.3
# executes actions and generates Remove_GMDs.ksh script
#########################################################################

# gets VG name from input file.
VG_NAME=`cat $INPUTFILE | head -5 | tail -1`


# skips commented lines from input file and creates temporary file.
cat ${INPUTFILE} | tail +10 | grep -v '^#' > /tmp/$$."Take_action_LV".$$

# creates header in Remove_GMDs.ksh script
echo "#!/bin/ksh" > Remove_GMDs.ksh
echo "#" >> Remove_GMDs.ksh
echo "# Remove_GMDs.ksh -- removes the GMDs in the local peers." >>
Remove_GMDs.ksh
echo
"#########################################################################"
>> Remove_GMDs.ksh


while read LVNAME LVTYPE NEW_LVNAME NEW_LVTYPE ACTION
do
case $ACTION in
"none")
;;
"remove")
echo "Removing logical volume $LVNAME"
rmlv -f $LVNAME
;;
"rename")
# checks if there is a GMD with the NEW_LVNAME
DEV_EXIST=`lsdev -Cl $NEW_LVNAME`
if [ ${#DEV_EXIST} -ne 0 ]; then
echo $DEV_EXIST | grep "Geographic Mirror Device" > /dev/null
if [ $? -eq 0 ]; then
echo "Removing GMD $NEW_LVNAME"
rmdev -l $NEW_LVNAME -d
echo "rmdev -l $NEW_LVNAME -d" >> Remove_GMDs.ksh
echo "Changing LV name from $LVNAME to $NEW_LVNAME"
chlv -n $NEW_LVNAME $LVNAME
echo "Changing LV type of $NEW_LVNAME from $LVTYPE to $NEW_LVTYPE"
chlv -t $NEW_LVTYPE $NEW_LVNAME
else
```

```
echo "Error: Cannot rename $LVNAME to $NEW_LVNAME"
    echo "The device name has already been assigned to another"
    echo "device, which is not a GMD. Choose a different name."
fi
else
echo "Changing LV name from $LVNAME to $NEW_LVNAME"
chlv -n $NEW_LVNAME $LVNAME
echo "Changing LV type of $LV_NAME from $LVTYPE to $NEW_LVTYPE"
chlv -t $NEW_LVTYPE $NEW_LVNAME
fi
;;
esac


done < /tmp/$$."Take_action_LV".$$ >> $OUTPUTFILE

# removes temporary file
rm /tmp/$$."Take_action_LV".$$

exit 0;

#
# end of Take_action_LV.ksh
#
```

Appendix B. geonet

geonet is a tool for measuring the time it takes to transmit and receive packets of various sizes across a network.

B.1 Copyright.h

```
//
// Licensed Materials - Property of IBM
//
// (C) COPYRIGHT International Business Machines Corp. 1999
// All Rights Reserved
//
// US Government Users Restricted Rights - Use, duplication or
// disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//
```

B.2 Makefile

```
CFLAGS=-O -qcpluscmt
#CFLAGS=-g -qcpluscmt
all: udpcli udpserv
udpcli: udpcli.o timers.o memdump.o
$(CC) $(CFLAGS) -o $@ udpcli.o timers.o memdump.o
udpserv: udpserv.o memdump.o
$(CC) $(CFLAGS) -o $@ udpserv.o memdump.o
```

B.3 udpcli.c

```
/*
* Example of client using UDP protocol.
*/
#include "timers.h"
#include "geonet.h"
#include "Copyright.h"
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#ifdef _AIX
#include <sys/lock.h>
#endif
```

```
main(argc, argv)
int argc;
char *argv[];
{
int sockfd;
struct sockaddr_in cli_addr, serv_addr;
struct sockaddr_in *pserv_addr;
struct hostent *ptemp,dest;
int on=1, servlen, verbose=TRUE;
int n=0, i=0;
char errmsg[255];
char sendbuf[MAXRECV];
char recvbuf[MAXRECV];
struct timeval now, then, diff, average, *results;
struct message {
int tsize;
int rsize;
struct timeval time;
} message, *pmessage;
int OptionValue=0;
int rc=0;
size_t OptionLength=sizeof(OptionValue);
extern char *optarg;
extern int optind;
signed char option;
char *usage="[-c] [-c] [-p port] [-n number of packets] [-t transmit bytes]
[-r receive bytes] [-v] [-q] [-h] [-?] destination IP address\n";
int flags=0, port=7, numwrites=1, tsize=4572, rsize=276;
/* Process parameters */
while ((option=getopt(argc,argv,"bch?vdp:n:t:r:"))!=EOF)
{
switch (option)
{
case 'b':
flags = flags | BROADCAST;
fprintf(stderr,"Setting BROADCAST\n");
break;
case 'c':
flags = flags | CONNECT;
fprintf(stderr,"Setting CONNECT\n");
break;
case 'p':
port=atoi(optarg);
fprintf(stderr,"Setting port to %d\n", port);
break;
case 'n':
numwrites=atoi(optarg);
```

```
break;
case 't':
tsize=atoi(optarg);
if ( tsize < 16 ) {
printf("transmit size too small (min %d)\n", 16);
exit(1);
}
break;
case 'r':
rsize=atoi(optarg);
if ( rsize < 16 ) {
printf("return size too small (min %d)\n", 16);
exit(1);
}
break;
case 'h':
case '?':
printf("\nUsage: %s %s\n",argv[0],usage);
printf(" -p server port number (default is 7 - echo)\n");
printf(" -n number of packets to send\n");
printf(" -b allow broadcast\n");
printf(" -c run connect()\n");
printf(" -t size in bytes of transmit packet\n");
printf(" -r size in bytes of return packet\n");
printf(" -? Print this help message\n");
printf(" -v Verbose option (default=no)\n");
printf(" -d Debug option (default=no)\n");
exit(0);
break;
case 'd':
flags = flags | DEBUG | VERBOSE;
break;
case 'v':
flags = flags | VERBOSE;
break;
default:
printf("Error: Invalid option\n");
printf("Usage: %s %s\n",argv[0],usage);
exit(0);
break;
}
}
if (optind < argc) /* check for leftover arguments */
printf("Will send to %s\n", argv[optind]);
else {
printf("Error: destination IP address\n");
printf("Usage: %s %s",argv[0],usage);
```

```
exit(1);
}
results=(struct timeval *)malloc(numwrites*sizeof(struct timeval));
if (results == NULL) printf("malloc() error\n"), exit(-1);
#ifdef _AIX
if (plock(PROCLOCK) < 0) perror("plock()");
#endif
pname = argv[0];
n = 4000;
/*
* Fill in the structure "serv_addr" with the address of the
* server that we want to send to.
*/
/*
if ( argv[1] != NULL )
{
ptemp=gethostbyname(argv[1]);
sprintf(errmsg,"gethostbyname(%s)",argv[1]);
if (ptemp == NULL) fprintf(stderr,"%s - host not found\n",errmsg);
else
printf("%s is %x\n", argv[1], (unsigned long int *)*ptemp->
h_addr);
}
*/
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
if ( optind < argc )
serv_addr.sin_addr.s_addr = inet_addr(argv[optind]);
else
serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
//serv_addr.sin_port = htons(SERV_UDP_PORT);
serv_addr.sin_port = htons((unsigned short)port);
if (flags & DEBUG) memdump(&serv_addr,sizeof(serv_addr));
/*
* Open a UDP socket (an Internet datagram socket).
*/
if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
printf("client: can't open datagram socket");
/*
* Bind any local address for us.
*/
bzero((char *) &cli_addr, sizeof(cli_addr)); /* zero out */
cli_addr.sin_family = AF_INET;
cli_addr.sin_addr.s_addr = htonl(INADDR_ANY);
// cli_addr.sin_addr.s_addr = inet_addr("9.180.72.21");
cli_addr.sin_port = htons(0);
if (bind(sockfd, (struct sockaddr *) &cli_addr, sizeof(cli_addr)) < 0)
```

```
printf("client: can't bind local address");
if (flags & DEBUG) printf("cli_addr:\n");
if (flags & DEBUG) fmemdump(stdout,&cli_addr, sizeof(cli_addr));
if (flags & DEBUG) printf("serv_addr:\n");
if (flags & DEBUG) fmemdump(stdout,&serv_addr, sizeof(serv_addr));
if (flags & BROADCAST) setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &on,
sizeof(on));
if (flags & DEBUG) printf("OptionLength is %d\n", OptionLength);
if (flags & DEBUG) printf("pmtu is %d\n", OptionValue);
if (flags & CONNECT) {
if (flags & DEBUG) printf("Connecting\n");
rc=connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr));
if (rc!=0) perror("connect()");
}
rc=getsockopt(sockfd, IPPROTO_IP, IP_FINDPMTU, &OptionValue,
&OptionLength);
if (flags & DEBUG) printf("rc is %d\n", rc);
if (flags & DEBUG) printf("pmtu is %d\n", OptionValue);
if (flags & DEBUG) printf("OptionLength is %d\n", OptionLength);
switch(OptionValue) {
case 0:
printf("PMTU discovery is not enabled\n");
break;
case -1:
printf("PMTU discovery is not complete\n");
break;
default:
printf("PMTU is %d\n", OptionValue);
break;
}
pserv_addr=&serv_addr;
servlen=sizeof(serv_addr);
if (flags & DEBUG) if (gettimeofday(&now,0) != 0) perror("gettimeofday()");
if (flags & DEBUG) if (gettimeofday(&then,0) != 0)
perror("gettimeofday()");
if (flags & DEBUG) printtimeval(&now,"First");
if (flags & DEBUG) printtimeval(&then,"Second");
if (flags & DEBUG) subtimeval(&diff,&then,&now);
if (flags & DEBUG) printtimeval(&diff,"Difference");
message.tsize=htonl(tsize);
message.rsize=htonl(rsize);
for (i=0; i<numwrites; i++) {
if (gettimeofday(&now,0) != 0) perror("gettimeofday()");
message.time=now;
if (flags & DEBUG) printf("Before sendto():\n");
if (flags & DEBUG) printf("cli_addr:\n");
if (flags & DEBUG) fmemdump(stdout,&cli_addr, sizeof(cli_addr));
```

```
if (flags & DEBUG) printf("serv_addr:\n");
if (flags & DEBUG) fmemdump(stdout,&serv_addr, sizeof(serv_addr));
if (flags & VERBOSE) printf("Sending %d bytes\n", tsize);
if (flags & CONNECT) {
if (send(sockfd, &message, tsize, 0) != tsize)
perror("dg_cli: send error on socket");
}
else
if (sendto(sockfd, &message, tsize, 0, (struct sockaddr *)
pserv_addr, servlen) != tsize)
perror("dg_cli: sendto error on socket");
if (flags & DEBUG) printf("Before recvfrom():\n");
if (flags & DEBUG) printf("cli_addr:\n");
if (flags & DEBUG) fmemdump(stdout,&cli_addr, sizeof(cli_addr));
if (flags & DEBUG) printf("serv_addr:\n");
if (flags & DEBUG) fmemdump(stdout,&serv_addr, sizeof(serv_addr));
if (flags & VERBOSE) printf("Waiting for reply\n");
if (flags & CONNECT) {
n = recv(sockfd, recvbuf, MAXRECV, 0);
if ( (n==0) || (n==-1) ) {
fprintf(stderr,"recv()=%d\n",n);
perror("dg_cli: recv error on socket");
}
}
else {
n = recvfrom(sockfd, recvbuf, MAXRECV, 0,
(struct sockaddr *) 0, (unsigned long *) 0);
if ( (n==0) || (n==-1) ) {
fprintf(stderr,"recvfrom()=%d\n",n);
perror("dg_cli: recvfrom error on socket");
}
}
if (gettimeofday(&now,0) != 0) perror("gettimeofday()");
if (flags & VERBOSE) printf("Received %d bytes\n", n, recvbuf);
pmessage=(struct message *)recvbuf;
if (flags & DEBUG) printtimeval(&now,"Received at");
if (flags & DEBUG) printtimeval(&(pmessage->time),"Sent at");
subtimeval(&diff,&now,&(pmessage->time));
//printtimeval(&diff,"Difference");
results[i]=diff;
if (n < 0)
printf("dg_cli: recvfrom error");
}
for (i=0; i<numwrites; i++) {
if (flags & VERBOSE) printtimeval(&results[i],"Round-trip");
}
meantimeval(&average, results, numwrites);
```

```
close(sockfd);
exit(0);
}
```

udpserv.c

```
/*
* Example of server using UDP protocol.
*/
#include "Copyright.h"
#include "geonet.h"
#include <sys/time.h>
#ifdef _AIX
#include <sys/lock.h>
#endif
main(argc, argv)
int argc;
char *argv[];
{
int n, flags=0;
unsigned long clilen;
unsigned long serv_ipaddr;
int sockfd;
struct sockaddr_in serv_addr, cli_addr;
struct message {
int tsize;
int rsize;
struct timeval time;
char data[MAXRECV];
} message, *pmessage;
int tsize, rsize;
char *sender;
extern char *optarg;
extern int optind;
signed char option;
char *usage="[-v] [-q] [-h] [-?]\n";
/* Process parameters */
while ((option=getopt(argc,argv,"h?vdi:"))!=EOF)
{
switch (option)
{
case 'h':
case '?':
printf("\nUsage: %s %s\n",argv[0],usage);
printf(" -? Print this help message\n");
printf(" -h Print this help message\n");
printf(" -v Verbose option (default=no)\n");
printf(" -d Debug option (default=no)\n");
```

```
exit(0);
break;
case 'd':
flags = flags | DEBUG | VERBOSE;
break;
case 'v':
flags = flags | VERBOSE;
break;
case 'i':
if (flags & DEBUG) printf("IP address is %s\n", optarg);
if ( (serv_ipaddr=inet_addr(optarg)) == -1 ) {
printf("Error: %s not a valid IP address\n", optarg);
exit(1);
}
else
{
flags = flags | IPADDR;
if (flags & DEBUG) memdump(&serv_ipaddr,4);
}
break;
default:
printf("Error: Invalid option\n");
printf("Usage: %s %s\n",argv[0],usage);
exit(0);
break;
}
}
pname = argv[0];
#ifdef _AIX
if (plock(PROCLOCK) < 0) perror("plock()");
#endif
/*
* Open a UDP socket (an Internet datagram socket).
*/
if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
printf("server: can't open datagram socket");
/*
* Bind our local address so that the client can send to us.
*/
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
// serv_addr.sin_addr.s_addr = inet_addr("9.180.72.21"); // gives prot err
// serv_addr.sin_addr.s_addr = inet_addr("10.180.72.22"); // doesn't
receive
// serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR); // gives prot err
if (flags & IPADDR) {
if (flags & DEBUG) printf("IPADDR set\n");
```

```
serv_addr.sin_addr.s_addr = htonl(serv_ipaddr);
}
else
{
if (flags & DEBUG) printf("IPADDR set\n");
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY); // gives prot err
}
serv_addr.sin_port = htons(SERV_UDP_PORT);
if (flags & DEBUG) printf("serv_addr before bind()\n");
if (flags & DEBUG) memdump(&serv_addr,sizeof(serv_addr));
clilen=sizeof(cli_addr);
if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
printf("server: can't bind local address");
if (flags & VERBOSE) printf("Waiting for traffic\n");
// dg_echo(sockfd, (struct sockaddr *) &cli_addr, sizeof(cli_addr));
for (;;) {
if ( (n=recvfrom(sockfd,&message,MAXRECV,0,(struct sockaddr
*)&cli_addr,&clilen)) < 0)
fprintf(stderr,"recvfrom() error\n");
if (flags & VERBOSE)
sender=inet_ntoa(cli_addr.sin_addr);
if ((int)sender==(-1) )
printf("Received %d bytes from unknown sender\n", n);
else
printf("Received %d bytes from %s\n", n, sender);
if (flags & DEBUG) printf("cli_addr from recvfrom()\n");
if (flags & DEBUG) memdump(&cli_addr,clilen);
if (flags & DEBUG) printf("message from recvfrom()\n");
if (flags & DEBUG) memdump(&message,16);
rsize=ntohl(message.rsize);
if (flags & VERBOSE) printf("Sending %d bytes to %s\n", rsize, sender);
if ( (n=sendto(sockfd, &message, rsize, 0, (struct sockaddr
*)&cli_addr, clilen)) != rsize)
fprintf(stderr,"sendto(%d)=%d
error\n",rsize,n),perror("sendto()");
}
/* NOTREACHED */
}
```

memdump.c

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
int
fmemdump(FILE *stream, const unsigned char * const buf, const size_t size)
{
int width=20;
```

```
int last;
int i,j;
char c;
fprintf(stream,"Address to be dumped is %08x\n",buf);
fprintf(stream,"Length to be dumped is %08d\n",size);
last=size%width;
for (i=0;i<(size/20);i++)
{
for (j=0;j<20;j++)
fprintf(stream," %02x",buf[i*20+j]);
fprintf(stream,"\n");
for (j=0;j<20;j++)
if ( isprint(buf[i*20+j]) )
fprintf(stream," %c",buf[i*20+j]);
else
fprintf(stream," ");
fprintf(stream,"\n");
}
if (last != 0)
{
for (j=0;j<(last);j++)
fprintf(stream," %02x",buf[(size/20)*20+j]);
fprintf(stream,"\n");
for (j=0;j<(last);j++)
{
c=buf[(size/20)*20+j];
if ( isprint(c) )
fprintf(stream," %c",c);
else
fprintf(stream," ");
}
fprintf(stream,"\n");
}
return(0);
}
int
memdump(const char * const buf, const size_t size)
{
return fmemdump(stderr,buf,size);
}
timers.c
#include <stdio.h>
#include <sys/time.h>
#define USECPERSEC 1000000
int
maxtimeval(struct timeval *result, struct timeval *first, struct timeval
*second)
```

```
{
if ( (first->tv_sec>second->tv_sec) ||
( (first->tv_sec==second->tv_sec) && (first->tv_usec>second->tv_usec) ) ) {
result->tv_sec = first->tv_sec;
result->tv_usec = first->tv_usec;
}
else {
result->tv_sec = second->tv_sec;
result->tv_usec = second->tv_usec;
}
return 0;
}
int
mintimeval(struct timeval *result, struct timeval *first, struct timeval
*second)
{
if ( (first->tv_sec>second->tv_sec) ||
( (first->tv_sec==second->tv_sec) && (first->tv_usec>second->tv_usec) ) ) {
result->tv_sec = second->tv_sec;
result->tv_usec = second->tv_usec;
}
else {
result->tv_sec = first->tv_sec;
result->tv_usec = first->tv_usec;
}
return 0;
}
int
addtimeval(struct timeval *result, struct timeval *first, struct timeval
*second)
{
result->tv_sec = first->tv_sec + second->tv_sec;
result->tv_usec = first->tv_usec + second->tv_usec;
if (result->tv_usec > (USECPERSEC-1)) {
result->tv_sec += 1;
result->tv_usec -= USECPERSEC;
}
return 0;
}
int
/*
calling this as subtimeval(&diff,&before,&after);
remember before and after are time remaining and therefore decrease
*/
subtimeval(struct timeval *result, struct timeval *more, struct timeval
*less)
{
```

```
if ( (less->tv_usec) > (more->tv_usec) ) {
more->tv_sec -= 1;
more->tv_usec += USECPERSEC;
}
result->tv_sec = more->tv_sec - less->tv_sec;
result->tv_usec = more->tv_usec - less->tv_usec;
return 0;
}
void
inittimeval(struct timeval *var, int tv_sec, int tv_usec)
{
var->tv_sec = tv_sec;
var->tv_usec = tv_usec;
return;
}
void
fprinttimeval(FILE *stream, struct timeval *var, char *name)
{
// fprintf(stream,"Value of %s.tv_sec is %d\n",name,var->tv_sec);
// fprintf(stream,"Value of %s.tv_usec is %d\n",name,var->tv_usec);
fprintf(stream,"%s: ",name);
// fprintf(stream,"Seconds since epoch (decimal): %9d, milliseconds %9d,
microseconds
%9d.\n", var->tv_sec, var->tv_usec/1000, var->tv_usec%1000);
fprintf(stream,"%3d s %3d ms %3d us\n", var->tv_sec, var->tv_usec/1000,
var->
tv_usec%1000);
// fprintf(stream,"Seconds since epoch (hex) : %08x, microseconds %08x.\n",
var->
tv_sec, var->tv_usec);
return;
}
#define printtimeval(A,B) fprinttimeval(stderr,A,B)
int
meantimeval(struct timeval *result, struct timeval *results, int n)
{
int i;
struct timeval total, average;
inittimeval(&total,0,0);
//memdump(results,n*sizeof(struct timeval));
for (i=0; i<n; i++) {
// fprintf(stderr,"Processing %d\n",i);
addtimeval(&total,&total,&results[i]);
// fprintf(stderr,"Total response time for %d writes is %d.%06d sec\n", n,
total.tv_sec, total.tv_usec);
}
average.tv_sec=total.tv_sec/n;
```

```
//fprintf(stderr,"Mean response time for %d writes is %d.%06d sec\n", n,
average.tv_sec, average.tv_usec);
average.tv_usec=USECPERSEC/n;
//fprintf(stderr,"Mean response time for %d writes is %d.%06d sec\n", n,
average.tv_sec, average.tv_usec);
average.tv_usec=average.tv_usec*(total.tv_sec%n);
//fprintf(stderr,"Mean response time for %d writes is %d.%06d sec\n", n,
average.tv_sec, average.tv_usec);
average.tv_usec=average.tv_usec+(total.tv_usec/n);
fprintf(stderr,"Mean response time for %d writes is %d.%06d sec\n", n,
average.tv_sec, average.tv_usec);
return(0);
}
```

B.4 timers.h

```
#include <stdio.h>
#include <sys/time.h>
#define USECPERSEC 1000000
extern
int
maxtimeval(struct timeval *result, struct timeval *first, struct timeval
*second);
extern
int
mintimeval(struct timeval *result, struct timeval *first, struct timeval
*second);
extern
int
addtimeval(struct timeval *result, struct timeval *first, struct timeval
*second);
extern
int
/*
calling this as subtimeval(&diff,&before,&after);
remember before and after are time remaining and therefore decrease
*/
subtimeval(struct timeval *result, struct timeval *more, struct timeval
*less);
extern
void
inittimeval(struct timeval *var, int tv_sec, int tv_usec);
extern
void
fprinttimeval(FILE *stream, struct timeval *var, char *name);
#define printtimeval(A,B) fprinttimeval(stderr,A,B)
```

geonet.h

```
/*
* Definitions for TCP and UDP client/server programs.
*/
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAXRECV 1000000
#define MAXSEND MWXRECV
#define SERV_UDP_PORT 6666
/* This is the "echo" service - should get it from services */
//#define SERV_UDP_PORT 7
#define SERV_TCP_PORT 6666
// #define SERV_HOST_ADDR "255.255.255.255" /* host addr for server */
#define SERV_HOST_ADDR "0.0.0.0" /* host addr for server */
#define VERBOSE 1
#define DEBUG 2
#define IPADDR 4
#define CONNECT 8
#define BROADCAST 16
char *pname;
```

Appendix C. diskio2

diskio2 is a tool for generating disk I/O workloads with various characteristics.

C.1 Copyright.h

```
//
// Licensed Materials - Property of IBM
//
// (C) COPYRIGHT International Business Machines Corp. 1999
// All Rights Reserved
//
// US Government Users Restricted Rights - Use, duplication or
// disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//
```

C.2 Makefile

```
CC=cc
CFLAGS=-O -qcpluscmt
#CFLAGS=-g -v -qcpluscmt
LDFLAGS=-lc -ls
DISKIO2= diskio2.o timers.o memdump.o
all: diskio2
diskio2: ${DISKIO2}
${CC} ${CFLAGS} -o $@ ${DISKIO2}
clean:
rm ${DISKIO2} diskio2
```

C.3 diskio2.c

```
// #define _LARGE_FILES
#include "Copyright.h"
#include <fcntl.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/devinfo.h>
#include <sys/errno.h>
#include <sys/events.h>
#include <sys/ioctl.h>
#include <sys/lock.h>
#include <sys/stat.h>
#include <sys/time.h>
```

```
#include <sys/time.h>
#include <sys/timer.h>
#include <sys/types.h>
#include <sys/processor.h>
#include <unistd.h>
#include <math.h>
#define SERIAL 1
#define RANDOM 2
#define READ 4
#define NSECPERSEC 1000000000
#ifndef _FDIRECT
#define _FDIRECT 0x08000000
#endif
#ifndef O_DIRECT
#define O_DIRECT _FDIRECT
#endif
/* Global Variable */
int io_running=FALSE;
int io_interrupted=FALSE;
int io_times_interrupted=0;
int io_total_times_interrupted=0;
int io_signal=0;
void
SigAlarmHandler(int signal)
{
if (io_running==TRUE) {
io_interrupted=TRUE;
io_times_interrupted+=1;
io_total_times_interrupted+=1;
io_signal=signal;
}
return;
}
void
usage(char progname[])
{
fprintf(stderr,
"Usage: %s [-c] [-S] [-D] [-d] [-q] [-v] [-s] [-P processor] [-r i/o s per
sec] [-n numwrites] [-f letter] [-b blocksize] [-l length] devicename\n",
progname);
fprintf(stderr,"or %s -h\n", progname);
exit(-1);
}
void
usage_long(char progname[])
{
fprintf(stderr,
```

```
"\nFlags:\n\n"
"\t-h show this help\n"
"\t-R perform reads (default is writes)\n"
"\t-c commit data to disk (O_SYNC)\n"
"\t-S commit data to disk (fsync)\n"
"\t-t write timestamp at beginning and end of data block\n"
"\t-q query file or device only (no writing)\n"
"\t-v verbose mode (print info during writes)\n"
"\t-d level debug mode (different info during writes)\n"
"\t-s serial mode rather than random (default)\n"
"\t-r i/o-rate attempt to do n i/os per second\n"
"\t-n number of writes to perform\n"
"\t-f letter use this letter to fill the write\n"
"\t-b blocksize blocksize to use\n"
"\t-l length maximum length of file to use\n"
"\t-P processor bind program to this processor\n"
"\t-D use AIX 4.3 Direct File I/O\n"
"\n"
);
exit(-1);
}
void
srandomize(unsigned int i)
{
srandom(1);
return;
}
// Scale a random number returned by random()
// Note that because we're using a remainder, then randomize(min,max) never
returns
max, only max-1
// Indeed, to guarantee this, I exit if i>=max
long
randomize(int min, int max)
{
long int i;
if ( ( (max-min) > (1<<30) ) || (min<0) || (max<0) || (min>=max) ) {
fprintf(stderr,"error: randomize(%d,%d)\n",min,max);
exit(1); }
else {
i=random();
i=(i % (max-min)) + min;
}
if ( (i<min) || (i>=max) ) {
fprintf(stderr,"Error: randomize(%d,%d)=%d\n", min, max, i);
exit(1);
}
```

```
return i;
}
int
deviceinfo(char *devname, int fd, int *numblks, int *blksize)
{
int rc;
struct devinfo tmp;
char *sdevtype, *sdevsubtype;
if (ioctl(fd,IOCINFO,&tmp) == -1) perror("ioctl()"), exit(1);
switch (tmp.devtype) {
case DD_DISK:
sdevtype="Disk";
break;
case DD_SCDISK:
sdevtype="SCSI Disk";
break;
default:
sdevtype="UNKNOWN";
break;
}
fprintf(stderr,"Type of %s is %s, ", devname, sdevtype);
switch (tmp.devsubtype) {
case DS_LV:
sdevsubtype="LV";
break;
case DS_PV:
sdevsubtype="PV";
break;
default:
sdevsubtype="UNKNOWN";
break;
}
fprintf(stderr,"subtype is %s\n", sdevtype);
switch (tmp.devtype) {
/*
case DD_SCDISK:
*/
case DD_DISK:
fprintf(stderr,
"Bytes per sector is %d, sectors per track is %d\nTracks per
cylinder is %d, numblks is %d\n",
tmp.un.dk.bytpsec,
tmp.un.dk.secptrk,
tmp.un.dk.trkpcyl,
tmp.un.dk.numblks
);
fprintf(stderr,
```

```
"Segment Size is %d, Segment Count is %d\n",
tmp.un.dk.segment_size,
tmp.un.dk.segment_count
);
*numblks=tmp.un.dk.numblks;
*blksize=tmp.un.dk.bytpsec;
break;
case DD_SCDISK:
fprintf(stderr,"Blocksize is %d, numblks is %d\n",
tmp.un.scdk.blksize,
tmp.un.scdk.numblks
);
fprintf(stderr,
"Segment Size is %d, Segment Count is %d\n",
tmp.un.scdk.segment_size,
tmp.un.scdk.segment_count
);
*numblks=tmp.un.scdk.numblks;
*blksize=tmp.un.scdk.blksize;
break;
default:
fprintf(stderr,"no further information available\n");
break;
}
return rc;
}
int
main(int argc, char **argv) {
short processor=-1, pid=0;
struct stat statbuf;
int flags=0, sync=FALSE, openflags=O_RDWR|O_EXCL, opt, mode=RANDOM,
numwrites=10,
maxwrites=10, query=FALSE, rc, blk, blksize;
off_t byte;
int numblks, rate=0, interval=0, timestamp=FALSE, debug=0, i, j, fd,
write_size=512,
verbose=FALSE;
// int minblk=0, maxblk=0;
int minwrite=0, maxwrite=0;
long int n=0, length=0;
char mychar, mymsg[255], progname[255], devname[PATH_MAX], *buf,
filler='a';
double dTotal=0, dThruput=0, dIOrate;
char sTotal[255];
struct block {
struct timeval timestamp1;
char data[496];
```

```
struct timeval timestamp2;
} curblock;
struct timeval now;
struct timezone nowtz;
struct sigaction ignore, sigalrm, savealrm;
timer_t interval_timer, response_timer;
struct itimerstruc_t zero, resolution, maximum, minimum, max_response,
min_response,
itimer, otimer, before, after, total, elapsed, diff, average, TimeOfDay;
struct timestruc_t *results;
void *target, *source;
inititimer(&max_response,0,0,0,0);
inititimer(&min_response,(NSECPERSEC-1),(NSECPERSEC-1),(NSECPERSEC-1),(NSE
CPERSEC-1));
inititimer(&minimum,0,0,0,0);
inititimer(&maximum,(NSECPERSEC-1),(NSECPERSEC-1),(NSECPERSEC-1),(NSECPERS
EC-1));
inititimer(&zero,0,0,0,0);
inititimer(&diff,0,0,0,0);
inititimer(&total,0,0,0,0);
if (plock(PROCLOCK) < 0) perror("plock()");
sigalrm.sa_handler = SigAlarmHandler;
sigalrm.sa_flags = 0;
sigemptyset(&sigalrm.sa_mask);
if (sigaction(SIGALRM, &sigalrm, &savealrm) < 0)
perror("sigaction()"),exit(-1);
strcpy(progname,argv[0]);
while((opt = getopt(argc, argv, "RSDP:td:qvcr:shn:b:l:f:")) != EOF) {
switch(opt) {
case 'd':
debug=atoi(optarg);
verbose=TRUE;
break;
case 'R':
flags=flags|READ;
break;
case 'S':
sync=TRUE;
break;
case 't':
timestamp=TRUE;
break;
case 'q':
query=TRUE;
openflags=openflags|O_RDONLY;
break;
case 'h':
```

```
usage_long(progname);
break;
case 'v':
verbose=TRUE;
break;
case 'c':
openflags=openflags|O_SYNC;
break;
case 'D':
openflags=openflags|O_DIRECT;
break;
case 'r':
rate=atoi(optarg);
interval=NSECPERSEC/rate;
itimer.it_value.tv_sec = 0;
itimer.it_value.tv_nsec = interval;
itimer.it_interval.tv_sec = 0;
itimer.it_interval.tv_nsec = interval;
break;
case 'P':
processor=atoi(optarg);
pid=getpid();
sprintf(mymsg,"bindprocessor(BINDPROCESS,%d,%d)",pid,processor);
if (bindprocessor(BINDPROCESS,pid,processor) != 0) {
perror(mymsg);
exit(1);
}
break;
case 'n':
numwrites=atoi(optarg);
break;
case 's':
mode=SERIAL;
break;
case 'f':
filler=*optarg;
break;
case 'b':
if ( sscanf(optarg,"%d%c",&write_size,&mychar) == 2 )
switch(mychar) {
case 'k':
case 'K':
write_size=atoi(optarg) * 1024;
fprintf(stderr,"Blocksize of writes is
%d\n",write_size);
break;
case 'm':
```

```
case 'M':
write_size=atoi(optarg) * 1024 * 1024;
fprintf(stderr,"Blocksize of writes is
%d\n",write_size);
break;
default:
fprintf(stderr,"Format of blocksize
incorrect\n");
usage(progname);
exit(1);
break;
}
else
write_size=atoi(optarg);
/*
if ( write_size > (32*1024) ) {
fprintf(stderr,"Sorry, %s doesn't work with write sizes
over 32k bytes\n",progname);
exit(1);
}
if ( write_size < 512 ) {
fprintf(stderr,"Sorry, %s doesn't work with write sizes
under 512 bytes\n",progname);
exit(1);
}
*/
break;
case 'l':
if ( sscanf(optarg,"%d%c",&length,&mychar) == 2 )
switch(mychar) {
case 'k':
case 'K':
length=atoi(optarg) * 1024;
fprintf(stderr,"Length of file is
%d\n",length);
break;
case 'm':
case 'M':
length=atoi(optarg) * 1024 * 1024;
fprintf(stderr,"Length of file is
%d\n",length);
break;
case 'g':
case 'G':
length=atoi(optarg) * 1024 * 1024 * 1024;
fprintf(stderr,"Length of file is
%d\n",length);
```

```
break;
default:
fprintf(stderr,"Format of length
incorrect\n");
usage(progname);
exit(1);
break;
}
else
length=atoi(optarg);
break;
} // switch
} // while getopt
if (debug>0) fprintf(stderr,"Flags: ");
if ( (flags&READ) && (debug>0) ) fprintf(stderr,"READ ");
if (debug>0) fprintf(stderr,"\n");
if ((optind+1) != argc) usage(progname);
switch (mode)
{
case RANDOM:
fprintf(stderr,"Mode is RANDOM\n");
break;
case SERIAL:
fprintf(stderr,"Mode is SERIAL\n");
break;
default:
fprintf(stderr,"Mode is ERROR\n");
break;
}
fprintf(stderr,"Number of writes is %d\n",numwrites);
strcpy(devname,argv[optind]);
if (rate != 0) {
fprintf(stderr,"Rate is %d per second\n",rate);
fprintf(stderr,"Interval is %10.0f seconds\n",(float)interval/NSECPERSEC);
fprintf(stderr,"Interval is %10.0f
milliseconds\n",(float)interval/1000000);
fprintf(stderr,"Interval is %10.0f microseconds\n",(float)interval/1000);
fprintf(stderr,"Interval is %10.0f nanoseconds\n",(float)interval);
fprintf(stderr,"Interval is %010.8f
seconds\n",(float)interval/NSECPERSEC);
fprintf(stderr,"Interval is %010.6f
milliseconds\n",(float)interval/1000000);
fprintf(stderr,"Interval is %010.3f microseconds\n",(float)interval/1000);
fprintf(stderr,"Interval is %010.0f nanoseconds\n",(float)interval);
}
if (gettimeofday(&now,&nowtz) != 0) perror("gettimeofday()");
```

```
if (verbose==TRUE) fprintf(stderr,"Seconds since epoch (decimal): %9d,
microseconds
%9d.\n", now.tv_sec, now.tv_usec);
if (verbose==TRUE) fprintf(stderr,"Seconds since epoch (hex) : %08x,
microseconds
%08x.\n", now.tv_sec, now.tv_usec);
if ( (buf=(char *)malloc(write_size)) == NULL)
perror("malloc(buf)");
if ( (results=(struct timestruc_t *)malloc(numwrites*sizeof(struct
timestruc_t))) ==
NULL)
perror("malloc(struct timerstruc [])");
for (i=1;i<=numwrites;i++) {
results[i-1].tv_sec=0;
results[i-1].tv_nsec=0;
}
if ((fd=open(devname,openflags))==-1) perror("open()"), exit(1);
if ((rc=stat(devname,&statbuf))==-1) perror("stat()"), exit(1);
if (S_ISREG(statbuf.st_mode)) {
printf("%s is a regular file\n",devname);
if (length==0) {
fprintf(stderr,"You must specify a length for a regular file\n");
exit(1);
} // if (length==0)
}
if ( (S_ISBLK(statbuf.st_mode)) || (S_ISCHR(statbuf.st_mode)) ) {
// This is as well here as anywhere
if (sync==TRUE) {
fprintf(stderr,"Warning: -S (fsync) is superfluous on special
files\n");
sync=FALSE;
}
deviceinfo(devname, fd, &numblks, &blksize);
maxwrites=(int) ( ((long)numblks*blksize) / write_size );
// maxwrites=numblks*blksize/write_size;
// Danger! Danger! Pushing arithmetic around top avoid 2GB / 512 = 4G = -1
problem
//maxwrites=numblks/write_size;
//maxwrites=blksize*maxwrites;
if (length!=0)
if (maxwrites>(length/write_size))
maxwrites=(length/write_size);
// Assumes bs>=512
// minwrite=1;
// maxwrite=maxwrites-1;
if (write_size>=512)
minwrite=1;
```

```
else
minwrite=(512/write_size)+1;
maxwrite=maxwrites-minwrite;
}
else
// REG
{
maxwrites=length/write_size;
minwrite=0;
maxwrite=maxwrites-1;
}
// if SI_ISBLK or S_ISCHR
if (S_ISBLK(statbuf.st_mode)) {
printf("%s is a block buffer device\n",devname);
}
if (S_ISCHR(statbuf.st_mode)) {
printf("%s is a raw character device\n",devname);
if (write_size%blksize != 0)
fprintf(stderr,
"Blocksize specified (%d) must be a multiple of character device
blocksize (%d)\n",
write_size, blksize), exit(1);
}
fprintf(stderr,"Numwrites=%d Maxwrites=%d Minwrite=%d
Maxwrite=%d\n",numwrites,maxwrites,minwrite,maxwrite);
if ( (numwrites>maxwrites) && (mode==SERIAL) ) {
fprintf(stderr,
"Numwrites (%d) too high, using maxwrites-minwrite (%d).\n",
numwrites, maxwrites-minwrite);
numwrites=maxwrites-minwrite;
}
if ( (S_ISREG(statbuf.st_mode)) && ( length < write_size ) ) {
fprintf(stderr,"Error write size is less than length of file, exiting\n");
exit(1);
}
lseek(fd,write_size*minwrite,SEEK_SET);
if (query==TRUE) exit(0);
if (debug>1) fprintf(stderr,"memset(%x,%x,%x)\n",buf,filler,write_size);
if (debug>1) fprintf(stderr,"memset(%d,%d,%d)\n",buf,filler,write_size);
memset(buf,filler,write_size);
if ( (interval_timer=(timer_t)gettimerid(TIMEOFDAY, DELIVERY_SIGNALS)) < 0)
perror("getitimerid(interval_timer)");
if ( (response_timer=(timer_t)gettimerid(TIMEOFDAY, DELIVERY_SIGNALS)) < 0)
perror("getitimerid(response_timer)");
if ( resabs(interval_timer, &resolution, &maximum) < 0 )
perror("resabs()");
if (verbose==TRUE) {
```

```
fprintitimer(stderr,&resolution,"resolution");
fprintitimer(stderr,&maximum,"maximum");
}
if ( incinterval(response_timer, &maximum, &otimer) < 0 )
perror("incinterval(0)");
if (rate != 0) {
if ( incinterval(interval_timer, &itimer, &otimer) < 0 )
perror("incinterval(1)");
if ( getinterval(interval_timer, &itimer) < 0 ) perror("getinterval()");
}
else {
inititimer(&itimer,1000000,0,1000000,0);
if ( incinterval(interval_timer, &itimer, &otimer) < 0 )
perror("incinterval(2)");
if ( getinterval(interval_timer, &itimer) < 0 ) perror("getinterval()");
}
srandomize(1);
for (i=1;i<=numwrites;i++) {
if (debug>0) fprintf(stderr,"Running i=%d\n",i);
switch (mode) {
case RANDOM:
// see function definition, randomize(min,max) never
returns max, only max-1
blk=randomize(minwrite,maxwrite+1);
if (debug>0)
fprintf(stderr,"randomize(%d,%d)=%d\n",minwrite,maxwrite+1,blk);
break;
case SERIAL:
break;
default:
fprintf(stderr,"SNRH: %s %d\n",__FILE__,__LINE__);
break;
}
byte=blk*write_size;
if (mode==RANDOM)
{
if (debug>0)
// fprintf(stderr,"Number %4d of %d, seeking to blk %10d, byte
%15lld\n", i, numwrites, blk, byte);
fprintf(stderr,"Number %4d of %d, seeking to blk %10d, byte
%15d\n", i, numwrites, blk, byte);
if (lseek(fd,byte,SEEK_SET) == -1) perror("lseek()"), exit(1);
}
else
if (debug>1) fprintf(stderr,"Number %4d of %d\n", i, numwrites);
if (gettimeofday(&now,&nowtz) != 0) perror("gettimeofday()");
if (timestamp==TRUE) {
```

```
memcpy(buf,&now,sizeof(now));
memcpy(buf+write_size-sizeof(now),&now,sizeof(now));
}
if (debug>2) memdump(buf,write_size);
if ( getinterval(response_timer, &before) < 0 ) perror("getinterval()");
if ( /* (rate!=0) && */ (debug>1) ) {
fprintf(stderr,"Before write()\n");
fprintitimer(stderr,&before,"before");
}
io_interrupted=FALSE; io_signal=0; io_times_interrupted=0;
io_running=TRUE;
if (flags&READ)
rc=read(fd,buf,write_size);
else
rc=write(fd,buf,write_size);
io_running=FALSE;
if (io_interrupted==TRUE)
fprintf(stderr,"SigAlarmHandler(%d) called %d times whilst i/o in
progress\n",io_signal,io_times_interrupted);
if (rc != write_size) {
sprintf(mymsg,"write(%d,buf,%d)=%d at byte %d",fd,write_size,rc,blk);
perror(mymsg) /*, exit(1) */;
}
if ( getinterval(response_timer, &after) < 0 ) perror("getinterval()");
if ( /* (rate!=0) && */ (debug>1) ) {
fprintf(stderr,"After write()\n");
fprintitimer(stderr,&after,"after");
}
// remember before and after are time remaining and therefore decrease
subitimer(&diff,&before,&after);
additimer(&total,&total,&diff);
if (debug>0) fprintf(stderr,"Total response time for %d writes is %d.%09d
sec\n", i, total.it_value.tv_sec, total.it_value.tv_nsec);
if ( /* (rate!=0) && */ (debug>0) ) {
fprintf(stderr,"Difference before and after write()\n");
fprintitimer(stderr,&diff,"diff");
}
maxitimer(&max_response,&max_response,&diff);
minitimer(&min_response,&min_response,&diff);
if (debug>0) fprintf(stderr,"Maximum response time for %d writes is %d.%09d
sec\n", numwrites, max_response.it_value.tv_sec,
max_response.it_value.tv_nsec);
if (debug>0) fprintf(stderr,"Minimum response time for %d writes is %d.%09d
sec\n", numwrites, min_response.it_value.tv_sec,
min_response.it_value.tv_nsec);
if (verbose==TRUE) {
// fprintf(stderr,"Response time for write %6d is %d.%09d sec\n", i,
```

```
diff.it_value.tv_sec, diff.it_value.tv_nsec);
results[i-1].tv_sec=diff.it_value.tv_sec;
results[i-1].tv_nsec=diff.it_value.tv_nsec;
}
if (rate != 0) pause();
} // for
if (sync==TRUE) {
rc=fsync(fd);
if (rc!=0)
perror("fsync()");
}
subitimer(&elapsed,&maximum,&after);
if (io_total_times_interrupted!=0) {
fprintf(stderr,"%d Overruns have occurred, no statistics
valid\n",io_total_times_interrupted);
}
if (verbose==TRUE)
for (i=1;i<=numwrites;i++)
fprintf(stderr,"Response time for write %6d is %d.%09d sec\n", i,
results[i-1].tv_sec, results[i-1].tv_nsec);
fprintf(stderr,"Total response time for %d writes is %d.%09d sec\n",
numwrites,
total.it_value.tv_sec, total.it_value.tv_nsec);
fprintf(stderr,"Total elapsed time for %d writes is %d.%09d sec\n",
numwrites,
elapsed.it_value.tv_sec, elapsed.it_value.tv_nsec);
average.it_value.tv_sec=total.it_value.tv_sec/numwrites;
if (debug>0) fprintf(stderr,"Mean response time for %d writes is %d.%09d
sec\n",
numwrites, average.it_value.tv_sec, average.it_value.tv_nsec);
average.it_value.tv_nsec=NSECPERSEC/numwrites;
if (debug>0) fprintf(stderr,"Mean response time for %d writes is %d.%09d
sec\n",
numwrites, average.it_value.tv_sec, average.it_value.tv_nsec);
average.it_value.tv_nsec=average.it_value.tv_nsec*(total.it_value.tv_sec%n
umwrites);
if (debug>0) fprintf(stderr,"Mean response time for %d writes is %d.%09d
sec\n",
numwrites, average.it_value.tv_sec, average.it_value.tv_nsec);
average.it_value.tv_nsec=average.it_value.tv_nsec+(total.it_value.tv_nsec/
numwrites);
fprintf(stderr,"Maximum response time for %d writes is %d.%09d sec\n",
numwrites,
max_response.it_value.tv_sec, max_response.it_value.tv_nsec);
fprintf(stderr,"Mean response time for %d writes is %d.%09d sec\n",
numwrites,
average.it_value.tv_sec, average.it_value.tv_nsec);
```

```
fprintf(stderr,"Minimum response time for %d writes is %d.%09d sec\n",
numwrites,
min_response.it_value.tv_sec, min_response.it_value.tv_nsec);
// if ( (sync==TRUE) || (debug>0) ) {
fprintf(stderr,"\nOver total I/O time (excludes fsync):\n\n");
// }
sprintf(sTotal,"%d.%09d", total.it_value.tv_sec, total.it_value.tv_nsec);
errno=0;
sprintf(mymsg,"atof(%s)",sTotal);
dTotal=atof(sTotal);
if ( ( (dTotal==0) && (errno!=0) ) || (dTotal==HUGE_VAL) ||
(dTotal==-HUGE_VAL) )
perror(mymsg);
else {
dIOrate=( (double)numwrites )/dTotal;
fprintf(stderr,"Average i/o rate for %d writes is %10.6f writes per
sec\n", numwrites, dIOrate);
dThruput=( (double) (numwrites*write_size) )/dTotal;
fprintf(stderr,"Average throughput for %d writes is %10.0f bytes per
sec\n",
numwrites, dThruput);
fprintf(stderr,"Average throughput for %d writes is %10.3f kbytes per
sec\n", numwrites, dThruput/1024);
fprintf(stderr,"Average throughput for %d writes is %10.6f Mbytes per
sec\n", numwrites, dThruput/(1024*1024));
}
// if ( (sync==TRUE) || (debug>0) ) {
fprintf(stderr,"\nOver elapsed I/O time (includes fsync):\n\n");
sprintf(sTotal,"%d.%09d", elapsed.it_value.tv_sec,
elapsed.it_value.tv_nsec);
errno=0;
sprintf(mymsg,"atof(%s)",sTotal);
dTotal=atof(sTotal);
if ( ( (dTotal==0) && (errno!=0) ) || (dTotal==HUGE_VAL) ||
(dTotal==-HUGE_VAL)
)
perror(mymsg);
else {
dIOrate=( (double)numwrites )/dTotal;
fprintf(stderr,"Average i/o rate for %d writes is %10.6f writes
per sec\n", numwrites, dIOrate);
dThruput=( (double) (numwrites*write_size) )/dTotal;
fprintf(stderr,"Average throughput for %d writes is %10.0f bytes per
sec\n", numwrites, dThruput);
fprintf(stderr,"Average throughput for %d writes is %10.3f kbytes
per sec\n", numwrites, dThruput/1024);
fprintf(stderr,"Average throughput for %d writes is %10.6f Mbytes
```

```
per sec\n", numwrites, dThruput/(1024*1024));
// }
}
exit(0);
}
```

memdump.c

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
int
memdump(const char * const buf, const size_t size)
{
int width=20;
int last;
int i,j;
char c;
fprintf(stderr,"Address to be dumped is %08x\n",buf);
fprintf(stderr,"Length to be dumped is %08d\n",size);
last=size%width;
for (i=0;i<(size/20);i++)
{
for (j=0;j<20;j++)
fprintf(stderr," %02x",buf[i*20+j]);
fprintf(stderr,"\n");
for (j=0;j<20;j++)
if ( isprint(buf[i*20+j]) )
fprintf(stderr," %c",buf[i*20+j]);
else
fprintf(stderr," ");
fprintf(stderr,"\n");
}
if (last != 0)
{
for (j=0;j<(last);j++)
fprintf(stderr," %02x",buf[(size/20)*20+j]);
fprintf(stderr,"\n");
for (j=0;j<(last);j++)
{
c=buf[(size/20)*20+j];
if ( isprint(c) )
fprintf(stderr," %c",c);
else
fprintf(stderr," ");
}
fprintf(stderr,"\n");
}
```

```
return(0);
}
```

timers.c

```
// #define _LARGE_FILES
#include <fcntl.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/devinfo.h>
#include <sys/errno.h>
#include <sys/events.h>
#include <sys/ioctl.h>
#include <sys/lock.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/time.h>
#include <sys/timer.h>
#include <sys/types.h>
#include <unistd.h>
#define SERIAL 1
#define RANDOM 2
#define NSECPERSEC 1000000000
int
maxitimer(struct itimerstruc_t *result, struct itimerstruc_t *first, struct
itimerstruc_t *second)
{
if ( (first->it_value.tv_sec>second->it_value.tv_sec) ||
( (first->it_value.tv_sec==second->it_value.tv_sec) && (first->
it_value.tv_nsec>second->it_value.tv_nsec) ) ) {
result->it_value.tv_sec = first->it_value.tv_sec;
result->it_value.tv_nsec = first->it_value.tv_nsec;
result->it_interval.tv_sec = first->it_interval.tv_sec;
result->it_interval.tv_nsec = first->it_interval.tv_nsec;
}
else {
result->it_value.tv_sec = second->it_value.tv_sec;
result->it_value.tv_nsec = second->it_value.tv_nsec;
result->it_interval.tv_sec = second->it_interval.tv_sec;
result->it_interval.tv_nsec = second->it_interval.tv_nsec;
}
return 0;
}
int
minitimer(struct itimerstruc_t *result, struct itimerstruc_t *first, struct
itimerstruc_t *second)
{
```

```
if ( (first->it_value.tv_sec>second->it_value.tv_sec) ||
( (first->it_value.tv_sec==second->it_value.tv_sec) && (first->
it_value.tv_nsec>second->it_value.tv_nsec) ) ) {
result->it_value.tv_sec = second->it_value.tv_sec;
result->it_value.tv_nsec = second->it_value.tv_nsec;
result->it_interval.tv_sec = second->it_interval.tv_sec;
result->it_interval.tv_nsec = second->it_interval.tv_nsec;
}
else {
result->it_value.tv_sec = first->it_value.tv_sec;
result->it_value.tv_nsec = first->it_value.tv_nsec;
result->it_interval.tv_sec = first->it_interval.tv_sec;
result->it_interval.tv_nsec = first->it_interval.tv_nsec;
}
return 0;
}
int
additimer(struct itimerstruc_t *result, struct itimerstruc_t *first, struct
itimerstruc_t *second)
{
result->it_value.tv_sec = first->it_value.tv_sec + second->it_value.tv_sec;
result->it_value.tv_nsec = first->it_value.tv_nsec +
second->it_value.tv_nsec;
result->it_interval.tv_sec = 0;
result->it_interval.tv_nsec = 0;
if (result->it_value.tv_nsec > (NSECPERSEC-1)) {
result->it_value.tv_sec += 1;
result->it_value.tv_nsec -= NSECPERSEC;
}
return 0;
}
int
// calling this as subitimer(&diff,&before,&after);
// remember before and after are time remaining and therefore decrease
subitimer(struct itimerstruc_t *result, struct itimerstruc_t *more, struct
itimerstruc_t *less)
{
if ( (less->it_value.tv_nsec) > (more->it_value.tv_nsec) ) {
more->it_value.tv_sec -= 1;
more->it_value.tv_nsec += NSECPERSEC;
}
result->it_value.tv_sec = more->it_value.tv_sec - less->it_value.tv_sec;
result->it_value.tv_nsec = more->it_value.tv_nsec - less->it_value.tv_nsec;
result->it_interval.tv_sec = 0;
result->it_interval.tv_nsec = 0;
return 0;
}
```

```
void
inititimer(struct itimerstruc_t *var, int vtv_sec, int vtv_nsec, int
itv_sec, int
itv_nsec)
{
var->it_value.tv_sec = vtv_sec;
var->it_value.tv_nsec = vtv_nsec;
var->it_interval.tv_sec = itv_sec;
var->it_interval.tv_nsec = itv_nsec;
return;
}
void
fprintitimer(FILE *stream, struct itimerstruc_t *var, char *name)
{
fprintf(stream,"Value of %s.it_value.tv_sec is
%d\n",name,var->it_value.tv_sec);
fprintf(stream,"Value of %s.it_value.tv_nsec is
%d\n",name,var->it_value.tv_nsec);
fprintf(stream,"Value of %s.it_interval.tv_sec is
%d\n",name,var->it_interval.tv_sec);
fprintf(stream,"Value of %s.it_interval.tv_nsec is %d\n",name,var->
it_interval.tv_nsec);
return;
}
#define printitimer(A,B) fprintitimer(stderr,A,B)
```

Appendix D. Special notices

This publication is intended to help systems engineers, service providers, and customers to successfully plan and implement an HAGEO cluster. The information in this publication is not intended as the specification of any programming interfaces that are provided by HAGEO Version 2.1. See the PUBLICATIONS section of the IBM Programming Announcement for IBM High Availability Geographic Cluster (HAGEO) for AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers

attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AS/400
CICS	CICS/6000
DB2	IBM
Netfinity	OS/2
RS/6000	SP
SP2	System/390

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Lotus Notes is a registered trademark of Lotus Development Corporation.

Other company, product, and service names may be trademarks or service marks of others.

Appendix E. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 385.

- *Bullet-Proofing Your Oracle Database with HACMP: A Guide to Implementing AIX Databases with HACMP*, SG24-4788
- *HACMP Enhanced Scalability Handbook*, SG24-5328
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, SG24-5163
- *High Availability Scenarios for Tivoli Software*, SG24-2032
- *IBM Certification Study Guide AIX HACMP*, SG24-5131
- *Migrating to HACMP/ES*, SG24-5526

E.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

E.3 Other resources

The following publications are also relevant as further information sources:

- *AIX Commands Reference V4.3*, SBOF-1877
- *GeoRM Concepts and Facilities*, *V1.1*, SC23-4307
- *GeoRM Planning and Administration Guide, V1.1*, SC23-4308
- *HACMP V4.3 AIX: Install Guide*, SC23-4278
- *HACMP V4.3 AIX: Installation and Administration Guide*, SC23-4283
- *HAGEO Concepts & Facilities Guide V2.1*, SC23-1922
- *High Availability Geographic Cluster V2.1 for AIX: Planning and Administration Guide*, SC23-1886

E.4 Referenced Web sites

The following Web sites are relevant as further information sources:

- For an IBM paper about disaster prevention, visit:
  `http://www-1.ibm.com/services/continuity/recover1.nsf/files/Downloads/$file/buscont.pdf`
- For FEMA disaster statistics, visit: `http://www.fema.gov/library/lib01.htm`
- For statistics about computer failures, visit:
  `http://www.drj.com/special/stats/tari.htm`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

ARP. Address Resolution Protocol.

ATM. Asynchronous Transfer Mode.

BISDN. Broadband ISDN.

BRI. Basic Rate Interface.

CBR. Constant bit Rate.

CSMA/CD. Carrier Sense Multiple Access with Collision Detection.

DNS. Domain Name Server.

DS Digital Signal.

FCS. Fiber Channel Switch.

FD. File Descriptor.

FDDI. Fiber Distributed Data Interface.

FRI. Frame Relay Interface.

GMD. Geo_mirror Device.

HACMP. High Availability Cluster Multi-Processing.

HAGEO. High Availability Geographic Cluster.

HWM. High Water Mark.

IBM. International Business Machines Corporation.

ICB. Individual-case Basis.

IEEE. Institute of Electrical and Electronics Engineers.

InterNIC. Internet Network Information Center.

IPAT. IP Address Takeover.

ISA. Industry Standard Architecture.

ITSO. International Technical Support Organization.

KRPC. Kernel Remote Procedure Call.

LPP. Licensed Program Product.

LVCB. Logical Volume Control Block.

LVM. Logical Volume Manager.

MAN. Metropolitan Area Networks.

MAU. Multistory Access Unit.

MCA. Micro Channel Architecture.

MTU. Maximum Transmission Unit.

MWC. Mirror Write Consistency.

NIM. New Network Interface Modules.

OAM&P. Operations, Administration, Maintenance, and Provisioning.

ODM. Object Data Manager.

PCI. Peripheral Component Interconnect.

PID. Process ID.

PRI. Primary Rate Interface.

PROFS. Professional Office System.

PVC. Permanent Virtual Circuit.

RAID. Redundant Array of Independent Disks.

RDBMS. Relational Database Management System.

RS-232. EIA standard specifying the electrical characteristics of low speed serial communications.

RTT. Round Trip Time.

SDEV. Standard Deviation.

SGN. Secondary Geo Network.

SMDS. Switched Multi-megabit Data Service.

SLIP. Serial Line Interface Protocol.

SMIT. System Management Interface Tool.

SOCC. Serial Optical Channel Converter.

SPOF. Single Point of Failure.

STP. Shielded Twisted Pair.

SSA. Serial Storage Architecture.

SVC. Switched Virtual Circuit.

TMSCSI. Target Mode SCSI.

TTL. Time to Live.

UDP. User Datagram Protocol.

UNI. User Network Interface.

UTP. Unshielded Twisted Pair.

VBR. Variable Bit Rate.

VSAT. Very Small Aperture Terminal.

Index

Symbols

Numerics

A

B

C

D

E

T

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-2018-01 Disaster Recovery with HAGEO and GeoRM
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

Disaster Recovery Using HAGEO and GeoRM

Disaster Recovery Using HAGEO and GeoRM

High availability through use of geographic clusters

Optimize performance for geographic mirroring

Step-by-step installation of HAGEO and GeoRM

GeoRM provides high availability for systems through data mirroring over geographic networks. HAGEO extends the function of GeoRM by adding tools that are capable of moving applications from one node to another in IP-based networks. With the combination of the two, HAGEO and GeoRM provide a quick and reliable recovery and backup from any possible site disaster.

This IBM Redbook is an essential guide to planning and implementing a successful HAGEO cluster with HAGEO or GeoRM Version 2.1. By using tested examples, hints, tips, and answers to frequently asked questions, this book provides in-depth coverage of the network types available for geographic mirroring and evaluates performance considerations for mirroring data over geographic networks.

The guides on planning, configuration, and administrative tasks will provide system engineers, service providers, and customers the technical insight necessary to design and successfully implement an HAGEO cluster.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by IBM's International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-2018-01 ISBN 0738416061